This page will not exist in the book.

Dedicated to Frederick Jelínek.
For all I've learned from you.

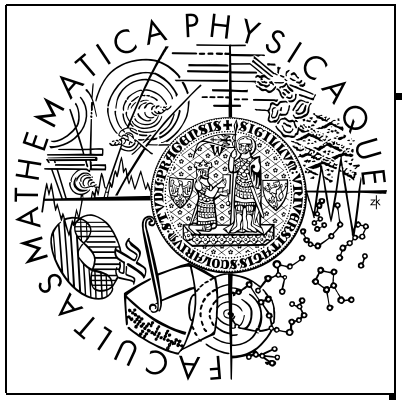
Hereby, I affirm that this thesis is solely my work and that all the literature used during its developement has been properly cited. I agree with its lending.

Prague, July 11, 2012

David Klusáček

. . . . . .

D O C T O R A L T H E S I S



David Klusáček

New Methods in Statistical Speech Recognition

Charles University in Prague
Faculty of Mathematics and Physics

Title: New Methods in Statistical Speech Recognition
Department: Institute of Formal and Applied Linguistics (ÚFAL)
Specialization: I-3 Computational Linguistics
Author: Mgr. David Klusáček
Advisor: Prof. Jan Hajič
Year: 2012

ÚFAL

Summary

Název Práce: Nové metody ve statistickém rozpoznávání řeči

Autor: David Klusáček

Katedra: Ústav formální a aplikované lingvistiky, MFF UK

Školitel: Prof. Jan Hajič, ÚFAL.

Abstrakt: Tato práce se pokouší identifikovat limity současných rozpoznávačů řeči a navrhnout metody jak jejich omezení překonat. Po historickém úvodu a popisu současného stavu je jako nejslabší článek řetězu prohlášen akustický front-end, zejména jeho činnost za zhoršených zvukových podmínek. Navrhované řešení, tzv. NUFIBA front-end, zahrnuje kompenzaci ozvěny, segmentaci zvuku na řečníka a pozadí, a průběžné sledování SNR, které v součinnosti s akustickým modelem zabraňuje lavinovému šíření chyb. Z důvodu nedostatku času jiz bohužel nedošlo k implementaci celého rozpoznávače řeči (i když některé části byly značně rozpracovány, například jazykový model založený na MMI třídách). Nové myšlenky tak byly vyzkoušeny pouze v jednodušším rozpoznávači fonémů.

Klíčová slova: Automatické rozpoznávání řeči, souvislá řeč, NUFIBA front-end, Jazykový model, Sluchová dráha, MMI, Shlukování, Slepá dekonvoluce, Časové a frekvenční maskování, Potlačení ozvěny, Rozpoznávání fonémů.

Podpora: Práce byla částečně podporována z následujících grantů:
GAČR 201/05/H014 and GAČR DG401/03/H047.

Title: New Methods in Statistical Speech Recognition

Author: David Klusáček

Department: Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics in Prague, Malostranské náměstí 25, 118 00 Praha 1.

Advisor: Prof. Jan Hajič, Institute of Formal and Applied Linguistics (ÚFAL).

Abstract: This works aims to identify limits of contemporary speech recognizers and tries to come up with methods that could push back the frontiers. After describing the state of the art, the weakest link of the chain has been identified in the acoustic front-end, especially when working in harsh acoustic conditions. NUFIBA front-end, the proposed solution, includes reverb compensation and speaker/background segmentation as well as continuous SNR monitoring which, thru cooperation with acoustic model, hinders from avalanche spreading of recognition errors. Owing to the lack of time, only a phoneme recognizer was finally implemented, although large blocks of originally intended word-based continuous speech recognizer were implemented and tested (such as the MMI-class based language model).

Keywords: Automatic Speech Recognition, Continuous Speech, NUFIBA Front End, Language Model, Auditory Pathways, Maximum Mutual Information, Clustering, Blind Deconvolution, Echo Suppression, Time and Frequency Masking, Phoneme Recognition.

Support: This work was partially supported from the following grants:
GAČR 201/05/H014 and GAČR DG401/03/H047.

Table of Contents

1 Introduction

A speech recognizer is a machine which automatically translates input speech sounds into a written output. First serious attempts to build such a machine date to 1960s [52, 9]. At that time, the approach to speech recognition was mainly rule-based[1], under the influence of general artificial intelligence of that time. As the rules were mostly human-written and language dependent, they required tremendous amount of linguists' work to be fully specified. Moreover, it turned out that linguistic theories did not cover language phenomena evenly, regarding to their occurrence. This led to poor performance because frequent non-grammatical constructs (like the same word repeated twice, sudden stop followed by complete change of the subject, etc.) confused the system awaiting the idealized language. Another cause of the low performance was naive acoustic front end. High quality microphones in quiet environment were required.

In 1970s, a group of IBM researchers led by Fred Jelínek came up with a revolutionary idea[2] of treating the problem from the standpoint of information theory. When compared to previous 'hierarchical rule-based approach', they employed relatively simple, yet powerful, idea of speech production — so called noisy channel model. Under plausible assumptions on the nature of probabilities involved, this can lead to hidden Markov models (HMMs) for which there exist effective decoding algorithms. Better than that, there also exist 'learning' algorithms, freeing linguists and phoneticians from creative but extremely laborious work of designing the rules. All that is needed, is long enough training sound and its transcription. Some systems require the text to be time aligned with the sound, and/or to be a phonetic instead of orthographic transcription, but these are implementation details. The learning also theoretically makes the system language independent, in the sense that the recognizer only needs to be retrained on new training data to acquire a new language[3].

In the meantime, special purpose small-vocabulary systems were developed. These systems used different approach, based on pattern matching via dynamic time warping (DTW). They were speaker dependent and originally could not process continuous speech, just separated words. Nevertheless, in tasks like digit and simple command dictation[4], they outperformed HMMs of that time [39]. When used on isolated words, they needed less memory and computational power, what made them favorable for cheap or portable devices. However, it was never fully solved how to adopt the method

[1] And sometimes highly hierarchical as in the case of HEARSAY-II system [25] which had separate levels for phonemes, syllables, words, syntax and semantics, all communicating via so called blackboard data structure.

[2] *Every time I fire a linguist, the performance of our speech recognition system goes up.*
— Fred Jelínek [7]

[3] Unfortunately, in the real world, it is not that easy because, for instance, languages with rich inflections like Czech, tend to have too many word forms (for which the original language modeling is unfeasible because of data sparsity problem). More sophisticated language modeling capability is needed in this case. Other languages, Chinese for instance, have distinct phonemes differing only by their F_0-trajectory. These would require different acoustic front end, retaining the F_0 information.

[4] For instance, in the late 1990s, there were cell phones capable to dial a number according to spoken callee's name. The number and the sound had to be already stored in the phonebook and the algorithm selecting the right entry was based on the DTW.

for large vocabulary continuous speech and it is essentially abandoned today — considering computational power of today's CPUs it has already lost its attractiveness even in cell-phone and toy market. Therefore, I will not treat it in this work at all.

For sake of completeness, it should be mentioned that there were also 'alternative' attempts, mainly by people from artificial neural networks community. The most notable person is Teuvo Kohonen who built what he called the 'Phonetic Typewriter' to demonstrate learning capabilities of his neural network, nowadays known as Kohonen's map.

Unfortunately, according to the papers I have read, I can't help myself from feeling that these people were uninformed about the state-of-the-art in speech recognition area. They were using inferior acoustic front ends, poor or no adaptation and usually no language model (sometimes they put weird devices instead of it, like rewriting systems correcting phoneme errors according to rules [24]). Moreover, they mostly did testing on their own data, sometimes very short data, which disabled any meaningful comparison with mainstream systems. Sometimes, their conception of how the testing and training should look alike is also noteworthy. In the 1991 Kohonen's paper [24] we can read:

> *Extensive experiments were performed for three male Finnish speakers in the speaker-dependent mode. For each speaker, four repetitions of a set of 311 words were used. Each set contained 1737 phonemes. Three of the repetitions were used for training, and the remaining one for testing.*

Although this is certainly better than using exactly the same data both for training and testing, it is quite close to it. It would, for example, alleviate co-articulation effects because these would be nearly same as in the training data, it would hide 'out-of-vocabulary word problem' and many others. Also, the data set is too small to support any definite conclusion. In this light, the word error rate (WER) of 6% seems more meaningless than amazing.

However, even in these papers, interesting ideas appear time after time. Only their implementation suffers from carelessness of their authors. But things are getting better, as we can see in a more recent paper [63], where they even used a standard (albeit quite small) corpus for evaluation and compared the results with standardly designed recognizer.

Some of these ideas were finally adopted by speech recognition community and experimental systems were built where neural networks are employed. They usually find their place in the front end of the recognizer [31], sometimes they are used in the language model [66]. The performance of these systems is at least comparable to the standard approach. But it has to be emphasized, that the heart of these systems still relies on HMMs or Graphical Models (GM) which can be regarded as HMMs' generalizations. So, after all, they are still within the probabilistic (noisy channel) framework.

There are several typical problems one has to struggle with, when writing a speech recognizer. In the probabilistic approach, the most notable problem is the problem of sparse training data. For the method to work well, we have to know the probabilities of events like 'the occurrence of a given triple of adjacent words at random place in the text'. The problem is that there are too many different triples. For English, vocabularies of 50000 words are considered adequate for recognition of common speech, so there are $50000^3 = 125 \cdot 10^{12}$ different triples. Even if we look aside from the difficulty

of storing so many numbers in computer's memory (which is in order of 10^9 numbers today) we have a problem how to acquire those numbers in the first place. We simply cannot hope to have long enough training data to estimate them by naive occurrence counting. Naive counting would lead to many triples receiving zero probability, which is certainly wrong. The main art here is thus in estimating the probability of events which were never encountered during the training[5]. It is called smoothing and has been in focus of researchers since the IBM-group times. Today, several powerful methods exist but, still, there is a certain room for improvement.

Another challenge is the acoustic front end. Reverberations in the room, background noise, different speakers with different voices all pose serious problem for the recognizer. Over the time, various methods (like cepstral mean subtraction and vocal tract length normalization) were developed to alleviate the influence of these effects. Nevertheless, I feel that we have more room for improvement here than in language modeling. Majority of current front ends are rather simplistic in comparison to the sophistication of other parts of the recognizer. For instance, they do not use segmentation (which has been common in image processing for years) therefore the speaker's voice is not separated from the background sounds, that can subsequently disrupt the recognition process. I will explain why I think it worths starting with improving front end in section 6.1. ▷ 176

Today, more than 30 years after its introduction, all commercial speech recognizers I am aware of, are based on the probabilistic method. They differ mainly in implementation details and in the way they tackle the above mentioned problems. The recognizer I am about to describe is not an exception. Novelty of my approach, lies in the way I deal with uncertain information. Most of current systems do not pay any attention to varying quality of the acoustic evidence from which they deduce what has been said. However, it is intuitively clear that if, for example, the door-bell rings in the middle of the word we should not give these sounds too high importance, and rather use context to recover the information about the words said.

1.1 Properties of the Recognizer

An automatic speech recognizer (ASR) can have various properties which characterize its abilities. Let us list them here. Each property can be either present or missing (or poorly implemented in which case it is not really missing but useless). Some of them are mutually dependent so there is actually less than 2^N kinds of speech recognizers. Note that this classification is only tentative and unofficial and is given here so we could fully appreciate broadness of the speech recognition problem.

(1) **Channel independence.** Early systems were channel dependent, requiring to use the same microphone and audio chain (that is amplifier and digitizer) for both training and testing. Often, recording in the same room was also needed to keep the reverberation patterns similar. All today's systems are channel independent to some extent, as they perform cepstral mean subtraction (introduced in section 5.1.1) or similar deconvolving process, which dereverberates the sound. Robustness ▷ 152

[5] In another words, we want to make judgments on things we have never seen based on those few we have seen. In broader sense, this could be regarded as one of many abilities the collection of we call the intelligence.

against (changing) background noise is also needed for good channel independence (especially in automotive applications). Note however, that other types of distortion are usually not treated at all. For instance it is known [53] that in telephone speech there is great distinction between electret and carbon microphones, the second one being slightly non-linear and, as such, causing harmonic distortion. In subtle applications, like speaker's voice verification, this must be accounted for. As for speech recognition, the HMMs are sufficiently robust not to be disturbed by that. Current research in this field focuses on background noise robustness and better dereverberation algorithms.

(2) **Speaker independence.** Speaker dependent systems are optimized for speech transcription of a single user. They tend to be more precise, provided there is enough training data[6]. On the other hand, they are less versatile. Vocal tract size differences among speakers cause that characteristic frequencies of phonemes vary among speakers. Fortunately, this can be nearly undone by technique called vocal tract length normalization (VTLN). Current systems are mostly speaker independent which is accomplished by VTLN (or a similar technique, like Mellin transform) and by acoustic model adaptation which copes with differences in pronunciation.

(3) **Continuous speech recognition capability.** By continuous speech I mean fluent spontaneous or read speech. Simple systems can only recognize isolated words, which means that the text has to be dictated word by word with sufficient pauses between them. Continuous speech systems are more complicated since they have to find proper splitting into words[7] and must cope with coarticulation effects. Pauses in a continuous speech occur as a part of plosive sounds[8] not between the words, unless the speaker makes them intentionally.

(4) **Large vocabulary.** Although no definite threshold exists, the system is said to be large vocabulary if it can recognize more than, say, 10000 words. Large vocabulary systems are usually continuous speech as well.

(5) **Domain independence.** Usually, even large vocabulary systems are not completely universal. A system for transcribing medical reports will undoubtedly work better if trained on real medical report dictations, than on fairytales. So every system today is domain dependent to some extent. However, there is theoretical possibility to have a system trained in many different domains with means enabling automatic selection of the right domain. There also exist systems for very restricted domains, such as voice control of car's accessories (like radio, navigation and windows), or systems for ticket reservations over the phone. These lack the ability to transcribe any speech. Instead, they follow a grammar (prepared by human experts) which can only generate domain relevant sentences. This makes the system more robust to noise (because there are fewer sentences among which the system has to select). The grammar is also used to reveal meaning of the commands.

(6) **Adaptability.** Adaptability refers to the system's ability to adjust itself to new conditions. Either it can be adapted on demand, for example, when a new user

[6] Which usually is only in case of politicians, actors and other people who unintentionally generate amounts of training data due to their jobs.

[7] Example taken from [9]: *'I scream'* ≈ *'ice cream'*

[8] p,t,k,b,d,g in English

is enrolled, generic acoustic models are shifted towards his style of pronunciation (after he reads certain amount of training text), or more sophisticated systems can be adapting themselves continually to fit speaker's actual speaking habits as closely as possible. Note however, that this spontaneous adaptation can cause more harm than good, as it can gradually destroy good models. Hence it is nontrivial to implement it right. The ability to add new words to the dictionary can also be understood as a kind of adaptation.

(7) **Real time processing ability.** Systems able to transcribe a speech at the speed it is dictated (hopefully with a small delay, too) are called real time. This is somewhat relative category as CPU speeds are still increasing. Most of today's systems can run in real time mode at the expense of higher error rates.

(8) **Awareness of errors.** Original ASR systems were unaware of their own errors. Although this is still common, there are methods now which can highlight suspicious words in the recognizer's output.

(9) **Command recognition ability.** One of often overlooked problems of ASR dictation programs is the fact that users not only want to dictate the text. They also want to edit it. It is not trivial, however, to recognize what has to be transcribed and what is meant as a command for the editor. This is comprehensively illustrated in [10]. Also, good dictation system should be able to delete unintentionally repeated words and be able to learn particular speaker's voice cues (which may indicate unpronounced commands).

(10) **Attention control.** A system without any attention control just tries to transcript everything it hears. Simple attention control usually has two control phrases, one of them turning the recognition system on (such as *'wake up'*) and the other turning it off (*'go to sleep'*). This is common today. More advanced attention control would include speaker's voice recognition so that it would ignore other people's voices. Also, the on/off phrases could be more flexible (so it would be also possible to use a sentence such as *'computer, let us return to the dictation'*) and more precise so that the system would not react to the user saying *'You better wake up, Joe.'* to his friend. At this level of sophistication, the attention control has to face similar problems as the command recognition. In fact, it can be implemented using already present command recognition module enriched by a speaker's voice recognizer. The ultimate attention system should be able to follow the dialog to the extent so that it would be aware whom the speaker is probably talking to, reacting to his voice only if the message was addressed to the computer.

(11) **Dialog capability.** Should the recognizer be a part of a larger system leading a spoken dialog with the user, other requirements arise on it. First of all, it has to be connected with the speech synthesizer to the extent it could detect its voice which must be 'subtracted' from the input sound so that the recognizer would ignore it. Secondly, there should be a subsystem for acknowledging and/or questioning the user during the time the ASR is listening. An appropriate questioning can be used to resolve misrecognized words, and acknowledging is important from psychological reasons, because users tend to get surprised and confused when the computer is listening to their monologue without interrupting them and without single breath for more than 2 minutes. Thirdly, there should be an appropriate attention control subsystem, especially good speaker tracking, so that the system would not react

to the voices of bystanders. Last but not least, the system should have low overall latency[9] so that the dialog could be naturally fluent.

1.2 Outline

This thesis is organized such that it reflects the signal travel inside the recognizer. So, after the necessary introduction into probabilistic speech recognition, which gives the reader general notion about the problems to be solved, chapters about front end, acoustic model and language model follow. Any theory needed along the way is introduced prior its first use in dedicated chapters (there is a chapter about signal processing and another one which introduces parts of probability theory we will need). Then, the components of the recognizer will be worked out, and the recognizer itself will be introduced in special chapter. Finally, tests, experiments and word error rates attained, will be presented.

I tried not to forward reference, except in cases where it is an informal note and the matter will be explained rigorously later. So, I hope that the book would be readable from the beginning to the end without much seeking, even for laymen. The mathematical knowledge required to understand this text roughly matches that of the second year of university (elementary calculus, basics of the set theory, intuitive understanding of the probability theory and general notion about computer programming).

However, the organization of the book I have chosen causes that my own results are intermixed with other ideas. For this reason, I clearly state what is mine in the beginning or at the end of each chapter.

[9] Normal latency of humans' speech recognition chain is being estimated to be about 170 ms, according to [16]. It can be higher, though, if we need longer context to decode the message.

2 Probabilistic Approach to Speech Recognition

In this chapter, classical probabilistic approach will be summarized. See [39] for excellent introduction. The beauty of the this approach is in the fact that it naturally leads to decomposition of the problem into acoustic and language model parts, under very general assumptions.

2.1 Noisy Channel Model

Let the input sound be encoded as finite sequence of symbols $A = \langle a_0 \dots a_n \rangle$ from a discrete[1], possibly very large, alphabet. For the moment, let us imagine that the a_is are directly the instantaneous amplitudes of the waveform as they are stored in the `WAV` file. In this case each a_i would be an integer from -32768 to 32767. Later on, more practical encoding will be introduced by insertion of the acoustic front end. But in the general setup, any encoding can be theoretically used at the price that less clever choice complicates estimation of the probabilities involved. The output from the recognizer will be finite sequence $W = \langle w_0 \dots w_m \rangle$ of `ASCII` codes making up the output text.

Now, one possible formulation of the recognition problem would be to find W maximizing $\Pr(W|A)$, which is the probability[2] of W being the correct transcription of the sound, provided that the sound was A. It is meant that we estimate this probability from the training data. Note that it inherently admits several possible transcriptions of the same sound to be present in the training set[3]. To make the estimation of the probability computationally feasible we rewrite it using the Bayes' rule, getting:

$$\widehat{W} = \arg\max_W \Pr(W|A) = \arg\max_W \frac{\Pr(A|W)\Pr(W)}{\Pr(A)} = \arg\max_W \Pr(A|W)\Pr(W) \quad (1)$$

This way, we obtained a decomposition into the language model $\Pr(W)$ and the acoustic model $\Pr(A|W)$. This not only splits the original problem into two easier ones, but also allows to train the language model on text-only data, enabling us to use much longer data for more precise probability estimation.

This decomposition also leads to the diagram shown in figure 1, so called noisy channel model of speech processing. First, the speaker (on the left) generates what ▷ 15

[1] This is completely general as the recognizer will in fact be a program running on a digital computer, inside which everything has to be quantized anyway.

[2] I will use axiomatic approach [38] to probability throughout this work. Nevertheless, frequencist viewpoint works in this case as well, if we assume $m < n$ (that is, the number of decoded letters to be less than the number of samples in the waveform, which is obviously unrestrictive, in the real world). Under this assumption, there is only finite number of possible W-sequences for the given observation A, and, in principle, they could be counted to give out the probability estimate. Practically, though, this is impossible, because huge amount of training data would be required to do so. I'd rather leave to philosophers weather this probability 'exists' or has any sense at all, when it cannot be practically measured. Axiomatic approach does not depend on it.

[3] Practically, there will hardly ever appear the same sound twice, however.

he intends to say. At the level of abstraction we are now, let us suppose that whole W is generated at once in the beginning, so we could speak of its probability, then it is transformed by the speaker to the movement of his articulation organs and finally it is radiated into space as a sound wave S_1. The hypothetical channel starts inside the speaker's head, because the noise[4] N_1 affects how W will be transformed into S_1. Then, another noise N_2 (this includes noise from the environment, reverberations, noise of the microphone and amplifier and dynamic range of A/D converter) affects how the sound wave changes into the (digital) electrical signal S_2. Finally, the acoustic front end usually discards[5] some information, which has the same effect as yet another noise N_3, changing S_2 to A — the channel's output. However complicated the channel may be, we are not interested in these details because we can fully describe its operation by conditional probability distribution $\Pr(A|W)$, where W is the message on its input and A is what the computer gets from its output. This A is then sent to the decoder which, according to (1), selects the most probable channel's input $\widehat{W}$, which could have caused the observed A. If we knew true distributions $\Pr(W)$ and $\Pr(A|W)$ exactly, this setup would minimize the probability of recognition error as I will show later. Also, the speaker and channel independence (with noise robustness) would be automatically included. This is because the ideal probabilities must account for every single utterance as if it appeared under all possible N_2-noise conditions (taking into account probability of occurrence of the particular noise condition, too).

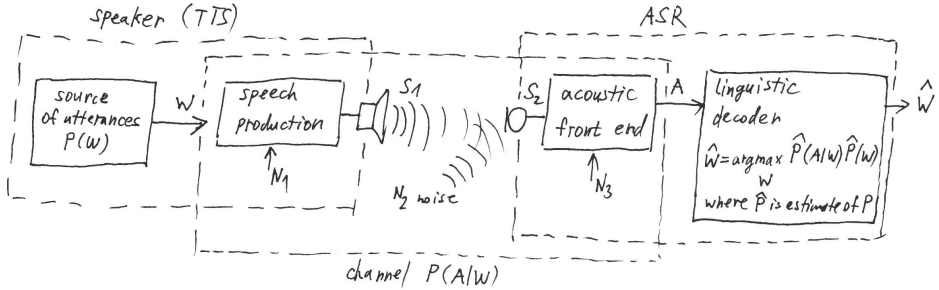


Fig. 1. *Noisy channel model. Those fainting at the idea that something as nasty as channel could be extending into someone's head should picture a text-to-speech synthesizer in place of the poor wretch.*

However, in a practical situation, we don't have the probabilities $\Pr(A|W)$ and $\Pr(W)$. All we can hope for, is their estimate based on the training data. To make this possible, however, the number of free parameters involved has to be substantially reduced. Until now, I took these probabilities as being described by their distributions, which could, in principle, be estimated from the respective histograms. Unfortunately, already $\Pr(W)$ leads to astronomical number of free parameters, when interpreted this way. For outputs of length 80 (which is one line on a typical computer terminal) we

[4] This noise 'simulates' various speaking styles and pronunciations. The noisy channel model originally came from Claude Shannon's theory of communication, which is mathematically describing communication of machines, like modems and such. In this theory, in place of N_1, there would be a modulator, proper selection of could counteract the effects of noise in the channel so that it would be possible to recover original W with arbitrarily high probability in the receiver. In speech recognition, we are in a situation where the modulator is already given to us, we do not know exactly how it works and also different people modulate W in different ways, so I decided to call it noise here.

[5] Note that we still consider no acoustic front end, therefore $A = S_2$.

would have $2^{640} \approx 4.5 \cdot 10^{192}$ different strings, each demanding its probability to be assigned to it. The situation with $\Pr(A|W)$ gets even worse.

To overcome this, we have to make an educated guess on processes generating these distributions. With smart selection, we can hope that the number of free parameters will be small enough to make their estimation from the training data possible, yet allowig the model to approximate true distributions sufficiently well to be usable in the decoder. This is called acoustic and language modeling and will be treated in sections 2.4 and 2.3. ▷ 20 ▷ 17

However, when approximating the true distributions, we may expect that we will lose something, too. For instance, optimality of the method (minimum probability of an error) presupposes exact distributions. It would also be naive to think that our model of $\Pr(A|W)$ would be able to incorporate all possible distortions of S_2 due to noises N_2. Even if it could, there would be another trouble with its training — we would have to artificially generate all possible reverberations and noise conditions out of our training data, which would prolong the training by several orders of magnitude.

To overcome this, the acoustic processor is inserted into the recognition chain at the place shown in figure 1. Its purpose is a reduction of dimensionality of S_2 in a clever way, which enhances those features useful for a speech recognition and attenuates the unwanted ones, such as noise. Another appreciated property would be the reduction of the variability of A provided that the same W is being sent over the channel multiple times, spoken by different speakers (N_1) and under different conditions (N_2).

Although some information about S_2 is irreversibly lost, this stage is necessary in any practical system, if we want to estimate $\Pr(A|W)$ from attainable amount of training data. In the next section, simple front end will be sketched to give the reader an initial idea. More complicated front ends will be discussed later in depth because the front end's quality is crucial for overall performance — what information is lost there is lost forever (as follows from theorem 3.2.31). ▷ 55

2.2 Simple Front End

In the early days of speech recognition, the speech recognizer consisted of digital computer connected to an 'acoustic processor'. This was special analog device[6] consisting of an input amplifier and a bank of about 20 bandpass filters spanning the speech frequencies of 300 Hz to 3 kHz. The output of each individual filter was rectified to yield an estimate of the amplitude envelope of the given band. These envelope estimates were then sampled hundred times a second and converted into (digitally encoded) numbers by the A/D converter (see fig. 2). These numbers were then sent to the computer. ▷ 17 So, every 10 ms, we had 20, say, 10-bit numbers (which were treated as real-valued in theory, allowing us not to think about resolution of the A/D — the limited precision could be simulated as noise, if needed). Each 20-tuple is called a frame. The sequence of these frames then constitutes A — the output from the acoustic processor, as it is depicted in fig. 1.

Sometimes, instead of plain amplitudes in the frame vector, their logarithm was used to match human perception of the sound intensity more closely. Sometimes, not

[6] Today, computers are fast enough to do all this signal processing digitally — the only material part of the acoustic processor which survived till now is the A/D converter located in the sound chip.

only the current output of the filter bank but also its difference from the previous one was used (so there would be 40 numbers instead of 20). Sometimes the vector was normalized and the energy of the frame was stored as another number sideways, etc. But these are details that used to vary from system to system (and all of them could be easily implemented in software using the hardware from fig. 2).

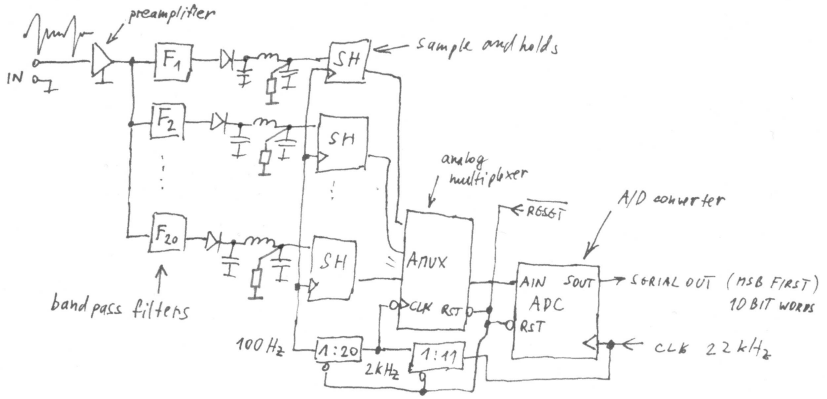


Fig. 2. *Rudimentary front end*

To make the input data A even more compact, there is a possibility, often used in the early times, when computers were not fast enough to do the decoding with full frame vectors. It is a vector quantization. If we looked at the 20 (or 40) dimensional space while being filled with points coming out of the front end processing the real speech, we would see that these points have a tendency to gather in clusters instead of being scattered uniformly. Often, the cluster would approximately correspond to certain characteristic sound. By use of a standard algorithm, such as K-means clustering[7] we can identify these clusters, eventually subdividing the larger ones, too. Usually, it is sufficient to partition the space into 200 to 300 regions using points obtained from 5 to 10 minutes of speech. Then, each frame of A would be just the label of the region into which the feature vector belongs. More elaborate clustering algorithms like Kohonen's maps or generative topographic mappings [48] may lead to better results.

The clustering leads to a very small acoustic alphabet and therefore to easier modeling of $\Pr(A|W)$. Its disadvantage is information loss, causing inferior performance. However, it has been demonstrated [54, 35, 27] that with careful selection of clusters, the performance competitive with unclustered systems is attainable. Nevertheless, this approach was finally abandoned.

2.3 Language Model

First, we need more realistic encoding to be used for W. From now on, I will regard w_is to be whole words, not just single letters. This means that we have to limit ourselves to a finite lexicon of words. Let it be represented by a set of words V. So, shall the systems hear a word which is not in it, a recognition error will occur. An advanced

[7] Using Euclidean distance as its metric is acceptable.

system may be able to detect this condition, allowing the user to add this new word into the system's dictionary manually, by providing its correct spelling.

Second, with this encoding we may write the probability of the given word string $W := \vec{w} = \langle w_0 \dots w_m \rangle$ to be

$$\Pr(\langle w_0 \dots w_m \rangle) = \Pr(w_0) \prod_{i=1}^{m} \Pr(w_i \mid w_{i-1}, \dots, w_0) \tag{2}$$

which comes directly from the Bayes' formula. There are many equivalent factorizations differing by the order in which w_is are taken. This one has an advantage of obeying the natural order of the words. It enables the decoder to start its work before the whole utterance has been recorded.

Strictly speaking, formula (2) only assigns the probability to the word sequences of fixed length $m+1$ and, as such, it should be written as a conditional probability $\Pr(w_0, \dots, w_m \mid m+1)$. Consequently, we cannot directly compare the probability of strings of different lengths, unless we knew $\Pr(m+1)$, the probability distribution of these lengths. This omission is often overlooked in the area of speech recognition. It can be usually tolerated because the acoustic model $\Pr(A|W)$ assigns high probability only to those W-strings whose length is similar to what has been said. Too long and too short strings simply don't match the acoustic signal A. It is also natural to expect that $\Pr(m+1)$ will be roughly equal for all values of m close to m_0, the average length of the hypothesis W. Thus, setting $\Pr(W) := \Pr(w_0, \dots, w_m \mid m+1)$ instead of the correct $\Pr(W) = \Pr(w_0, \dots, w_m \mid m+1) \Pr(m+1)$ does not make much difference in (1), unless for very short utterances. But note that $\Pr(W)$ does not sum to 1 then. ▷ 14

If we desired to have correct expression for $\Pr(W)$ we could construct the following probability space[8] with the help of string terminating symbol $\diamond$.

$$\begin{aligned} \Omega &:= \{\langle w_0 \dots w_m \rangle \mid \forall i < m : w_i \in V \ \&\ w_m = \diamond\} \\ \Pr{}_\Omega(\langle w_0 \dots w_m \rangle) &:= \prod_{i=0}^{m} \Pr(w_i \mid w_{i-1}, \dots, w_0) \end{aligned} \tag{3}$$

where $\Pr(w_0 \mid w_{-1}, \dots, w_0) := \Pr(w_0)$. Ω can be imagined as an infinite tree with finite branching, each node representing the word history $\langle w_0 \dots w_i \rangle$. Unless $w_i = \diamond$, the node has $\#V+1$ continuations ($\diamond \notin V$). The probability of strings from Ω is defined by means of conditional distributions $\Pr(w_{i+1} \mid w_i, \dots, w_0)$ — every finite path ending with a terminal leaf $\diamond$ corresponds to a string from Ω and vice versa. It is an easy exercise to prove that $\sum_{\omega \in \Omega} \Pr_\Omega(\omega) = 1$.

But let us return to our language modeling effort using the original formula (2). Now, we still don't have a model. Apart from the fact that we cannot represent lexically incorrect strings in W anymore, (2) is still fully general, and still requires too many parameters. But, realizing that the given word is less dependent on words which are farther from it in the history we could assume that $\Pr(w_i \mid w_{i-1}, \dots, w_{i-K})$ would be a good approximation to $\Pr(w_i \mid w_{i-1}, \dots, w_0)$, for moderate values of K like 2 or 3.

[8] Probability space will be rigorously defined in 3.1.1. Until then, we get by with our intuition. ▷ 39

The most common choice here is 2, for which we get so called trigram model assigning the probability of $\vec{w}$ to be

$$\Pr(\vec{w}) = \Pr(\langle w_0 \dots w_m \rangle) := \Pr(w_0) \Pr(w_1 \mid w_0) \prod_{i=2}^{m} \Pr(w_i \mid w_{i-1}, w_{i-2}) \quad (4)$$

Immediate generalization of this would be to come up with a partitioning function $\Phi : V^* \to \mathbb{N}$, which takes an arbitrary string $\vec{w}$ as an argument, returning the class number this string belongs to. Then the model would look the following way

$$\Pr(\vec{w}) := \Pr(w_0) \prod_{i=1}^{m} \Pr(w_i \mid \Phi(\langle w_0 \dots w_{i-1} \rangle)) \quad (5)$$

This model is really a generalization of the trigram model, which can be demonstrated by defining $\Phi(\langle \dots, b, a \rangle) := a + \#V \cdot b$ and $\Phi(\langle a \rangle) = a$, where $\#V$ is a size of the vocabulary and the words are identified with their ordinal numbers in the vocabulary V (numbering starts by 0), so we would be able to do computations with them. Both models yield to identical $\Pr(\vec{w})$ under this choice of Φ, hence the class-based one can simulate the trigram one and is therefore its generalization.

The partitioning function Φ can be very complicated (for instance, there could be whole parser hidden in it [20]). It can be either given by a linguist expert, or it
207 ◁
can even be generated automatically as I will show in chapter 8. But let us return back to the trigram model, which is surprisingly successful despite its simplicity. Even very complicated selections of Φ usually do not outperform it significantly. When it is accounted for memory and CPU demands of the respective models, the trigram model often comes out as a winner and that is why it is still in use in many commercial systems.

Now, for the trigram model, we have to justify how do we determine values of $\Pr(w_0)$, $\Pr(w_0 \mid w_1)$, and $\Pr(w_0 \mid w_1, w_2)$ for all possible triples $\langle w_0, w_1, w_2 \rangle \in V^3$. Obviously, we want to estimate them from the training data. The simpliest way of doing that, would be to use frequetists' notion of probability. Let the sequence $T = \langle t_0 \dots t_N \rangle$, $t_i \in V$ be the training text. Then, let

$$c(v_1, \dots, v_n) := \#\{ k \mid \exists\, a_1 \dots a_k, b_1 \dots b_l : \langle a_1, \dots, a_k, v_1, \dots, v_n, b_1, \dots, b_l \rangle = T \} \quad (6)$$

that is the number of times the n-tuple appeared in the training data. Using it, we can set[9]

$$\Pr(w_0 \mid w_1, w_2) := f(w_0 \mid w_1, w_2) = c(w_2, w_1, w_0)/c(w_2, w_1, \bullet) \quad (7)$$

where f is called the frequency of occurrence (of an event in the data T). Analogical definition would be used for $\Pr(w_0)$ and $\Pr(w_0 \mid w_1)$. Although it seems fairly sound, it is incorrect. The flaw is hidden in the fact that many possible triples will never appear in the training data. Note that for $\#V = 50000$ we have $125 \cdot 10^{12}$ triples. So, unless the training text is longer than that, at least one triple is always uncovered by the training data. This is a problem because it can well happen that the uncovered

[9] $c(y, \bullet)$ stands for $\sum_x c(y, x)$, where x goes over the whole range in question.

triple later appears at the input of the system. But our model states that it has zero probability. As the recognition is governed by formula (1), we can easily see that the
▷ 14
system would never be able to output correct transcription in this case.

The remedy of this situation is called *smoothing*. It works by taking some amount of probability from the observed triples, redistributing it to the unobserved ones. The most simple solution would be to define

$$\Pr(w_0 \mid w_1, w_2) := \alpha f(w_0 \mid w_1, w_2) + (1 - \alpha)/\#V \quad (8)$$

for suitable $\alpha \in (0, 1)$. But we can do better than that. Observing that for two probability distributions P_1 and P_2, $\alpha P_1(w) + (1 - \alpha) P_2(w)$ is also a probability distribution (obviously, it is non-negative as P_1 and P_2 were, and it sums to 1 as well), we can use the fact that if $\langle w_0, w_1, w_2 \rangle$ was not in the training data, then perhaps $\langle w_0, w_1 \rangle$ was. This line of thought brought us so called linear smoothing, given by

$$\Pr(w_0 \mid w_1, w_2) := \alpha f(w_0 \mid w_1, w_2) + (1-\alpha)\Big(\beta f(w_0 \mid w_1) + (1-\beta)\gamma f(w_0) + \frac{1-\gamma}{\#V}\Big) \quad (9)$$

The only trouble now, is how to set the coefficients α, β, γ. Any values in $(0, 1)$ will lead to valid distribution, but obviously we want to select those leading to the best performance. However, this would be too complicated in general, because the performance of the whole ASR depends on acoustics as well. Instead of it, we will try to set α, β, γ so as to maximize the probability of the data. But this has to be done carefully. Would the data over which we maximize the probability (9) be T, the optimal α would be $\alpha = 1$, that is no smoothing at all (this is because we would not encounter missing triple in T and therefore the best estimate of the probability would be the most detailed one). So we need extra data H, on which we would maximize α, β, γ, while keeping the underlying frequencies trained on T fixed. Once having it, we can merge T with H and reestimate the frequencies to obtain better precision. Alternatively, we can split the data several times to different (T, H) pairs and obtain different estimates of α, β, γ. These would then be averaged to get final α, β, γ (so the random glitches due to unfortunate splits should be averaged out to some extent). Although this smoothing model works acceptably, more sophisticated methods will be presented later, as it was empirically observed that the quality of smoothing often played the most important role in the whole system.

2.4 Acoustic Model

Acoustic modeling tries to assume something plausible about the probability $P(A|W)$ so that the number of its free parameters would become so small that it could be estimated from obtainable amount of training data.

We will express this plausible assumption in a form of non-deterministic generative model. This model, given a word transcript W, will generate possible acoustic sequence A with probability $P(A|W)$. It may be regarded as a primitive physical model of speech production which does not take actual mechanisms of pronunciation into account. But even though it does not describe speech production exactly, it can still provide sufficiently good fit, provided that it will have several times more parameters than an ideal physical model would require.

Let us first consider models of single phonemes and words sequenced from these phonemes. Commonly, they are based on so called *Hidden Markov Models (HMM)*. HMM can be intuitively described as a directed graph with nodes called *states* and arrows labeled by *transition probabilities* and *output distributions*. HMM generates its output by performing a random walk along these arrows, starting from certain initial state. In each state, it randomly selects the next arrow to go along, in accordance with its transition probability (among all arrows leaving that state). On each traversal of the arrow, it emits single letter $a \in \mathcal{A}$ by randomly drawing it from an *output probability distribution*. Each arrow has its own output distribution. Parallel to normal arrows, there may also be special *null arrows* which do not emit any letter and only change the state. For technical reasons they are not allowed to form a (directed) cycle. Normal arrows can form a cycle and there may be even loops (which means that the arrow may point to its own starting node). Look at fig. 3 for a typical model of single phoneme. Note that there are two paths — the short one and the long one — which can capture normal and abnormal pronunciations. Also notice loops and null transitions, which can prolong or shorten the duration of a phoneme, thereby simulating different speaking rates. Note that fig. 3 does not specify any values of transition and output probabilities. These are supposed to be obtained by training.

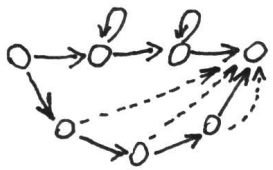


Fig. 3. *Phoneme HMM*

This phoneme model serves as a building block from which we can compose HMM models of words by simple concatenation, as in fig. 4. Only the structure is repeated — probability tables remain shared among different instances of the same phoneme. Hence they require constant amount of parameters regardless of richness of the vocabulary. Typically, stressed and unstressed phones are distinguished, which leads to a phonetic alphabet of size around 100, in case of English [39]. So we have 100 phonetic models, each consisting of nine output arrows, each of them capable of emitting one of about 250 distinct characters of the acoustic alphabet $\mathcal{A}$. This yields less than 230 thousand parameters that have to be estimated by training. This number may be further reduced by so called *tying*. For instance, we can notice that plosive phonemes have similar beginning (that is a silence) so we can decide that we make them share the output distribution of their first arrow.

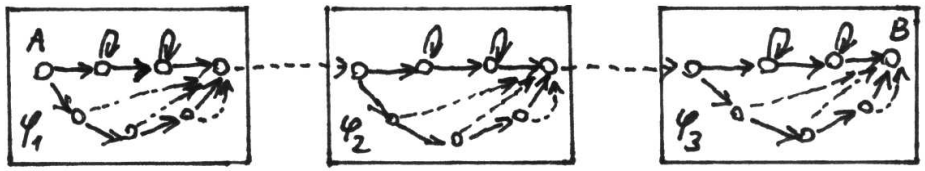


Fig. 4. *Hidden Markov Model for a word made up from three phonemes $\varphi_1\,\varphi_2\,\varphi_3$. The only entry point is A, the only exit is B. The respective phoneme models have identical structure but (in general) different transition and emission probabilities. The phoneme models are connected via null arrows to form a word.*

Although the probabilities can be learned from training data, the structure is given in advance and does not change by learning. Before proceeding, let us write formal definition of HMM, so that we could use this formalism in description of decoding and learning.

2.4.1 Definition *(Discrete) Hidden Markov Model (HMM)*
HMM is $\langle \mathcal{A},\ S,\ s_0,\ p : (S \times S) \to \mathbb{R},\ q : (S \times S \times (\mathcal{A} \cup \{\varepsilon\})) \to \mathbb{R} \rangle$, where $\mathcal{A}$ is a finite output alphabet, S is a finite set of states, $p(x|y)$ specifies probability of selecting state x when we stand in state y and $q(a \mid y \to x)$ is a probability of emitting letter $a \in \mathcal{A} \cup \{\varepsilon\}$ while going along the arrow from y to x. The special value ε means 'no output' and there must not exist any sequence of states $s_1, \ldots, s_{n-1}, s_n = s_1$ such that $q(\varepsilon \mid s_k \to s_{k+1}) > 0$, which means that ε-capable arrows must not form a cycle. Finally, $s_0 \in S$ is the starting state.

2.4.2 Note Equivalently, we could have defined HMM on a multigraph[10] with probability of going from y to x along the edge labeled by a being

$$r(a, x \mid y) := p(x \mid y)\, q(a \mid y \to x) \tag{10}$$

It is also convenient to assign names $e_1, e_2, \ldots, e_T$ to all arrows and define functions $A(e) := a$, $L(e) := y$, $R(e) := x$ and $P(e) := r(A(e), R(e) \mid L(e))$ for all $e = y \xrightarrow{a} x$. The set of all arrows in the HMM will be denoted by E. I will use this notation when appropriate.

2.4.3 Note Discrete HMM can be generalized to *continuous HMM*, whose output alphabet $\mathcal{A}$ is a subset of $\mathbb{R}^K$. The letters of $\mathcal{A}$ are real-valued frame vectors as supplied by the front end and $q(a \mid y \to x)$ is probability density instead of distribution. It is usually modeled as linear combination of several multidimensional Gaussians, whose parameters and weights have to be obtained by training. Some parameters can be tied, for instance we may share covariance matrices among all Gaussians in a phoneme. As my system uses discrete alphabet I will not go into continuous HMMs any deeper. Consult [39], if interested.

2.4.4 The Trellis

Let us draw all states of an HMM as a column of vertices, sorted such that null-arrows can only point downwards and let's draw these arrows, too. Then, copy this column N-times, obtaining $\#S$ by N grid of vertices x_i^t, where $i \in S$ and $t \in \{0, \ldots, N{-}1\}$, indexing the time. Finally, connect adjacent columns by arrows such that x_i^t will be connected with x_j^{t+1} by an arrow labeled with a and r if and only if $r := r(a, x_j \mid x_i) > 0$. This unrolled representation of HMM is called the *trellis*.

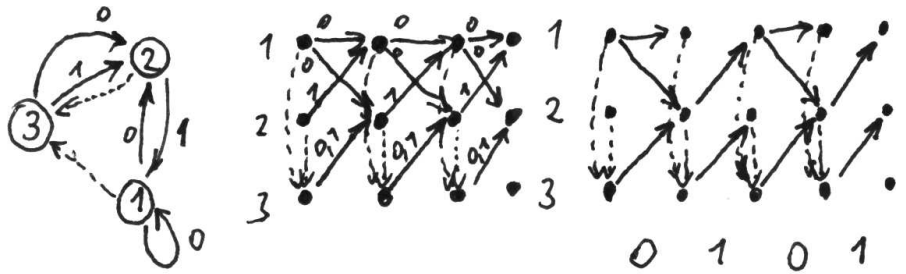


Fig. 5. *Simple HMM generating strings over the alphabet $\{0, 1\}$, its trellis for 3 output symbols and, on the right, there is its trellis punned according to the output string 0101. Probabilities of individual arrows are not explicitly depicted to keep the drawing readable.*

[10] Graph, where multiple arrows between two nodes are allowed.

It allows to simulate any N-step computation of the HMM simply by following the arrows and emitting their associated symbols along the way, starting in the initial state $x^0_{s_0}$. More importantly, it can be used to compute a probability that a given observation $a_0, \ldots, a_{N-1}$ was generated by the HMM. For this, we first need to *prune* the trellis such that the only remaining arrows will be null-arrows and those that emit symbol a_t between the states x^t_i and x^{t+1}_j for all i, j and t. See fig. 5 for a simple HMM and its pruned trellis. Then, we will define $\mathcal{T}(\vec{a})$, the set of all edge-paths thru the trellis, pruned by acoustic observation $\vec{a}$. That is, each element of $\mathcal{T}(\vec{a})$ is a sequence $\vec{e} = \langle e_0 \ldots e_{\#\vec{e}-1} \rangle$ such that $R(e_i) = L(e_{i+1})$ and $A(e_i)$ is either ε or a_k, the later iff e_i was k-th output producing arrow. For technical reasons which will become clear later, the last arrow in each path thru the trellis must be output-producing. This is also depicted in fig. 5 as missing null arrows in its last column. As the trellis is finite and we demanded that ε-arrows cannot form a cycle the set $\mathcal{T}(\vec{a})$ will also be finite. If we allowed cycles, $\mathcal{T}(\vec{a})$ would become infinite[11].

2.4.5 Definition

$\mathcal{T}^z_s(\vec{a})$ will stand for the set of all $\vec{a}$-compatible paths from x^0_s to $x^{\#\vec{a}}_z$. If z is omitted, the final state is unconstrained.

$$\begin{aligned} \mathcal{T}_s(\vec{a}) &:= \{\vec{e} \in \mathcal{T}(\vec{a}) \mid L(e_0) = s\} \\ \mathcal{T}^z_s(\vec{a}) &:= \{\vec{e} \in \mathcal{T}(\vec{a}) \mid L(e_0) = s \;\&\; R(e_{\#\vec{e}-1}) = z\} \end{aligned} \tag{11}$$

The probability that the HMM will follow a specific path $\vec{e} \in \mathcal{T}$, provided that it has started in state $L(e_0)$, is $\prod_{k=0}^{\#\vec{e}-1} P(e_k)$. Now we can derive the probability that the string $\vec{a}$ was generated by an HMM starting from s_0 and ending in state k, such that the last arrow taken (if any) has been output producing. Let us designate this probability by $P(a_0 \ldots a_{t-1}, k \mid s_0)$ or shortly as $\pi_t(k)$, where $t = \#\vec{a}$. Clearly, $\pi_0(k) = 1$ just for $k = s_0$. Because individual paths thru the trellis are disjoint events, the probability of taking any of them is sum of individual probabilities and we can write

$$\pi_{\#\vec{a}}(k) = \sum_{\vec{e} \in \mathcal{T}^k_{s_0}(\vec{a})} \prod_{i=0}^{\#\vec{e}-1} P(e_i) \tag{12}$$

For each path $\vec{e}$, the last outputting arrow and its immediately preceding null arrows can be factored out, yielding to

$$\pi_t(k) = \sum_{n \in S} \left(\sum_{\vec{e} \in \mathcal{T}^n_{s_0}(\langle a_0 \ldots a_{t-2} \rangle)} \prod_{i=0}^{\#\vec{e}-1} P(e_i) \cdot \sum_{\vec{f} \in \mathcal{T}^k_n(\langle a_{t-1} \rangle)} \prod_{j=0}^{\#\vec{f}-1} P(f_j) \right) \tag{13}$$

Recognizing $\pi_{t-1}(n)$ in the above sum we obtain

$$\pi_t(k) = \sum_{n \in S} \pi_{t-1}(n) \sum_{\vec{f} \in \mathcal{T}^k_n(\langle a_{t-1} \rangle)} r(a_{t-1}, k \mid f_{\#\vec{f}-1}) \prod_{j=0}^{\#\vec{f}-2} P(f_j) \tag{14}$$

[11] There is no fundamental obstacle to develop a theory even for cyclic null-arrows. However, without them, the formulas come out shorter and as we do not need cycles in our speech related HMMs, it is better to forbid them altogether.

which can be rewritten as

$$\pi_t(k) = \sum_{n \in S} \pi_{t-1}(n)\, \check{r}(a_{t-1}, k \mid n) \tag{15}$$

where $\check{r}(a_{t-1}, k \mid n)$ is a probability of taking any number of null steps, starting in state n, followed by output producing step into the state k. For HMM without null arrows $\check{r}(a_{t-1}, k \mid n) = r(a_{t-1}, k \mid n)$. For general case, we need the following notation for the set of all null-paths that can be prolonged by an outputting arrow:

$$\mathcal{N}^m_n := \{\langle e_0 \ldots e_{k-1} \rangle \mid \exists\, e_k \in E : A(e_k) \neq \varepsilon \;\&\; \langle e_0 \ldots e_k \rangle \in \mathcal{T}^m_n(\langle A(e_k) \rangle)\} \tag{16}$$

According to (14), general $\check{r}(a_t, k \mid n)$ can be computed as follows:

$$\check{r}(a_t, k \mid n) = \sum_{m \in S} r(a_t, k \mid m) \underbrace{\sum_{\vec{e} \in \mathcal{N}^m_n} \prod_{i=0}^{\#\vec{e}-1} P(e_i)}_{\eta(m \mid n)} \tag{17}$$

Note that $\eta(m|n)$ is not a probability because it may sum to more than 1. The correct probability is formed only after this is multiplied by $r(a_t, k \mid m)$. Note that (17) does not require null arrows to be acyclic, but if they are not, we have to sum over (countably) infinitely many paths. Recurrence (15) can be rewritten as follows:

$$\begin{aligned} \pi_t(k) &= \sum_n \pi_{t-1}(n) \sum_m r(a_{t-1}, k \mid m)\, \eta(m \mid n) \\ &= \sum_m r(a_{t-1}, k \mid m) \underbrace{\sum_n \pi_{t-1}(n) \eta(m \mid n)}_{\alpha_{t-1}(m)} = \sum_m r(a_{t-1}, k \mid m) \alpha_{t-1}(m) \end{aligned} \tag{18}$$

To treat the acyclic case, let us define an ordering $<$ on the set of states S, such that $a < b$ iff there is a null-path from a to b. This condition only enforces partial ordering, so we $<$-bind other elements of S make it a linear ordering. This is the same ordering that is used to draw trellis, with $<$-higher elements of S drawn lower on the page. In acyclic case, η can be easily determined using the following recurrence, which follows from (17):

$$\begin{aligned} \eta(n \mid n) &:= 1 \\ \eta(m \mid n) &:= \sum_{n < k \leq m} \eta(m \mid k)\, r(\varepsilon, k \mid n) \quad = \sum_{n \leq k < m} r(\varepsilon, m \mid k)\, \eta(k \mid n) \end{aligned} \tag{19}$$

Notice that $\eta(a|b) = 0$ for $a < b$. Substituting this into (18) we get:

$$\begin{aligned} \alpha_t(m) &= \sum_{n \leq m} \pi_t(n)\, \eta(m \mid n) = \pi_t(m) + \sum_{n < m} \pi_t(n)\, \eta(m \mid n) \\ &= \pi_t(m) + \sum_{n < m} \pi_t(n) \sum_{n \leq k < m} r(\varepsilon, m \mid k)\, \eta(k \mid n) \\ &= \pi_t(m) + \sum_{k < m} r(\varepsilon, m \mid k) \sum_{n \leq k} \pi_t(n)\, \eta(k \mid n) \end{aligned} \tag{20}$$

Comparing the last formula with the first one we arrive to the following recurrence:

$$\begin{aligned}\alpha_{t-1}(m) &:= \pi_{t-1}(m) + \sum_{k<m} r(\varepsilon, m \mid k)\,\alpha_{t-1}(k)\\ \pi_t(k) &:= \sum_m r(a_{t-1}, k \mid m)\,\alpha_{t-1}(m)\end{aligned} \tag{21}$$

where $\pi_0(s_0) := 1$, and 0 for all other states, and $\alpha(m)$ is self-initializing as it is evaluated from topmost to bottommost states m. Note that although π is probability distribution, α is not.

Total probability that N-tuple $a_0, \ldots, a_{N-1}$ was generated is then $\sum_{k\in S} \pi_N(k)$.

2.5 Decoding

The decoding is simply a search for the most probable path thru the trellis that could have generated the observed data $a_0, \ldots, a_{N-1}$, that is

$$\vec{\hat{s}} := \arg\max_{\vec{s}} P(\vec{s} \mid s_0, a_0, \ldots, a_{N-1}) = \arg\max_{\vec{s}} P(\vec{s}\ \&\ a_0, \ldots, a_{N-1} \mid s_0) \tag{22}$$

where s_0 is the initial state, $\vec{s} = s_1, \ldots, s_K$ is a state sequence and $K \geq N$. For a sequence without null transitions we have $K = N$ and

$$P(s_1 \ldots s_N\ \&\ a_0 \ldots a_{N-1} \mid s_0) = \prod_{k=0}^{N-1} \frac{p(s_{k+1}|s_k)\, q(a_k|s_k \to s_{k+1})}{r(a_k, s_{k+1}|s_k)} \tag{23}$$

as follows from the definition of HMM. The same idea applies even if null transitions are present, only the closed formula is not so simple as indexes of states and outputs fall out of sync. It is then easier to formulate it on a trellis, pruned according to an output $a_0, \ldots, a_{N-1}$. Let $e_0, \ldots, e_{K-1}$ be a path thru that trellis starting in x_s^0 and ending in x_k^N for some $k \in S$. Then its probability is

$$P(\vec{a}\ \&\ \vec{e} \mid s_0) = \prod_{k=0}^{K-1} r(A(e_k), R(e_k) \mid L(e_k)) \tag{24}$$

Among all those paths we want to find the one with the highest probability. Taking minus logarithm of quantity (24) we obtain $\sum_{k=1}^{K} \delta(e_k)$, where $\delta(e_k) := -\log(r(A(e_k), R(e_k) \mid L(e_k)))$ can be regarded as a 'length' of the arrow. The most probable path will be the shortest one. We can use Dijkstra's algorithm (invented 1959) to find it because $\delta(e) \geq 0$ for any arrow e from the trellis.

I will present its special case here, called a *Viterbi algorithm*, which exploits special topology of the trellis graph to avoid the heap of visited vertices. As a trade-off, it requires null transitions to be acyclic, whereas the general Dijkstra has no such restriction. But this restriction is almost always fulfilled because null arrows are mainly used to connect phonemes into words and words into a language model. Moreover, any HMM with null transitions can be transformed into an equivalent HMM without them.

2.5.1 The Viterbi Algorithm

Let us have a pruned trellis with arrows labeled by $\delta(e)$. Let us define an array of real valued variables γ so that for each trellis node x_k^t we have γ_k^t, holding a length of the shortest path found so far from x_s^0 to x_k^t. Algorithm proceeds as follows:

(0) Initialize all elements of γ to $+\infty$ except $\gamma_s^0 := 0$; set t:=0.

(1) Walk thru states $n \in S$ in t-th trellis column, starting at its top. Note that the states in trellis columns are sorted such that null arrows can only point downwards. For each state $n \in S$ and each trellis arrow e such that $L(e) = n$ and $R(e) = m$, perform the following operation

$$\gamma_m^{t+1} := \min(\gamma_m^{t+1}, \gamma_n^t + \delta(e)) \tag{25}$$

(2) If $t < N$, then $t := t + 1$ and go to (1).

(3) Else the last column contains lengths of shortest paths ending in the respective HMM states. By finding a state having this value minimal and by backtracking towards the origin of the trellis, we can recover the sequence of states which most probably caused the observed output.

Because the correctness of Dijkstra's algorithm is treated extensively elsewhere [33] and the Viterbi algorithm is just a special case of it[12], I will omit a proof here.

2.5.2 Simple Recognizer

Consider word models connected by null-arrows as shown in fig. 6. The probability of going from word w_i to word w_j is set in accordance with bigram language model $P(w_j|w_i)$. All other probabilities in this composed HMM are determined by acoustic training which will be described in section 2.6. According to (1), the most probable sequence of words can be determined as follows:

▷ 27 ▷ 29 ▷ 14

$$\vec{\hat{w}} := \arg\max_{\vec{w}} \sum_{\vec{s}\in\mathcal{S}(\vec{w})} P(\vec{a}, \vec{s} \mid \vec{w}) P(\vec{w}) \tag{27}$$

[12] The order in which the Viterbi algorithm processes nodes of the trellis can be exactly achieved by Dijkstra's algorithm if we redefine lengths of non-null arrows as $\delta_1(e) := \delta(e) + NK$, where $K := N \sum_e \delta(e)$, keeping $\delta_1(e) := \delta(e)$ for null arrows. As all paths ending in a given column of the trellis use the same number of non-null arrows, it does not change the result (up to the known constant, which can be subtracted afterwards), while at the same time we have enforced that the next column will be processed only after the current one has been finished because K is greater than any possible path thru the trellis. To enforce the right order within each column we can use the very same trick, defining

$$\delta_2(x_a^t \to x_b^p) := \delta_1(x_a^t \to x_b^p) + K(\nu(a) - \nu(b)) \tag{26}$$

where $\nu : S \to \mathbb{N}$ maps each state to the number of its row in the trellis (for the topmost state s_t we have $\nu(s_t) = 0$).

Besides the order, the update equation (25) is the same in both algorithms. As we have shown that the Viterbi order is achievable by Dijkstra's algorithm and that order does not change the result (up to the known constants) we can drop Dijkstra's heap machinery because actual values of $\delta(e)$ have no influence on the processing order. Hence, the correctness of Dijkstra's algorithm implies the correctness of the Viterbi algorithm.

where $\mathcal{S}(\vec{w})$ is a set of all state sequences, each of which going inside HMM models of the word sequence $\vec{w}$. In trellis, (27) translates into[13]

$$\vec{\hat{w}} := \arg\max_{\vec{w}} \sum_{\substack{\vec{e}\in\mathcal{T}(\vec{a}) \text{ and} \\ \langle L(e_0),\ldots,L(e_{\#\vec{e}-1}),R(e_{\#\vec{e}-1})\rangle\in\mathcal{S}(\vec{w})}} \prod_{k=0}^{\#\vec{e}-1} r(A(e_k),R(e_k) \mid L(e_k)) \qquad (28)$$

where $\mathcal{T}(\vec{a})$ is a set of all arrow paths in a trellis pruned by an acoustic evidence vector $\vec{a}$. Note that the language model term $P(\vec{w}) = P(w_0)\prod_i P(w_i|w_{i-1})$ is hidden in the null arrows connecting word HMMs, and $P(w_0)$ are those null arrows going from the initial state to every word in fig. 6.

Unfortunately, formula (28) is too complicated to be practically usable. For this reason we will approximate it by changing the sum into a maximum operator. Since the sequence $\vec{e}$ uniquely determines the sequence $\vec{w}$, we can write the modified (28) as

$$\vec{\hat{e}} := \arg\max_{\vec{e}\in\mathcal{T}(\vec{a})} \prod_{k=0}^{\#\vec{e}-1} r(A(e_k),R(e_k) \mid L(e_k)) \qquad (29)$$

translating the best transition sequence $\vec{\hat{e}}$ into a word sequence $\vec{\hat{w}}$ afterwards. Noticing that (29) is exactly what the Viterbi search does, we can say that we already have basic off-line[14] recognizer based on bigram language model.

When we have decided to change sum into maximum while going from (28) to (29), we have tactically assumed that alternative state sequences of any given word sequence $\vec{\hat{w}}$ would contribute to the total probability in a uniform way so that the best word sequence in formula (28) would likely be the best one in case of formula (29), too.

This would hold exactly if we had only one path thru each word and the instants of word beginnings and word endings would be correctly given to us in advance. This is certainly not the case but we can still hope that alternative paths with non-negligible probabilities will have their total probability mass somewhat proportional to the probability of the best path. In that case both methods would be likely to find the same winning word sequence, especially if the acoustic front would provide features allowing for high confidence discrimination between the phonemes.

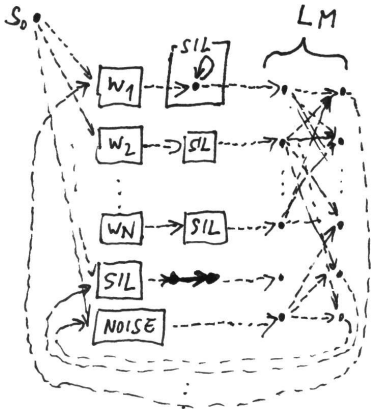


Fig. 6. *Bigram HMM constructed from word models $W_1 \ldots W_N$, model of optional silence SIL and model for non-speech sounds NOISE. All these models are connected by null arrows which form complete bipartite graph denoted as LM with probabilities set according to the language model. SIL is also used to model long silence in which case it is followed by single output producing arrow to keep null-arrows acyclic. Note that HMM states in the LM part are superfluous and are drawn only to increase readability.*

[13] $\#\vec{s}$ denotes the number of elements of vector $\vec{s}$.

[14] Whole sound $\vec{a}$ must be known before the recognition could begin.

The Viterbi algorithm cannot be used with trigram language model directly. Having a two-word history represented by an HMM state would force us to make copies of all word models for each word. This would roughly square the number of HMM states, making the model practically useless.

Nevertheless, the trigram model and even more sophisticated models can be approximated to some extent. When the Viterbi algorithm tries to extend the path from a trellis node x_k^t using a formula (25) the distance γ_k^t has already achieved its final value and, as such, it contains the length of the shortest path from x_s^0 to x_k^t. This path will not be changed by future computations and can be traced back to obtain the list of its words. Now, all we have to do is to mark interword arrows as *dynamic* ones and modify the Viterbi search in such a way that each time it encounters a dynamic arrow, it extracts words of its history and uses them, in whatever language model we like, to compute the transition probability of that arrow. Besides that, it proceeds the same way as the original Viterbi search.

Unfortunately, it may happen that the correct word sequence remains undiscovered by this algorithm. Let us see why this is so. What we would like to achieve by incorporation of the general language model into the Viterbi search (29), is the following optimization[15]:

$$\vec{\hat{w}} := \arg\max_{\vec{w}} \max_{\vec{s}\in\mathcal{S}(\vec{w})} P(\vec{a},\vec{s} \mid \vec{w})P(\vec{w}) \qquad (30)$$

where $P(\vec{a},\vec{s} \mid \vec{w})$ is a probability of walking thru the trellis along the path $\vec{s}$ while generating output sequence $\vec{a}$, where only the contributions from intra-word arrows are counted. The probability mass of inter-word arrows is subsequently managed globally by a language model $P(\vec{w})$. In the case of the bigram language model, the Viterbi algorithm applied on HMM graph from fig. 6 finds as probable solution as (30) with bigram model $P(\vec{w})$ would return.

By use of dynamic arrows, we can account for arbitrarily long histories in $P(\vec{w})$ but once we leave trellis node x_k^t, the winning state path from x_s^0 to x_k^t becomes fixed for that node and so becomes its associated word sequence. Hence, if there are two different word sequences passing thru this node, such that the first one seems more promising by the time we process the node x_k^t but the second one is a prefix of the actual optimal path, the Viterbi algorithm with dynamic arrows will not discover it because there is no way to revert its decision after the node x_k^t has been left. Note that this situation can really arise. For example, consider two utterances *'I scream is a sentence'* and *'Ice cream is cold'*. Assume that both *'I scream'* and *'Ice cream'* are roughly equally probable, given the acoustic data. But the language model favors the trigram *'Ice cream is'*. If the input utterance was the first one, recognized words might be *'Ice cream is a sentence'* or the last words might also be misrecognized, if the language model assigned too low probability to *'Ice cream is a sentence'*. In either case, the output would start by incorrect *'Ice cream is'*.

2.5.3 Computational Demands

For real world recognition tasks, even the approximation (29) is too demanding, should it be computed directly by the Viterbi algorithm. For a vocabulary of 100k words, a word being 4 phonemes long on average, we would need roughly 2.9M HMM states for

[15] Which can be also obtained from (27) by turning the sum into the maximum.

21 ◁ the model from fig. 6 where phonemes are modeled as 7 state HMMs of fig. 3. When the trellis is pruned according to an observation $\vec{a}$, each state fans-out about two arrows on average, except word-connecting states, each of which is fanning-out 100k null-arrows. As there are averagely $7 \cdot 4 = 28$ inner states per every connecting state, we have about 3450 outgoing arrows per state, on average. For a standard front end with 100 Hz frame
26 ◁ rate, this would amount to $100 \cdot 2.9 \cdot 10^6 \cdot 3450 = 10^{12}$ evaluations of (25) per second. Even if this would take only 10 CPU clocks, which is a very optimistic estimate, we would need 10 000 GHz CPU to keep up with the real time. It is clear that we have to do something radical to cut the computational cost down.

2.5.4 The Beam Search

This radical modification is known as the *beam search* and it works by abandoning poor paths before investing too much effort into them. Specifically, during processing of operation (25) we keep track of minimal γ_m^{t+1} among those nodes we have modified. Let us call it μ_{t+1}. Then, in the next step $t_1 = t + 1$, we perform (25) only at those trellis nodes $x_m^{t_1}$ for which $\gamma_m^{t_1} < K\mu_{t_1}$, where K is a suitable constant. The effect of excluding unpromising trellis nodes from computation (25) is the same as setting their γ back to $+\infty$.

Of course, if we do this too aggressively we may throw away the path we are looking for. The aggressiveness can be controlled by constant K, which has to be tuned for each specific application. $K = 5$ may be a good starting guess.

Note that in each step we have to walk thru all active states when selecting the minimum and then again in the next step, when locating states below the threshold. The speedup comes from two effects. First, by omitting certain nodes we don't need to examine their arrows, which can already save a lot of work as there can be many arrows emanating from certain nodes. Secondly, by not examining these arrows we don't even reach unpromising nodes they point to, which effectively stops the avalanche of uninteresting nodes, keeping the number of active nodes low.

2.6 Training

The language model can be trained independently from the acoustic model as was
17 ◁ already described in section 2.3. This section explains how to train the acoustic model from the sound recording and its phonetic transcription.

Standardly, the acoustics is trained using *Maximum Likelihood Estimation (MLE)*. That is, having a time-aligned training data $\vec{w}$ and $\vec{a}$ and fixed HMM structure (such as the one in fig. 6), we want to set the parameters of the model[16] such that the probability of the training data will be maximal. Formally, if we store all our model's parameters in vector $\vec{\lambda}$, defining $P_{\vec{\lambda}}(\vec{w}\ \&\ \vec{a})$ to be the probability the model assigns to the occurrence of specific word string $\vec{w}$ and its acoustic representation $\vec{a}$, we want to find $\vec{\lambda}$ which would maximize this probability on our training data:

$$\vec{\lambda} := \arg\max_{\vec{\lambda}} P_{\vec{\lambda}}(\vec{w}\ \&\ \vec{a}) = \arg\max_{\vec{\lambda}} P_{\vec{\lambda}}(\vec{a} \mid \vec{w}) P(\vec{w}) = \arg\max_{\vec{\lambda}} \prod_k P_{\vec{\lambda}}(\vec{a}_k \mid \vec{w}_k) \qquad (31)$$

[16] That is transition probabilities $r(a, x \mid y)$ in our case.

The term $P(\vec{w})$ can be missing in the last formula because we consider the language model as already trained, hence independent of parameters $\vec{\lambda}$. Known time alignment is denoted by subvectors $\vec{a}_k$ and $\vec{w}_k$ which make up original vectors $\vec{a}$ and $\vec{w}$, respectively. For instance, we might have the training data chunked (and therefore aligned) by sentences or even by words. Although the training process can, in principle, find the correspondence between words and acoustics it is better to guide it by this kind of alignment. It makes training faster and less ambiguous, leading to slightly better performance of the recognizer.

$P_{\vec{\lambda}}(\vec{a}_k \mid \vec{w}_k)$ is a probability that the HMM in which we have visited the words as specified in $\vec{w}_k$ generates the output $\vec{a}_k$. Following the training text $\vec{w}_k$, we can unroll the HMM model from fig. 6 into the model of fig. 7. Then, $P_{\vec{\lambda}}(\vec{a}_k \mid \vec{w}_k)$ can be computed simply as a probability that this new unrolled HMM generates the acoustics $\vec{a}_k$. Note that the parameters $\vec{\lambda}$ are shared among many arrows in the model. At least every occurrence of a given phoneme uses the same set of parameters and there may be even other shared parameters of some arrows, induced by explicit parameter tying. This situation is particularly welcome in training because it makes estimation of probabilities possible. For each parameter that has to be estimated, we (hopefully) have several of its arrows in the unrolled HMM.

In principle, $\vec{\lambda}$ can be found by numerical search, such as *conjugate gradients* or *Monte-Carlo* method. Moreover, there is much faster way for finding local maximum, called *Baum-Welsh algorithm*. It works iteratively, starting with random parameters, increasing probability of the data in each iteration. Usually 3 or 4 iterations are sufficient. It is advisable to repeat the whole procedure from different starting points to prevent us from stranding in poor local maximum. Alternatively, the Baum-Welsh could be used as a local optimization subroutine in more involved Monte-Carlo method.

Baum-Welsh algorithm is a special case of so called *expectation maximization (EM) algorithm* which can be proved not to decrease already achieved probability of the data.

For each and every arrow in the trellis of HMM from fig. 7, we compute how many times on average it has been taken when $\vec{a}$ was generated. This is averaged over all possible paths thru that trellis (starting in s_0, ending in s_1), according to probabilities dictated by initial parameters $\vec{\lambda}$ and over all occurrences of each arrow in the trellis (which happens due to tying). Then, new arrow probabilities $\vec{\omega}$ are made to be proportional to these averages.

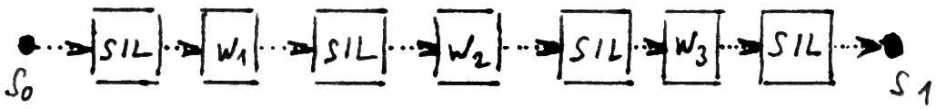


Fig. 7. *Hidden Markov Model for training text $w_1w_2w_3$ composed of the respective word models strung between boundary states s_0 and s_1. SIL model captures optional silence between the words.*

2.6.1 Baum-Welsh Algorithm

Let $E \subseteq S^2$ be the set of all arrows present in the HMM model from fig. 7, except those that are used to connect words or phonemes — these have the probability of 1 since there is only one way across the words in the training HMM. Functions A, L, R, and P have the meaning defined in note 2.4.2. For each arrow $e \in E$, its *tuft* will be defined ▷ 22
as a set of all arrows emanating from $L(e)$, that is $Y(e) := \{x \in E \mid L(x) = L(e)\}$. This

notation will be useful upon summing over all arrows in the tuft (which will be done to ensure that the total probability sums to 1).

To tie parameters, we define the transition probabilities as $P(e) := \lambda_{M(e)}$. The mapping $M : E \to \mathbb{N}$ from arrows to indexes specifies which arrows share their parameters by putting all arrows with the same index into single group. To be consistent, M must satisfy the following property:

$$\begin{aligned} &\forall a,b \in E: \quad M(a) = M(b) \Longrightarrow \\ &\quad \exists \varphi : Y(a) \to Y(b) \text{ s.t. } \varphi \text{ is bijective and } \forall x \in Y(a) : M(x) = M(\varphi(x)) \end{aligned} \tag{32}$$

which means that once two arrows a and b are tied, then their tufts must be tied in the same way. This ensures that $\sum_{e\in E:L(e)=L(x)} P(e) = \sum_{e\in Y(x)} P(e) = \sum_{e\in Y(x)} \lambda_M(e) =$ 1 for all[17] $x \in M^{-1}[\{k\}]$ once it holds for one such $x \in M^{-1}[\{k\}]$. Note that it is allowed to tie arrows e_1 and e_2 for which $L(e_1) = L(e_2)$ and $R(e_1) = R(e_2)$. This may be used to tie acoustically similar arrows to artificially reduce effective number of symbols in the acoustic alphabet for some phonemes.

After the algorithm gathers pseudo-counts $\widehat{c}_k$ in a way that will be explained later on this page, new parameters $\vec{\omega}$ will be finally assigned as

$$\omega_k := \frac{\widehat{c}_k}{\sum_{e\in Y(m(k))} \widehat{c}_{M(e)}} \tag{33}$$

where $m : \mathrm{Rng}(M) \to E$ is arbitrary function such that $M(m(k)) = k$ for $\forall k \in \mathrm{Rng}(M)$.

Each pseudo-count c_k has to contain how many times, on average, we traversed an arrow belonging into k-th group if we used HMM to generate string $\vec{a} = \langle a_0 \dots a_{N-1}\rangle$. Formally:

$$\widehat{c}_k := \sum_{e\in M^{-1}[\{k\}]} \sum_{t=0}^{N-1} P(e \textit{ was taken to exit trellis node } x^t_{L(e)} \mid \vec{a}, \vec{\lambda}) \tag{34}$$

It will be easier to compute $P(e \textit{ was taken to exit trellis node } x^t_{L(e)}, \vec{a} \mid \vec{\lambda})$. It differs only in the multiplicative factor $P(\vec{a} \mid \vec{\lambda})$ which is constant and cancels in (33) anyway. Note that both null and non-null arrows are considered. These two cases have to be worked out separately. Let us begin with a non-null arrow $e = x \xrightarrow{a_t} y$. The probability $P(e, \vec{a}, t \mid \lambda)$ is

$$\sum_{\vec{e}\in\mathcal{T}_{s_0}(\langle a_0\dots a_{t-1}\rangle)} \left(\prod_i P(e_i)\right) \eta(x \mid R(e_{\#\vec{e}-1}))\, r(a_t, y \mid x) \sum_{\vec{f}\in\mathcal{T}^{s_1}_y(\langle a_{t+1}\dots a_{N-1}\rangle)} \prod_j P(f_j) \tag{35}$$

where the first sum is considered 1 when $t = 0$ and the last sum is considered 1 for $t = N-1$ iff $y = s_1$. Otherwise it is treated as 0 for $t = N-1$. Rewriting the first sum

40 ◁ [17] $M^{-1}[A]$ is a *preimage of set A*, defined in 3.1.5

according to (11) we get: ▷ 23

$$\sum_z \sum_{\vec{e}\in\mathcal{T}^z_{s_0}(\langle a_0\dots a_{t-1}\rangle)} \left(\prod_i P(e_i)\right) \eta(x \mid z)\, r(a_t, y \mid x) \underbrace{\sum_{\vec{f}\in\mathcal{T}^{s_1}_y(\langle a_{t+1}\dots a_{N-1}\rangle)} \prod_j P(f_j)}_{\beta_{t+1}(y)} \tag{36}$$

Recalling (12) and (18), we have ▷ 23 ▷ 24

$$P(e, \vec{a}, t \mid \lambda) = \sum_z \pi_t(z)\, \eta(x \mid z)\, r(a_t, y \mid x)\beta_{t+1}(y) = \alpha_t(x)\, r(a_t, y \mid x)\beta_{t+1}(y) \tag{37}$$

For a null arrow $x \to y$ we can write $P(e, \vec{a}, t \mid \lambda)$ as follows:

$$\sum_w \sum_{\vec{e}\in\mathcal{T}^w_{s_0}(\langle a_0\dots a_{t-1}\rangle)} \left(\prod_i P(e_i)\right) \eta(x \mid w)\, r(\varepsilon, y \mid x) \sum_{\vec{f}\in\mathcal{T}^{s_1}_y(\langle a_t\dots a_{N-1}\rangle)} \prod_j P(f_j) \tag{38}$$

where empty sums are treated as before. This boils down to:

$$P(e, \vec{a}, t \mid \lambda) = \alpha_t(x)\, r(\varepsilon, y \mid x)\beta_t(y) \tag{39}$$

All we need now is a recurrence for β. We have

$$\beta_t(y) = \sum_{\vec{f}\in\mathcal{T}^{s_1}_y(\langle a_t\dots a_{N-1}\rangle)} \prod_j P(f_j) = \sum_{z,w} \eta(z|y)\, r(a_t, w|z) \sum_{\vec{f}\in\mathcal{T}^{s_1}_w(\langle a_{t+1}\dots a_{N-1}\rangle)} \prod_j P(f_j) \tag{40}$$

Empty sum gives $\beta_N(s_1) = 1$ and $\beta_N(x) = 0$ for all other x. From (40) we have

$$\beta_t(y) = \sum_z \eta(z|y) \sum_w r(a_t, w|z)\, \beta_{t+1}(w) \tag{41}$$

In an acyclic case, we can expand $\eta(z \mid y)$ according to (19): ▷ 24

$$\begin{aligned} \beta_t(y) &= \sum_w r(a_t, w|y)\beta_{t+1}(w) + \sum_{z>y} \sum_{y<k\le z} \eta(z|k)\, r(\varepsilon, k|y) \sum_w r(a_t, w|z)\beta_{t+1}(w) \\ &= \sum_w r(a_t, w|y)\beta_{t+1}(w) + \sum_{k>y} r(\varepsilon, k|y) \underbrace{\sum_{z>y} \eta(z|k) \sum_w r(a_t, w|z)\beta_{t+1}}_{\beta_t(k)} \end{aligned} \tag{42}$$

Now we have all that is needed for the algorithm. Let us state it here.

(1) **Forward Pass:** Compute $\alpha_t(k)$ for each trellis node x^t_k, $k \in S$, $t \in [0, N-1]$, starting with $\alpha_0(s_0) := 1$ and $\alpha_0(z) := 0$ for all $z < s_0$.

$$\begin{aligned} \alpha_0(m) &:= \sum_{k<m} r(\varepsilon, m \mid k)\, \alpha_0(k) \\ \alpha_t(m) &:= \sum_k r(a_{t-1}, m \mid k)\, \alpha_{t-1}(k) \;+ \sum_{k<m} r(\varepsilon, m \mid k)\, \alpha_t(k) \end{aligned} \tag{43}$$

(2) **Backward Pass:** Compute $\beta_t(k)$ for each node x_k^t, $k \in S$, $t \in [0, N]$, starting with $\beta_N(s_1) := 1$ and $\beta_N(m) := 0$ for all other $m \in S$.

$$\beta_t(y) := \sum_w r(a_t, w|y)\beta_{t+1}(w) + \sum_{k>y} r(\varepsilon, k|y)\beta_t(k) \tag{44}$$

(3) **Pseudocount Formation:**

$$c_k := \sum_{e \in M^{-1}[\{k\}]} \sum_{t=0}^{N-1} \alpha_t(L(e))\ P(e)\ \beta_{Z(e,t)}(R(e)) \tag{45}$$

where Z(e,t)=t for null arrow e, and Z(e,t)=t+1 otherwise. For multiple time aligned utterances $(\vec{a}_1, \vec{w}_1, \vec{a}_2, \vec{w}_2, \ldots)$ we have to construct the trellis and perform steps (1) and (2) for each of them, then computing pseudocounts such that (45) would add contribution from all respective trellises, perhaps weighting them somehow to account for their varying acoustic relevance.

(4) **Parameter Update:** Compute new parameters from the pseudocounts as follows:

$$\omega_k := \frac{c_k}{\sum_{e \in Y(m(k))} c_{M(e)}} \tag{46}$$

Finally, assign $\vec{\lambda} := \vec{\omega}$ and go to (1) until convergence.

2.6.2 Note During computation, the probabilities involved in α and β usually become so small that they cannot be represented by standard floating point numbers. Fortunately, α_t and β_t are linear in α_{t-1} and β_{t+1}, respectively, hence we can normalize whole column α_t such that it could be represented with sufficient precision. This can be done by obtaining the binary exponent w_t of $\sum_s \alpha_t(s)$ using standard C function `frexp()` followed by multiplication of $\alpha_t(s)$ with 2^{-w_t} with the `ldexp()` function for each $s \in S$. Let us call this new normalized value $\alpha_t^\star$. As this is performed in every iteration of (43) and we have $\alpha_t^\star = \alpha_t \prod_{i=0}^{t} 2^{-w_i}$ as a result. Defining $\beta_t^\star = \beta_t \prod_{i=t}^{N} 2^{-w_i}$, we have $\alpha_t^\star \beta_{t+1}^\star = \alpha_t \beta_{t+1} \prod_{i=0}^{N} 2^{-w_i}$ where the very small quantity $\prod_{i=0}^{N} 2^{-w_i}$ cancels in (46) so we can omit it altogether in (45), obtaining:

$$c_k := \sum_{e \in M^{-1}[\{k\}]} \sum_{t=0}^{N-1} \alpha_t^\star(L(e))\ P(e)\ B(t, e) \tag{47}$$

where $B(t, e)$ is $2^{w_t}\beta_t^\star(R(e))$ iff e was a null arrow and $\beta_{t+1}^\star(R(e))$ otherwise.

2.7 Word Error Rate (WER)

Now as we have trained our recognizer we want to know how good it really is. We may play with it to get a general feeling but we will soon find out we need something that could be automated, that would condense the performance into a single number. This would allow us to quickly compare different recognizers (or different versions of single recognizer), provided that we use the same training and testing data. This number is called *word error rate*, or shortly *WER*, and it is based on so called Levenshtein distance.

2.7.1 Definition *Levenshtein Distance*

For two strings A and B, their *Levenshtein distance* is defined as minimal number of single-character deletions, insertions and substitutions, which transform A into B.

Proof Let us proof that it is really a metric. It is obviously non-negative and symmetric (since we can read insertion backwards as deletions and vice versa) and once we know the distance (thus the editing steps) from A to B and from B to C, we immediately obtain an upper bound for a distance from A to C just by concatenating the editing commands. Hence the triangle inequality also holds. Q.E.D.

2.7.2 Definition *Word Error Rate*

Given the recognizer's output and true (human) transcript of the audio recording, we define WER using the Levenshtein distance between output and true transcript applied on words the following way:

$$\mathrm{WER}_\alpha := \frac{S + \alpha I + \alpha D}{N} \tag{48}$$

where N is the length of the transcribed text, I is the number of inserted words, D the number of deleted words and S is the number of substituted words. The purpose of the parameter α is to balance costs of different kinds of errors[18]. Normally $\alpha = 1$, but $\alpha = 1/2$ is also used.

The WER metric is quite coarse, completely ignoring severity of errors. This can be partially alleviated. For instance, we might want a misrecognition of certain words to be more costly. By computing S such that we will count $\sigma(A_i, B_i)$ for each error instead of 1, where σ is a symmetric ($\sigma(a, b) = \sigma(b, a)$) cost, we obtain the modified Levenshtein distance (owing to the symmetry it is still a metric), which does the job. This can be useful when we need WER to be more benevolent upon errors such as a recognition of *'cannot'* for *'can't'* and the like. It is especially helpful in case of Czech language, where most words have several pronunciation variants[19].

Non-symmetric σ would be also meaningful — imagine that the computer erroneously recognizes the command *'edit the file'* as *'delete that file'*. Intuitively we feel that this is worse error than if it was the other way round. But then, WER could not be symmetric anymore and consequently $N \cdot \mathrm{WER}$ would not be a metric.

These extensions to WER are rarely used due to their arbitrariness and application specificity. For these reasons, WER_1 remains standard when comparing performance of different systems.

2.8 Limitations of the Noisy Channel Setup

In this section, I present a list of various limitations I encountered. Some of them are fundamental, stemming from the noisy channel model guided by formula (1), others are rather implementation-related — for example, by imposing HMM structure on $P(A|W)$ ▷ 14

[18] It cannot be generalized by introducing different coefficient β for deletions without losing symmetry.

[19] For instance, the word *'cautious'* can be uttered as *'opatrný'*, *'vopatrný'*, *'vopatrnej'* or *'opatrnej'*, where only the first form is considered proper for writing but very few people actually speak that way, unless when reading what was previously written.

we lose the ability to adequately model slowly changing long range dependencies in A such as those present in prosody. However, it was our choice to use the HMM, the formula (1) does not prescribe that, therefore this limitation is not fundamental. ◁14

2.8.1 Too Coarse Notion of the Recognition Error

It can be proved that (1) minimizes the probability of an error, if the distributions $\Pr(W)$ and $\Pr(A|W)$ are exactly known. However, an error is a situation when the decoder returned $\widehat{W}$ different from the true transcript W. It does not distinguish severity of an error, not even the number of misrecognized words.

Therefore, we may feel that more detailed formula should be used which would allow us to minimize WER directly, optionally allowing us to specify which words, or even which word phrases, are the most costly to be interchanged (something like σ that we have considered in section 2.7 for fine-grained WER).

This approach was tried in work [62]. Their goal function was so called *Minimum Bayes Risk Classifier*:

$$\delta(A) := \widehat{W} = \arg\min_{W'} \sum_{W} \sigma(W, W')\, P(W|A) \tag{49}$$

which can be proved to minimize the expectation of the error cost function σ:

$$\mathcal{E}_{P(A,W)}\left(\sigma(W, \delta(A))\right) \tag{50}$$

provided that the distribution $P(A, W)$ (which is invoked in $\mathcal{E}$) was known exactly. Note that this is a generalization of the original objective (1) as it can be simulated by (49) with[20] $\sigma(W, W') := (W{=}W')\,?\,0 : 1$.

However, (49) is quite hard to implement and it was found in [62] that it brings only 1.6 % relative improvement on WER, when used with $\sigma = \mathrm{WER}_1$. So it seems that it does not pay-off after all. The rationale for this behavior is that shortcuts used in evaluation of (49) might have outweighed its benefits. Another explanation might be that ordinary recognizers in fact do not use (1) precisely the way it is stated. They do not run Viterbi algorithm over the entire audio file. Instead, it is run only locally on a sentence or two, hence it can be thought of to minimize the probability of an error in that sentence which is somewhat close to what (49) does with $\sigma = \mathrm{WER}_1$, especially when the error rate is low.

2.8.2 Rigidity of the Language Model

Another fundamental limitation is that we assume $P(W)$ to be constant. But this is not quite true in the real world. First of all, if we model it using N-grams, we can hardly capture that the distribution is globally different for different people and different topics so it will appear to us as changing on each topic change. By using constant $P(W)$ whose N-gram model was averaged across all topics and speakers we would lose performance. But this is merely an implementation issue and methods exist to cope with it, namely the *Latent Semantics Analysis* [34] which combines trigrams with overall distributions trying to capture topic related words and phrases.

[20] Notation $a\,?\,b : c$ is borrowed from the C language and means b iff a was true; else it means c.

The fundamental issue is that the language evolves. New words are being invented, new phrases come in and out of fashion, etc. Hence $P(W)$ really depends on time. It therefore seems that the ultimate speech recognition engine should be constantly learning up-to-date distribution $P(W)$ from what it hears, departing from the original formula (1) which assumed constant $P(W)$.

A small step in that direction is so called *Cache Language Model*. It accumulates trigram counts from the text dictated so far into so called *cache* and interpolates between the cache and big (general purpose) language model. It is based on observation that an improbable word, once said, considerably raises its own probability to be said again within certain time-window. This may be justified by imagining this word to be a technical term, say a *bearing*. If we talk about something related to engines we might use this word several times within a while, although it is quite improbable on average. Nevertheless, this technique has to be implemented very carefully so that misrecognized words would not enter the cache at all. Otherwise they could trigger an avalanche of recognition errors later, as they spoil the combined language model.

2.8.3 Ignorance of Noise Fluctuations

Typical non-fundamental limitation of traditional ASR systems is that they assume the noise in the channel to be constant, for which reason sudden non-speech sounds usually confuse the recognizer which in turn makes an error which is then propagated due to the language model, possibly leading to yet other errors. However, this can be overcome within the noisy channel framework as will be explained in subsection 3.3.15. ▷69 It is just not commonly used today.

2.8.4 Practical Suboptimality of the Decoder

Although (1) can be proved to minimize the probability of returning word transcript containing an error, this holds only if we knew true distributions $\Pr(A|W)$ and $\Pr(W)$ exactly. As this is almost never the case,[21] (1) is likely to be suboptimal classifier. There are two practical workarounds addressing this problem (the two can even be combined together). The first method works in recognition time, using slightly modified version of classifier (1), while the second one, so called *Maximum Mutual Information Estimation (MMIE)*, is used during training to estimate $P(A|W)$ from the training data in a way which typically improves performance of the recognizer over the MLE training method of section 2.6. The two methods will be briefly described in the following two ▷29 subsections.

2.8.5 Fudge Factor

It was found that the recognizer actually works better if it performs decoding governed by the following formula

$$\widehat{W} := \arg\max_{W} P(A|W)^{\alpha} P(W)^{1-\alpha} \tag{51}$$

[21] There are two reasons for this: First, from finite training data we cannot derive possibly infinite information present in the distribution and secondly, by imposing model structure on the distribution, such as trigrams or HMMs, we deliberately reject details present in the training data that are incompatible with assumptions of our model.

14 ◁ instead of formula (1), where the *fudge factor* α is determined by a grid search, which involves training and testing whole recognizer for many values of $\alpha \in (0,1)$ with reasonably small step. The parameter α was about 1/17 during eighties in the IBM group[22]. This can be interpreted such that the decoder prefers decisions from the language model over the information the machine actually hears.

2.8.6 MMIE Training

Traditional MLE training tries to maximize $P_{\vec{\lambda}}(A|W)$ on the training data by proper selection of $\vec{\lambda}$. However, in recognition time we are selecting W with maximal $P_{\vec{\lambda}}(W|A)$ as the winning hypothesis. This brings an idea that it might be worthwhile to estimate $P_{\vec{\lambda}}(A|W)$ by maximizing $P_{\vec{\lambda}}(W|A)$ on the training data, instead[23]. Formally:

$$\vec{\lambda} := \arg\max_{\vec{\lambda}} P_{\vec{\lambda}}(W|A) = \arg\max_{\vec{\lambda}} \prod_k P_{\vec{\lambda}}(\vec{w}_k \mid \vec{a}_k) = \arg\max_{\vec{\lambda}} \prod_k \frac{P_{\vec{\lambda}}(\vec{w}_k \,\&\, \vec{a}_k)}{P_{\vec{\lambda}}(\vec{a}_k)} \tag{52}$$

where $\vec{a}_k$ and $\vec{w}_k$ are segmented training utterances. As the language model is trained separately and does not depend on $\vec{\lambda}$ we can also write:

$$\vec{\lambda} := \arg\max_{\vec{\lambda}} \sum_k \log_2 \frac{P_{\vec{\lambda}}(\vec{w}_k \,\&\, \vec{a}_k)}{P_{\vec{\lambda}}(\vec{a}_k) P(\vec{w}_k)} \tag{53}$$

54 ◁ Formally, this is a cross mutual information (see 3.2.19), hence the name of the method. Rewriting it once more we get the following expression.

$$\vec{\lambda} := \arg\max_{\vec{\lambda}} \prod_k \frac{P_{\vec{\lambda}}(\vec{a}_k|\vec{w}_k) P(\vec{w}_k)}{\sum_v P_{\vec{\lambda}}(\vec{a}_k|\vec{v}) P(\vec{v})} \tag{54}$$

As we try to maximize it, we are in fact making the nominator high and denominator low, on average over all training utterances. Since the nominator and each term of the denominator is a quantity that guides the decoder (1), we can interpret (54) as a discriminative training rule which tries to find $\vec{\lambda}$ to make the training text $\vec{w}_k$ easily recognizable by (1) from the training sound $\vec{a}_k$, while penalizing other texts $\vec{v}$ as transcriptions or $\vec{a}_k$. This should lower the probability of error if we assume that the testing data will behave similarly in this respect. Note that $\vec{v}$ goes over all word-strings, including $\vec{w}_k$. This makes the fraction ≤ 1. Perhaps the formula

$$\vec{\lambda} := \arg\max_{\vec{\lambda}} \prod_k \frac{P_{\vec{\lambda}}(\vec{a}_k|\vec{w}_k) P(\vec{w}_k)}{\sum_{v \neq w_k} P_{\vec{\lambda}}(\vec{a}_k|\vec{v}) P(\vec{v}) + \beta P_{\vec{\lambda}}(\vec{a}_k|\vec{w}_k) P(\vec{w}_k)} \tag{55}$$

could work even better as the parameter $\beta > 0$ could be tuned on the heldout data. Also notice presence of the language model (which is supposed to be trained in advance) in

[22] Personal communication with Fred Jelínek.

[23] Note that the two cases are different. On recognition we maximize over W with $\vec{\lambda}$ fixed, while during training it is the other way round. Optimal $\vec{\lambda}_0$ then does not guarantee that $\forall T : P_{\vec{\lambda}_0}(W|A) \geq P_{\vec{\lambda}_0}(T|A)$ even on the training data. Nevertheless, it makes the inequality more 'probable' since λ_0 can be understood as a result of discriminative training.

(54). It allows acoustics to be trained so that its free parameters would not be wasted on sounds for which the context makes the next word almost determined.

This can be appreciated from yet another point of view. MLE training tries to fit $P(\vec{a}_k|\vec{w}_k)$ but what happens with $P(\vec{a}_k|\vec{v})$ for $v \neq w_k$ is far less clear (due to the limited amount of training data). MMIE, on the other hand, does not insist on 'precise' shape of $P(A|W)$ distribution and cares about classification boundaries instead. It is believed that these boundaries are more immune to statistical mismatch between the training and testing data. This can be somewhat justified by observing that a single system of boundaries can be generated by infinitely many distributions $P(W|A)$. In regions where MLE training would end up with too flat $P(W|A)$, leading to unreliable estimate of the boundary, MMIE tracks the boundary by using actual word errors and makes $P(W|A)$ artificially steeper where confusable words occur.

Major problem of MMIE is the amount of computation it requires. First of all, the sum in the denominator over all word-strings is prohibitive. Usually, it is replaced by calculation (21) of probability of generating acoustic output A by a fully connected ▷ 25 HMM — either unigram or bigram such as the one in fig. 6. However, this is still ▷ 27 too hard and other optimizations must take place. For instance we can first train the system by MLE, then let it recognize some held-out data in order to create word-confusion lists and then use those list in MMIE training by building a lattice from confusable words and the training text W. The lattice would represent all important (probable) misrecognitions of A in a compact way (note that it must also include the correct transcript W). The word-lattice would then be transformed into an HMM by replacing the words with their HMM models. Finally we would run (21) on the resulting lattice-HMM to obtain an approximation of the denominator.

Second implementation problem is that there is no known analog of the EM-algorithm 2.6.1. In [67] they use gradient descent search to find $\vec{\lambda}$. They developed ▷ 30 gradient formula which reuses Baum-Welsh pseudocounts (45) in the following way ▷ 33

$$\frac{\partial P_{\vec{\lambda}}(W|A)}{\partial \lambda_p} = \lambda_p^{-1} \sum_{k=1}^{K} \Big(c_p(\vec{a}_k, \vec{w}_k) - c_p\left(\vec{a}_k, \mathcal{L}(\vec{w}_k)\right) \Big) \tag{56}$$

where $c_p(\vec{a}_k, \vec{w}_k)$ is a pseudocount of p-th parameter computed by (45) on the k-th training segment, while the pseudocount $c_p\left(\vec{a}_k, \mathcal{L}(\vec{w}_k)\right)$ uses the lattice-HMM described above, instead of transcript-HMM of fig. 7. This gradient is then used to update $\vec{\lambda}$ in ▷ 30 the following way

$$\vec{\lambda}_{n+1} := \vec{\lambda}_n + \rho \nabla P_{\vec{\lambda}}(W|A) \tag{57}$$

The step size ρ has to be selected empirically. For too small steps, the convergence is slow, for large ones it might not converge at all. The search is started from $\vec{\lambda}_1$ obtained by MLE training which is also used to initialize word confusion lists. These lists have to be recreated once in a while as the training proceeds, using current $P_{\vec{\lambda}_n}(A|W)$.

According to [46], the relative WER improvement of the MMIE training over the same system trained with MLE rule was about 18%. They also report substantial increase of $P(W|A)$ — the probability the model assigns to the training data W given their acoustic evidence A — from about 10^{-190} for MLE trained system to about 0.07 observed in MMIE system. The system from [67] shows 30% relative WER improvement. Both systems were small vocabulary ones, though.

3 Probability and Information Theory

This chapter summarizes basic notions of information theory, mainly the entropy and mutual information and some properties of noisy channels. Those familiar with the subject are invited to skip it entirely.

3.1 Probability

To base the probability on solid ground I will list axioms which are assumed in this book. They are equivalent to the Kolmogorov axioms restricted to countable domains. This is also closely related to approach taken by Cox and Jaynes [38], who showed how similar axioms can be derived from natural conditions we expect from a rational reasoning that has to base its decisions on incomplete input information.

3.1.1 Definition *Probability*

Let Ω be countable set of basic events and[1] let $\Pr_\Omega : \mathcal{P}(\Omega) \to \mathbb{R}$ be a function with the following properties:

$$\begin{array}{rl} \text{normalization} & \Pr_\Omega(\Omega) = 1 \\ \text{non-negativity} & \forall a \subseteq \Omega : \Pr_\Omega(a) \geq 0 \\ \text{additivity} & \forall A \subseteq \mathcal{P}(\Omega), (\forall a, b \in A, a \neq b : a \cap b = \emptyset) : \\ & \qquad \Pr_\Omega\left(\bigcup A\right) = \sum_{a \in A} \Pr_\Omega(a) \end{array} \tag{58}$$

Then we call $\langle \Omega, \Pr_\Omega \rangle$ a *probability space* with the *probability distribution* $\Pr_\Omega$ on a carrier set (or *domain*) Ω. Usually, we only write $\Pr$ instead of $\Pr_\Omega$, if there is no danger of confusion.

3.1.2 Definition *Conditional Probability*

For a probability space $\langle \Omega, \Pr_\Omega \rangle$ and sets $A, B \subseteq \Omega$ such that $\Pr_\Omega(B) > 0$ we define the *conditional probability* $\Pr_\Omega : \mathcal{P}(\Omega) \times \mathcal{P}(\Omega) \to \mathbb{R}$ as $\Pr_\Omega(A|B) := \Pr_\Omega(A \cap B) / \Pr_\Omega(B)$. For fixed B, the pair $\langle B, \Pr_B \rangle$, where $\Pr_B(A) := \Pr_\Omega(A|B)$ for $A \subseteq B$, is also a probability space according to definition 3.1.1. Often, we omit index Ω in $\Pr_\Omega(A|B)$.

3.1.3 Note The above definition tells us what is the probability of intersection. It is $\Pr_\Omega(A \cap B) = \Pr_\Omega(A|B) \Pr_\Omega(B)$, when $\Pr_\Omega(B) \neq 0$, otherwise it is 0, since $A \cap B = \emptyset$. We can also use it to determine probability of a union of countably many events $a_i \subseteq \Omega$. All we have to do is to define $b_k := a_k \cap \left(\Omega \setminus \bigcup_{j=0}^{k-1} a_j\right)$ getting $\Pr\left(\bigcup_{j=0}^{\infty} a_k\right) = \sum_{k=0}^{\infty} \Pr(b_k)$. This follows from additivity axiom as b_ks are disjoint. Probability of intersection is then $\Pr\left(\bigcap_{j=0}^{\infty} a_k\right) = \Pr\left(\Omega \setminus \bigcup_{j=0}^{\infty} (\Omega \setminus a_k)\right) = 1 - \Pr\left(\bigcup_{j=0}^{\infty} (\Omega \setminus a_k)\right)$.

Now what does it mean practically. We consider Ω to be a space for the model of the world we are interested in. For instance, if we wanted to talk about noisy channels, it would consist of all pairs of all finite strings over some fixed finite alphabet V — formally $\Omega = V^+ \times V^+$, where V^+ is a set of all non-empty strings over alphabet V, that is $V^+ = \bigcup_{k=1}^{\infty} V^k$, where $V^1 = V$, $V^2 = V \times V$, etc. $\Pr_\Omega(\{\langle a, b \rangle\})$ would then mean a probability that the input string a gets transformed into string b by the channel. Once we have $\Pr_\Omega$ given, we can ask what is the probability of generating certain output b_0. That is, we are interested in all events containing b_0 regardless of a. This event corresponds to a set $S = \{x \mid \exists a : x = \langle a, b_0 \rangle \ \& \ a \text{ is a string over } V\}$, and its probability is therefore $\Pr_\Omega(S)$. Unfortunately, the set theoretic notation gets tedious as we try to talk about more complicated things. For this reason, casual notation is commonly used — instead of a set, we write formula $\varphi(\omega)$ into the parentheses of $\Pr$. The meaning of $\Pr(\text{“}\varphi(\omega)\text{”})$ is then $\Pr_\Omega(\{\omega \in \Omega \mid \varphi(\omega)\})$. Our noisy channel example would be written as $\Pr(\text{“}\exists a : \omega = \langle a, b_0 \rangle \ \& \ a \text{ is a string over } V\text{”})$. In this new notation, conjunctions and disjunctions correspond to intersections and unions, respectively. The following definition makes even more compact notation possible.

3.1.4 Definition *Random Variable*

Let us be given a probability space Ω. Then the function $X : \Omega \to \mathbb{X}$ is called a *random variable* over the set $\mathbb{X}$.

Often, the domain of probability space can be written as a Cartesian product, such as $\Omega = \mathbb{X}_1 \times \mathbb{X}_2 \times \cdots \times \mathbb{X}_n$. The notable random variables $X_i : \Omega \to \mathbb{X}_i$ are projections from Ω to X_i. In our example, where $\Omega = V^+ \times V^+$, we have projections $A(\langle a, b \rangle) := a$ and $B(\langle a, b \rangle) := b$. This allows us to write $\Pr_\Omega(\{x \mid \exists a : x = \langle a, b_0 \rangle \ \& \ a \in V^+\})$ as $\Pr_\Omega(\text{“}B(\omega) = b_0\text{”})$, or even more shortly as $\Pr(B = b_0)$, which is the standard notation. In this section, I will stay with the rich notation using quotation marks and ω to avoid confusion. But in the rest of the book I will use standard notation in its full power.

Random variables are especially useful when we want to describe more involved properties of sets. For example let $l : V^+ \to \mathbb{N}$ be a function returning the length of the string. Simply by writing $\Pr(\text{“}l(B(\omega)) < l(A(\omega))\text{”})$ we mean a probability that the output string will be shorter than the input one.

3.1.5 Definition *Image and Preimage of a Set,* Rng*,* Dom

For sets $\mathcal{A}$ and $\mathcal{B}$ and function $F : \mathcal{A} \to \mathcal{B}$, $F[A]$ stands for an *image* of a set $A \subseteq \mathcal{A}$, being defined as $F[A] := \{F(a) \mid a \in A\}$ and $F^{-1}[B]$ for a *preimage* of $B \subseteq \mathcal{B}$, which is defined as $F^{-1}[B] := \{a \in \mathcal{A} \mid F(a) \in B\}$. Also, I shall write $\mathrm{Rng}(F)$ for $F[\mathcal{A}]$ and $\mathrm{Dom}(F)$ for $\mathcal{A}$.

3.1.6 Observation *Let us have a probability space $\langle \Omega, \Pr_\Omega \rangle$ and a random variable $X : \Omega \to \mathbb{X}$, s.t. $\mathrm{Rng}(X) = \mathbb{X}$. Let us define $\Pr_\mathbb{X} : \mathcal{P}(\mathbb{X}) \to \mathbb{R}$ in the following way:*

$$\Pr_\mathbb{X}(M) := \Pr_\Omega\left(\{\omega \in \Omega \mid \exists m \in M : X(\omega) = m\}\right) = \Pr_\Omega\left(X^{-1}[M]\right) \tag{59}$$

Then the pair $\langle \mathbb{X}, \Pr_\mathbb{X} \rangle$ is a probability space, too.

Proof Evident. Q.E.D.

3.1.7 Note The space $\langle \mathbb{X}, \Pr_\mathbb{X} \rangle$ will be denoted as $\langle X[\Omega], \Pr_{X[\Omega]} \rangle$.

[1] $\bigcup A := \{x \mid \exists X \in A : x \in X\}$ and $\mathcal{P}(X)$ stands for all subsets of X, that is $\mathcal{P}(X) := \{A \mid A \subseteq X\}$.

3.1.8 Note It should be explained what is meant by the notation $\Pr(w_0 \mid w_1, w_2)$ used in (4). Intuitively, it means the probability of encountering w_0 preceded by w_1, w_2 for a randomly selected trigram from the text. So, casually written[2] it should be

$$\Pr_\Delta(\text{“}\exists y, z : T_k(\omega) = \langle w_0, y, z\rangle \ \&\ k \in \{0, \ldots, l(\omega) - 3\}\text{”} \mid \text{“}\exists x : T_k(\omega) = \langle x, w_1, w_2\rangle \ \&\ k \in \{0, \ldots, l(\omega) - 3\}\text{”}) \tag{60}$$

where $\Delta = V^+$, $l(s)$ maps the string s to its length and $T_k(s)$ is a random variable extracting k-th trigram from s. Unfortunately, this is not quite correct. The problem is that k is left unquantified in the formula. Simply adding $\exists$ quantifier does not lead to the correct result because it does not account for multiple occurrences of a trigram in a string.

To illustrate the heart of the problem let us consider the following probability space: $\Gamma = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}$ with $\Pr_\Gamma(\{\gamma\}) = 0.25$ for $\forall \gamma \in \Gamma$. Obviously, the probability of getting 0 after we randomly select an element of Γ followed by a random selection of the position within it is 0.5. But computing it as $\Pr_\Gamma(\text{“}\exists k : C_k(\gamma) = 0\text{”})$, where C_k selects k-th component of γ, gives 0.75. This is because the selection of k is also a random event but Γ is too coarse to represent it. A remedy would be to define new probability space $\langle \Gamma_1, \Pr_{\Gamma_1}\rangle$ as $\Gamma_1 := \Gamma \times \{0,1\}$ and $\Pr_{\Gamma_1}(\{\gamma\}) := 0.5 \Pr_\Gamma(G(\gamma))$, where G is a random variable extracting the first part from elements of Γ_1. Then the probability we are after could be written as $\Pr_{\Gamma_1}(\text{“}C_{S(\gamma)}(G(\gamma)) = 0\text{”})$, where S is a random variable extracting second part from $\gamma \in \Gamma_1$.

Using this trick we can define $\Delta_1 := \Delta \times \mathbb{N}$, distributing the original probability among possible selections uniformly: $\Pr_{\Delta_1}(\{\langle \omega, k\rangle\}) := \Pr_\Delta(\omega)/l(\omega)$ for $k < l(\omega)$ and 0 otherwise. Then, let us define trigram-extracting random variable $T : \Delta_1 \to V^3$ as $T(\{\langle \omega, k\rangle\}) := \langle U_k(\omega), U_{k-1}(\omega), U_{k-2}(\omega)\rangle$, where $U_k(s)$ extracts k-th letter from the string (being defined arbitrarily for $k < 0$). The probability we were originally interested in is then

$$\Pr_{\Delta_1}(\text{“}S(\omega) \geq 2 \ \&\ \exists y, z : T(\langle G(\omega), S(\omega)\rangle) = \langle w_0, y, z\rangle\text{”} \mid \text{“}S(\omega) \geq 2 \ \&\ \exists x : T(\langle G(\omega), S(\omega)\rangle) = \langle x, w_1, w_2\rangle\text{”}) \tag{61}$$

We can easily return from the new space to the old one via a random variable G if we use 3.1.6, obtaining $\langle \Delta, \Pr_\Delta\rangle = \langle G[\Delta_1], \Pr_{G[\Delta_1]}\rangle$.

Alternatively, we could enrich the alphabet V by ‘·’-character to build a probability space $\langle \Delta_2, \Pr_{\Delta_2}\rangle$, where $\Delta_2 := (V \cup \{\cdot\})^+$, and $\Pr_{\Delta_2}(\omega) := \Pr_\Delta(\delta)/l(\delta)$, for $\delta \in \Delta$ and all ω generated from δ by placing dot at every possible position in the string except the last one[3]. $\Pr_{\Delta_2}(\omega) := 0$ for all other ω. The probability $\Pr(w_0, w_1, w_2)$ then means

$$\Pr_{\Delta_2}(\text{“}\exists k : T_k(\omega) = \langle w_0, \cdot, w_1, w_2\rangle\text{”}) \tag{62}$$

where the random variable $T_k : \Delta_2 \to (V \cup \{\cdot\})^4$ extracts 4 characters from a string at the offset k, returning $\cdot$ for characters which would lie outside the string. Using a random variable G which removes the dot from the string we could go back to the original space Δ by 3.1.6 as before.

[2] $\Pr_\Omega(\text{“}\varphi(\omega)\text{”} \mid \text{“}\psi(\omega)\text{”})$ stands for $\Pr_\Omega(\{\omega \in \Omega \mid \varphi(\omega)\} \mid \{\omega \in \Omega \mid \psi(\omega)\})$.

[3] For instance, in case of space Γ we would assign non-zero probability only to strings: ·00, 0·0, ·01, 0·1, ·10, 1·0, ·11 and 1·1.

3.1.9 Definition *(Conditional) Expectation Value*

For a probability space $\langle \Omega, \Pr_\Omega\rangle$, an event $E \subseteq \Omega$ such that $\Pr_\Omega(E) > 0$ and random variable $X : \Omega \to \mathbb{R} \cup \{\infty, -\infty\}$ we define the *conditional expectation value* as

$$\mathcal{E}_\Omega(X|E) := \sum_{x \in \mathrm{Rng}(X)} x \Pr_\Omega(\text{“}X(\omega) = x\text{”} \mid E) \tag{63}$$

where we take $\infty \cdot 0$ as 0, if it appears in the sum. The sum is meant to be defined only if it converges to the same number, regardless of reordering (i.e. if it is *absolutely convergent*). If it diverges (regardless of reordering) it is considered as $+\infty$ or $-\infty$. Otherwise it is undefined. For $E = \Omega$, we write $\mathcal{E}_\Omega(X)$, calling it the *expectation value*. We can write $\mathcal{E}(X)$ when Ω is clear from the context. We can also colloquially write $\mathcal{E}_{P(\omega)}(X(\omega))$ instead of $\mathcal{E}_\Omega(X)$, as it was used in (50). ▷35

3.1.10 Observation *Expectation could have been defined the following way as well.*

$$\mathcal{E}_\Omega(X|E) = \sum_{\omega \in \Omega} X(\omega) \Pr_\Omega(\{\omega\} \mid E) \tag{64}$$

where again $\infty \cdot 0 := 0$.

Proof $\Pr_\Omega(\text{“}X(\omega) = x\text{”} \mid E) = \Pr_\Omega(\{\omega | X(\omega) = x\}|E) = \sum_{\omega \in \Omega, X(\omega) = x} \Pr_\Omega(\{\omega\}|E)$. Plugging this into (63), we get (64). Q.E.D.

3.1.11 Note $\mathcal{E}$ is linear ($\mathcal{E}(X + \alpha Y) = \mathcal{E}(X) + \alpha\mathcal{E}(Y)$), monotonic ($X < Y$ implies $\mathcal{E}(X) < \mathcal{E}(Y)$) and satisfies triangle inequality ($|\mathcal{E}(X)| \leq \mathcal{E}(|X|)$ for any X). Generally, however, $\mathcal{E}(X \cdot Y) \neq \mathcal{E}(X)\mathcal{E}(Y)$. If the equality holds, we call X and Y *uncorrelated*.

3.1.12 Definition *Expectation Conditioned by Random Variable*

Let us be given a probability domain Ω and random variables $X, Y : \Omega \to \mathbb{R} \cup \{\infty, -\infty\}$, such that $\Pr(\text{“}Y(\omega) = y\text{”}) > 0$ for all $y \in \mathrm{Rng}(Y)$. Then, the partial function $\mathcal{E}_\Omega(X|Y) : \mathrm{Rng}(Y) \to \mathbb{R} \cup \{\infty, -\infty\}$ is defined in the following way[4]:

$$\mathcal{E}_\Omega(X|Y) := \lambda y. \mathcal{E}_\Omega(X \mid \text{“}Y(\omega) = y\text{”}) \tag{65}$$

3.1.13 Note According to 3.1.6, $\mathcal{E}_\Omega(X|Y)$ is a random variable in probability space $\langle \mathrm{Rng}(Y), \Pr_{\mathrm{Rng}(Y)}\rangle$, if the expectation converges for each $y \in \mathrm{Rng}(Y)$. It is an easy exercise to check that $\mathcal{E}_{\mathrm{Rng}(Y)}(\mathcal{E}_\Omega(X|Y)) = \mathcal{E}_\Omega(X)$ then.

3.1.14 Definition *Convex Function*

A function $f : D \to \mathbb{R}$, $D \subseteq \mathbb{R}$, is said to be *convex over an interval* $J \subseteq D$ iff for any $a, b \in J$ s.t. $a \neq b$ and $\lambda \in (0,1)$ the following holds:

$$f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda) f(b) \tag{66}$$

That is, if the function plot always lies below any chord. Function f is called *convex* iff it is convex on $\mathrm{Dom}(f) = D$ which must be an interval then. The function is called *strictly convex* iff it is convex and an equality never occurs in (66). Function f is *concave* iff $-f$ is convex.

[4] $\lambda x.T$, where T is a term with free variable x, denotes so called *lambda abstraction* that is an anonymous function $x \mapsto T(x)$. This comes from the *lambda calculus* but here it will be used merely as a notation shortcut. So, instead of writing “let $X(\omega) := 1 + l(\omega)$, consider $\mathcal{E}(X)$” we can use more compact “consider $\mathcal{E}(\lambda\omega.(1 + l(\omega)))$”. Note that the lambda is lexical part of the construct and should not be confused with other occurrences of lambda designating a variable. For instance, the term $\lambda + \lambda\lambda.(\lambda + 1)$ is valid and equivalent to $\lambda + \lambda x.(x + 1)$, where the first occurrence of lambda is a variable.

3.1.15 Lemma *Convex function f on an open interval J is continuous.*

Proof Assume for contradiction that we have $x_0 \in J$ and $\varepsilon_0 > 0$ such that $\forall \delta > 0 : \exists c \in J,\ |x_0 - c| < \delta : |f(x_0) - f(c)| \geq \varepsilon_0$. Take $a, b \in J$ s.t. $a < x_0 < b$ and select δ_0 such that $\alpha < x_0 - \delta_0$ and $x_0 + \delta_0 < \beta$, where α is an intersection of the line-segment going from point a to x_0 with $y = f(x_0) - \varepsilon_0$ and β is analogical point for segment from x_0 to b (see fig. 8). By an assumption, there is a point c in a δ-belt which lies off the ε-belt.

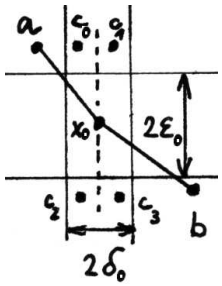


Fig. 8.

Now it is a matter of tedious but simple checking that all 12 possible configurations[5] lead to the situation where we can identify $f(c)$ or $f(x_0)$ to be lying above a chord whose endpoints were suitably chosen from the set $\{a, b, x_0, c\}$, which would be in contradiction with convexity. Hence the function must have been continuous.

Q.E.D.

3.1.16 Note A convex function on a closed interval may fail to be continuous in its endpoints.

3.1.17 Observation *Function $f : J \to \mathbb{R}$ whose $\partial^2 f / \partial x^2$ exists, being positive on interval J, is strictly convex on J.*

Proof Take $g(\lambda) := f\big(\lambda a + (1-\lambda)b\big) - \lambda f(a) - (1-\lambda) f(b)$ for $a, b \in J$ s.t. $a < b$. Then $g''(\lambda) > 0$ for $\lambda \in (0,1)$ and all we have to prove is that $g(\lambda) < 0$. From Rolle's theorem there is $\lambda_0 \in (0,1)$ such that $g'(\lambda_0) = 0$. Moreover, g' is increasing on J since $g'' > 0$. By integrating and back-integrating g'' from λ_0 we obtain $g'(0) < 0$ and $g'(1) > 0$. This makes $g(\lambda) < 0$ on $(0, \eta) \cup (1-\eta, 1)$ for small enough $\eta > 0$. If there was λ_1 for which $g(\lambda_1) > 0$, there would also be $0 < \lambda_2 < \lambda_1$ such that $g(\lambda_2) = 0$ because g would be a continuous function of changing sign and it would have to intersect zero as such. But once g would get above zero it could not get back to $g(1) = 0$ (note that $\lambda_0 < \lambda_2$ and $g'(\lambda_2) > 0$ as g is decreasing on $(0, \lambda_0)$ and increasing on $(\lambda_0, 1)$). Thus g is negative on whole $(0,1)$. Q.E.D.

3.1.18 Theorem *(Jensen's Inequality) For a probability space $\langle \Omega, \Pr_\Omega \rangle$, an interval $D \subseteq \mathbb{R} \cup \{-\infty, \infty\}$, random variable $X : \Omega \to D$ and convex function $f : D \to \mathbb{R} \cup \{-\infty, \infty\}$ the following holds[6], provided that both $\mathcal{E}_\Omega(X)$ and $\mathcal{E}_\Omega(f \circ X)$ are defined (finite or infinite) and that f is continuous in the endpoints of D if there are any.*

$$\mathcal{E}_\Omega(f \circ X) \geq f(\mathcal{E}_\Omega(X)) \tag{67}$$

Proof For more compact notation, let us write x_k instead of $X(\omega_k)$ and λ_k for $\Pr_\Omega(\{\omega_k\})$, where $\Omega = \{\omega_k \mid k \in \mathbb{N},\ k \leq K\}$. Note that $K = \infty$ for infinite space Ω. Obviously $\lambda_\bullet = 1$ and we have to prove that for each $K \in \mathbb{N} \cup \{\infty\}$:

$$\forall x_k \in \mathbb{R}, \lambda_k \in \mathbb{R} \text{ s.t. } \lambda_\bullet = 1 \text{ and } \lambda_k \geq 0 \ : \quad \sum_{k=0}^{K} \lambda_k f(x_k) \geq f\left(\sum_{k=0}^{K} \lambda_k x_k\right) \tag{68}$$

[5] a and b above and below the ε-belt make up 3 cases (for which f could be convex), while c above/below the ε-belt and on the left/on the right of x_0 makes another 4 cases.

[6] $\circ$ stands for function composition, defined as $(A \circ B)(x) := A(B(x))$. Nevertheless, colloquial notation $\mathcal{E}_\Omega(\text{“}f(X)\text{”})$ instead of logical $\mathcal{E}_\Omega(f \circ X)$ is more common.

For finite K, it can be proved by induction. For $K = 1$ this obviously holds, for $K = 2$ this in fact says that f is convex, which it is. Let the theorem hold for $K - 1 \geq 1$ and let ω_k be sorted such that $\lambda_0 \neq 0$. Defining $\sigma := \sum_{k=0}^{K-1} \lambda_k = 1 - \lambda_K$ we have

$$\sum_{k=0}^{K} \lambda_k f(x_k) = \lambda_K f(x_K) + (1-\lambda_K) \sum_{k=0}^{K-1} \frac{\lambda_k}{\sigma} f(x_k) \geq \lambda_K f(x_K) + (1-\lambda_K) f\left(\sum_{k=0}^{K-1} \frac{\lambda_k}{\sigma} x_k\right) \tag{69}$$

Where the induction hypothesis could be used thanks to $\sum_{k=0}^{K-1} \lambda_k / \sigma = 1$. Now, using convexity and canceling σ with $1 - \lambda_K$, we obtain (68). Note that $\sigma \neq 0$ since $\lambda_0 \neq 0$ due to sorting.

For infinite Ω, we have to note that $\lim_{N\to\infty} \sum_{n=0}^{N} \lambda_n = 1$. Consequently

$$\lim_{N\to\infty} \sum_{k=0}^{N} \lambda_k f(x_k) = \lim_{N\to\infty} \sum_{k=0}^{N} \frac{\lambda_k}{\sum_{n=0}^{N} \lambda_n} f(x_k) \tag{70}$$

Now we can use finite version of the theorem to bound (70) from below by

$$\lim_{N\to\infty} f\left(\sum_{k=0}^{N} \frac{\lambda_k}{\sum_{n=0}^{N} \lambda_n} x_k\right) = f\left(\lim_{N\to\infty} \sum_{k=0}^{N} \frac{\lambda_k}{\sum_{n=0}^{N} \lambda_n} x_k\right) = f\left(\sum_{k=0}^{\infty} \lambda_k x_k\right) \tag{71}$$

where f's continuousness due to 3.1.15 allowed the limit to be moved inside f.

Q.E.D.

3.1.19 Definition *(Conditional) Independence*

For probability space $\langle \Omega, \Pr_\Omega \rangle$ and random variables $A_1, \ldots, A_n$, s.t. $A_i : \Omega \to \mathbb{A}_i$, we say that variables A_i are *conditionally (mutually) independent given an event* $E \subseteq \Omega$, writing $\perp\!\!\!\perp(A_1, \ldots, A_n \mid E)$, iff for any $a_1 \in \mathbb{A}_1, \ldots, a_n \in \mathbb{A}_n$

$$\Pr_\Omega(\text{“}A_1(\omega) = a_1 \ \&\cdots\&\ A_n(\omega) = a_n\text{”} \mid E) = \prod_{k=1}^{n} \Pr_\Omega(\text{“}A_k(\omega) = a_k\text{”} \mid E) \tag{72}$$

Note that (72) in fact means $\Pr_\Omega\left(\bigcap_{k=1}^{n} A_k^{-1}[\{a_k\}] \mid E\right) = \prod_{k=1}^{n} \Pr_\Omega\left(A_k^{-1}[\{a_k\}] \mid E\right)$. For $E = \Omega$, we just say that they are *independent* and write $\perp\!\!\!\perp(A_1, \ldots, A_n)$. In case of $n = 2$ we can write $A_1 \perp\!\!\!\perp A_2$ as well. We call variables A_i *conditionally (mutually) independent given random variable* $C : \Omega \to \mathbb{C}$ and write $\perp\!\!\!\perp(A_1, \ldots, A_n \mid C)$ iff they are independent given all events $C^{-1}[c] = \{\omega \in \Omega \mid C(\omega) = c\}$ with non-zero probability $\Pr_\Omega(C^{-1}[c])$, where $c \in C[\Omega]$.

3.1.20 Note $A \perp\!\!\!\perp B \mid C$ means that for each $c \in \mathbb{C}$, random variables A and B become independent of each other when restricted to those events ω for which $\omega \in C^{-1}[c]$.

3.1.21 Note It can easily happen that $A \perp\!\!\!\perp B \mid E$ holds but $A \perp\!\!\!\perp B \mid \Omega \setminus E$ does not.

3.1.22 Note Mutual independence implies pairwise independence (that is $A_i \perp\!\!\!\perp A_j$ for $\forall\ i \neq j$) but not vice versa (consider space $\Omega = \{001, 010, 100, 111\}$ with uniform distribution and three random variables A_k each extracting k-th letter of the string).

3.1.22 Note Independence implies uncorrelatedness but uncorrelatedness does not imply independence, in general.

3.1.24 Observation *Let us be given a probability space $\langle\Omega, \Pr_\Omega\rangle$ and random variables $A_1, \ldots, A_n$, s.t. $A_i : \Omega \to \mathbb{A}_i$, and an event $E \subseteq \Omega$. Then $\perp\!\!\!\perp(A_1, \ldots, A_n|E)$ iff for any $\mathcal{A}_1 \subseteq \mathbb{A}_1, \ldots, \mathcal{A}_n \subseteq \mathbb{A}_n$:*

$$\Pr_\Omega(\text{“}\forall k : A_k(\omega) \in \mathcal{A}_k\text{”} \mid E) = \prod_{k=1}^{n} \Pr_\Omega(\text{“}A_k(\omega) \in \mathcal{A}_k\text{”} \mid E) \tag{73}$$

Proof “(73) ⇒ (72)” is trivial. Let us assume (72). We can write the probability $p := \Pr_\Omega(\text{“}\forall k : A_k(\omega) \in \mathcal{A}_k\text{”} \mid E)$ as $\Pr_\Omega(M|E)$, where $M = \bigcap_{k=1}^{n} \bigcup_{a \in \mathcal{A}_k} \{\omega \in \Omega \mid A_k(\omega) = a\} = \bigcup_{\vec{a} \in \vec{\mathcal{A}}} W(\vec{a})$, where $W(\vec{a}) = \{\omega \in \Omega \mid \forall k : A_k(\omega) = a_k\}$. Since $W(\vec{a}) \cap W(\vec{b}) = \emptyset$ whenever $\vec{a} \neq \vec{b}$, we can use additivity axiom, getting $p = \sum_{\vec{a} \in \vec{\mathcal{A}}} \Pr(\text{“}\forall k : A_k(\omega) = a_k\text{”} \mid E)$. Now, using assumption (72), we get $p = \sum_{\vec{a} \in \vec{\mathcal{A}}} \prod_{k=1}^{n} \Pr(\text{“}A_k(\omega) = a_k\text{”} \mid E) = \prod_{k=1}^{n} \sum_{a_k \in \mathcal{A}_k} \Pr(\text{“}A_k(\omega) = a_k\text{”} \mid E) = \prod_{k=1}^{n} \Pr_\Omega(\text{“}A_k(\omega) \in \mathcal{A}_k\text{”} \mid E)$. Q.E.D.

True values of probability distribution Pr are usually unreachable to us[7]. There are at least two practical methods to deal with this. Both postulate that certain random variables definable in a probability space are independent (or conditionally independent). This is inevitable — otherwise the training data would be too sparse to say anything useful about $\Pr_\Omega : \mathcal{P}(\Omega) \to \mathbb{R}$. We have already encountered this in section 2.3, where N-gram models were introduced. ◁ 17

The first method which is mainly followed in this book goes along the lines of ‘frequencist approach’, as it tries to estimate $\Pr_\Omega$ from the training data. Although there are many methods achieving that, starting with maximum likelihood equipped with various smoothing algorithms and ending with maximum entropy methods, they all have one thing in common. They compute their estimates from frequencies of occurrence of certain events in the training data[8].

The other method is ‘Bayesian’ in its nature. It assumes that Pr itself is random, meaning that both training and testing data were drawn from some fixed distribution Pr, which was randomly selected from distribution of Pr-distributions before. As the set of all probability distributions is uncountable, our simple probability axioms cannot be used here and we should work in full Kolmogorov system[9]. This method elegantly side-steps the need for smoothing and, theoretically, it also allows to incorporate general prior knowledge about the distribution, such as the Zipf's law[22]. Nevertheless, it has its own problems, too. First of all it leads to more complicated formulas requiring substantially larger computational power, and secondly it is more sensitive to Pr's violation of postulated independencies. It is explored to some extent in my previous work [42, 43]. ▷ 64

[7] Except in cases where Ω would represent outputs of deterministic machine which generates observable output $x \in \Omega$, based on a hidden N-bit number, according to the rules that are known to us. Each time the machine is used, it also changes its hidden input number such that it will appear again only after all other numbers have been used. So, if we come to the machine in an unknown state, not knowing its previous output, we can expect specific result x with probability $\Pr_\Omega(\{x\})$, for if we had 2^N such machines each one being started in different state, we would observe that exactly $2^N \Pr_\Omega(\{x\})$ machines returned x. This is also general (albeit slow) method how the function Pr could be exactly determined from the machine's description.

The above example can be reformulated on probability space $\Gamma = \{0,1\}^N \times \Omega$, where the first part represents the input number y. The probability $\Pr_\Gamma(\{x\}|\{y\})$ would be 1 only for x computed by a machine from the input $y \in \{0,1\}^N$ and 0 for otherwise. The observable probability would then be

$$\Pr_\Omega(\{x\}) = \sum_{y \in \{0,1\}^N} \Pr_\Gamma(\{x\}|\{y\}) \Pr_\Gamma(\{y\}) \tag{74}$$

where $\Pr_\Gamma(\{y\})$ was uniform in the above example.

Note that once we have computed $\Pr_\Omega$, we do not need to know internal structure of that machine anymore. In fact, different machines can lead to identical $\Pr_\Omega$. They can differ not only in the way exactly how they compute x from y but even the results of their computation can differ, as long as value of (74) does not change. This allows us to abstract from detailed properties of the machine, treating many of them at the same time.

Note that there are only countably many machines, while there are uncountably many possible functions Pr, therefore majority of probability spaces are really unreachable for us. But they can be approximated to arbitrary precision if Ω was finite.

3.2 Entropy and Information

Let us begin with formal definitions, followed by an explanation 3.2.12 of its meaning in the real world. ▷ 49

3.2.1 Definition *Entropy*

For a probability space $\langle\Omega, \Pr_\Omega\rangle$ we define its *entropy* as follows.

$$H_{\langle\Omega, \Pr_\Omega\rangle} := \sum_{\omega \in \Omega} \Pr_\Omega(\{\omega\}) \log_2 \frac{1}{\Pr_\Omega(\{\omega\})} = \mathcal{E}_\Omega(-\log_2 \circ P) \tag{75}$$

where random variable[10] $P(\omega) := \Pr_\Omega(\{\omega\})$ and we treat $0 \cdot \log_2(0)$ as 0, in accordance with 3.1.9. It is often useful to speak about the *entropy of random variable*. For random variable $X : \Omega \to \mathbb{X}$ it is defined as an entropy of subspace $\langle X[\Omega], \Pr_{X[\Omega]}\rangle$, which leads to the following formula, by applying 3.1.6 in (75). ▷ 42 ▷ 40

$$H_\Omega(X) := H_{\langle X[\Omega], \Pr_{X[\Omega]}\rangle} = -\sum_{x \in \mathrm{Rng}(X)} \Pr_\Omega\left(X^{-1}[\{x\}]\right) \log_2 \left(\Pr_\Omega\left(X^{-1}[\{x\}]\right)\right) \tag{76}$$

For $\mathbb{X} = \{x_1 \ldots x_n\}$, we can write $H(X)$ as $H(p_1, \ldots, p_{n-1})$, where $p_i := \Pr(\text{“}X = x_i\text{”})$.

3.2.2 Note The entropy is measured in units called *bits*. Bit stands for *BInary digiT* and it also denotes single place in computer memory, capable of holding 0 or 1. As such, it is used as a unit of memory capacity. Theorem 3.2.13 will reveal close connection between these two meanings. ▷ 50

[8] To avoid confusion with true probability $\Pr_\Omega$, I will write P for the frequency-based estimate. $\langle\Omega, P\rangle$ is also a probability space, hopefully only slightly differing in values of P from $\langle\Omega, \Pr_\Omega\rangle$.

[9] I will not go into it here, to save space. See [38] for its axioms. In our countable system we can at least make arbitrarily close approximation of it by representing the distribution of distributions by sufficiently dense countable grid of points in the space of all possible distributions Pr.

[10] That it depends on $\Pr_\Omega$ should not confuse us. Definition 3.1.4 does not forbid that. ▷ 40

3.2.3 Definition *Joint Entropy*

Entropy $H_\Omega(X,Y)$ of a pair of random variables is defined as an entropy $H_\Omega(X\times Y)$ of random variable $X\times Y:\Omega\to\mathbb{X}\times\mathbb{Y}$, where $(X\times Y)(\omega):=\langle X(\omega),Y(\omega)\rangle$.

3.2.4 Definition *Conditional Entropy*

Entropy of random variable X *conditioned by an event* $E\neq\emptyset$, $E\subseteq\Omega$ is:

$$H_\Omega(X|E):=-\sum_{x\in X[\Omega]}\Pr{}_\Omega(\text{“}X=x\text{”}\mid E)\log_2\Pr{}_\Omega(\text{“}X=x\text{”}\mid E)\tag{77}$$

Entropy *conditioned by a random variable* $Y:\Omega\to\mathbb{Y}$ is defined the following way.

$$\begin{aligned}H_\Omega(X|Y)&:=\sum_{y\in Y[\Omega]}\Pr{}_\Omega(\text{“}Y=y\text{”})\,H_\Omega(X\mid\text{“}Y=y\text{”})\\&=-\sum_{\substack{x\in X[\Omega]\\y\in Y[\Omega]}}\Pr{}_\Omega(\text{“}X=x\ \&\ Y=y\text{”})\log_2\Pr{}_\Omega(\text{“}X=x\text{”}\mid\text{“}Y=y\text{”})\end{aligned}\tag{78}$$

where the sum ranges only over those y for which $\Pr_\Omega(\text{“}Y=y\text{”})>0$.

3.2.5 Lemma *(Chain Rule)* $H_\Omega(X,Y)=H_\Omega(X|Y)+H_\Omega(Y)$.

Proof Expanding $H_\Omega(X|Y)+H_\Omega(Y)$ we get:

$$\begin{aligned}&-\sum_{\substack{x\in X[\Omega]\\y\in Y[\Omega]}}\Pr{}_\Omega(\text{“}X=x\ \&\ Y=y\text{”})\log_2\Pr{}_\Omega(\text{“}X=x\text{”}\mid\text{“}Y=y\text{”})\\&-\sum_{\substack{x\in X[\Omega]\\y\in Y[\Omega]}}\Pr{}_\Omega(\text{“}X=x\ \&\ Y=y\text{”})\log_2\Pr{}_\Omega(\text{“}Y=y\text{”})\\&=-\sum_{\substack{x\in X[\Omega]\\y\in Y[\Omega]}}\Pr{}_\Omega(\text{“}X=x\ \&\ Y=y\text{”})\log_2\Pr{}_\Omega(\text{“}X=x\ \&\ Y=y\text{”})=H(X,Y)\end{aligned}\tag{79}$$

Q.E.D.

3.2.6 Note Last lemma can be generalized into $H(X,Y|Z)=H(X|Y,Z)+H(Y|Z)$.

3.2.7 Note For random variables s.t. $\perp\!\!\!\perp(X_1,X_2\ldots X_n)$ we have

$$H_\Omega(X_1,X_2\ldots X_n)=\sum_{k=1}^{n}H(X_i)\tag{80}$$

which follows from 3.2.5, since $H_\Omega(X|Y)=H_\Omega(X)$ for $X\perp\!\!\!\perp Y$.

3.2.8 Observation $H(f(A))\leq H(A)$ *for any deterministic function* $f:\mathbb{A}\to\mathbb{B}$.

Proof Knowing A, $f(A)$ does not bring any new information, hence $0=H(f(A)|A)=H(f(A),A)-H(A)\geq H(f(A))-H(A)$, therefore $H(A)\geq H(f(A))$. Q.E.D.

3.2.9 Note The entropy $H_{\langle\Omega,\Pr_\Omega\rangle}$ of a distribution with finite carrier set Ω is always bounded from above by $\log_2\#\Omega$. This follows from Jensen's inequality used on a concave function (in which case the inequality is reverse than in 3.1.18). Taking a random variable P coming from definition 3.2.1 and concave function $\log_2$ we obtain: ▷ 43

$$H_{\langle\Omega,\Pr_\Omega\rangle}=\mathcal{E}_\Omega\Big(\lambda\omega.\log_2\frac{1}{P(\omega)}\Big)\leq\log_2\mathcal{E}_\Omega\left(\frac{1}{P}\right)=\log_2\sum_{\substack{\omega\in\Omega\\P(\omega)\neq0}}\frac{P(\omega)}{P(\omega)}=\log_2 N\tag{81}$$

where N is the number of basic events $\omega\in\Omega$ with non-zero probability. This upper bound is achieved by a uniform distribution.

For infinite Ω, the entropy may be infinite. An example distribution can be constructed by first dividing the $[0,1]$-interval into infinitely many pieces of probability mass $q_n:=6\pi^{-2}n^{-2}$ (note that $q_\bullet=1$), followed by subdividing each segment to 2^n equal parts. This leads to the following distribution

$$p_{m,n}:=2^{-n}q_n=6\pi^{-2}n^{-2}2^{-n}\tag{82}$$

where — formally — Ω would consist of all integer pairs $\langle m,n\rangle$, such that $n>0$ and $0<m\leq 2^n$. The entropy is then

$$-\sum_n\sum_{m=1}^{2^n}\frac{6}{\pi^2}n^{-2}2^{-n}\log_2\left(6\pi^{-2}n^{-2}2^{-n}\right)=\sum_n\frac{6}{\pi^2}2^nn^{-2}2^{-n}\left(n+2\log_2 n-c\right)\tag{83}$$

Up to a constant, this boils down to $\sum_n n^{-2}(n+2\log_2 n-c)$ where already the first term in the parenthesis causes the series to diverge, a fate which the constant c cannot save us from. On the contrary, the following observation shows that the entropy can be finite even for infinite distributions.

3.2.10 Observation *The entropy is finite for probability space* $\langle\Omega,\Pr_\Omega\rangle$ *for which there exists an ordering of basic events* ω_n *such that there are constants* $\alpha>0$ *and* $c>0$ *s.t. for all* $n\in\mathbb{N}\setminus\{0\}$ *we have*

$$\Pr{}_\Omega(\omega_n)\leq c\,n^{-1-\alpha}\tag{84}$$

Proof Obviously $-x\log_2 x<1$ for $x\in(0,1]$. Then also $-x^\beta\log_2 x^\beta<1$ for any $\beta>0$, which leads to $-\log_2 x<\beta^{-1}x^{-\beta}$. Writing p_n for $\Pr_\Omega(\omega_n)$, the entropy is

$$-\sum_n p_n\log_2 p_n\leq\beta^{-1}\sum_n p_np_n^{-\beta}\leq\beta^{-1}c\sum_n n^{-(1+\alpha)(1-\beta)}c^{1-\beta}\tag{85}$$

where the sum is finite whenever $(1+\alpha)(1-\beta)>1$, which could have been achieved by choosing sufficiently small β in the beginning. Q.E.D.

3.2.11 Lemma *For random variable $N : \Omega \to \mathbb{N}\backslash\{0\}$ on a space $\langle \Omega, \Pr_\Omega \rangle$ which has a finite entropy and expectation, we have:*

$$H_\Omega(N) < 1 + 2\log_2 \mathcal{E}_\Omega(N) \tag{86}$$

Proof Let p_k denote $\Pr_\Omega\left(N^{-1}[\{k\}]\right)$. Let $m := \{k \in \mathbb{N}\backslash\{0\} \mid p_k > 0\}$. Using Jensen's inequality on convex function $-\log_2$ we get:

$$\sum_{k\in m} p_k(-\log_2)\frac{k^{-2}}{2p_k} \geq -\log_2 \sum_{k\in m}\frac{k^{-2}p_k}{2p_k} \geq \log_2 \frac{1}{\sum_{k=1}^{\infty}\frac{1}{2}k^{-2}} > \log_2 \frac{2}{1+\int_1^\infty x^{-2}dx} \tag{87}$$

The denominator is 2. Hence (87) ≥ 0, leading to $-\sum_{k=1}^{\infty} p_k \log_2 2p_k < \sum_{k=1}^{\infty} p_k \log_2 k^2$. From Jensen's inequality (used reversely on concave function $\log_2$) we finally obtain $H_\Omega(N) < 1 + 2\sum_{k=1}^{\infty} p_k \log_2 k \leq 1 + 2\log_2 \mathcal{E}_\Omega(N)$. Q.E.D.

3.2.12 Relationship Between Entropy and Coding

Let us have an information source transmitting messages of Ω in accordance with a fixed probability distribution $\Pr_\Omega$. Let's assume that recipients know the distribution $\Pr_\Omega$ and that there is a different recipient for each message, so they cannot take advantage of statistical dependence of consecutive messages. For this reason we can as well assume the messages to be independent.

Now, the question is how to encode these messages in a binary alphabet, such that the source would use the least possible number of bits in the long run of messages (or in the limit, which frees us from definition of what we mean by long).

One possible solution is the *arithmetic coding:* Let us represent the messages as intervals on a $[0, 1)$ line, the length of each interval being equal to the probability of the respective message to be transmitted, as depicted in fig. 9. This picture is known both to the sender and to the receiver.

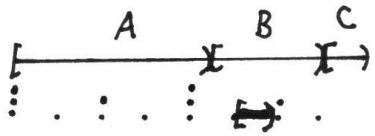


Fig. 9. *Arithmetic Coding. Dots denote possible codewords. The left point of the bottom interval is the code of message B.*

The selected message is then represented by an N-bit binary number x, which lies in its interval. As x is less than 1, only the bits after the point are transmitted and counted in N. Moreover, we want the whole interval $[x, x + 2^{-N})$ to reside in the message's interval. This would ensure that the decoder will be able to detect the end of bitstream even if there would be some extra bits after the encoded message.

So, the encoder has to find the shortest x, being N bits long, such that $[x, x+2^{-N})$ would fit inside the interval associated with the message we want to send. The decoder then reads x bit by bit, incrementing its N for each read bit and updating its interval to $[x_N, x_N+2^{-N})$, where x_N is the value of x read so far. Once this fits into an interval of certain message, the decoder knows that it has decoded entire message and which message it was.

Let us now count how many bits per message will the sender need on average, that is $\mathcal{E}_\Omega(N)$, where $N : \Omega \to \mathbb{N}$ is random variable assigning to each message $\omega \in \Omega$ the length of its codeword. Each N-bit codeword occupies an interval of length 2^{-N}. On one hand, it must fit into the message's interval, that is $2^{-N(\omega)} \leq \Pr_\Omega(\{\omega\})$. On the other hand, we don't want to waste bits by making 2^{-N} unnecessarily tiny. By setting $N(\omega) := \lceil -\log_2 \frac{1}{2}\Pr_\Omega(\{\omega\})\rceil$ there will always be N-bit number x_N such that $[x_N, x_N + 2^{-N})$ fits into message's interval. As it implies[11] $N(\omega) < -\log_2 \frac{1}{4}\Pr_\Omega(\{\omega\})$, we have:

$$\begin{aligned} \log_2 \frac{\Pr_\Omega(\{\omega\})}{4} \;<\; -N(\omega) \;\leq\; \log_2 \Pr_\Omega(\{\omega\}) \\ -\log_2 \Pr_\Omega(\{\omega\}) \;\leq N(\omega) \;<\; 2 - \log_2 \Pr_\Omega(\{\omega\}) \end{aligned} \tag{88}$$

Now, taking expectation and using its monotonicity, we obtain

$$H_{\langle\Omega,\Pr_\Omega\rangle} \leq \mathcal{E}_\Omega(N) < H_{\langle\Omega,\Pr_\Omega\rangle} + 2 \tag{89}$$

which means that the length of the arithmetic code will closely match the entropy. In the following theorem there will be shown that there is no code which would achieve $\mathcal{E}_\Omega(N)$ shorter than $H_{\Pr_\Omega}$. These findings would justify the definition of entropy as a measure of average number of bits needed to describe messages coming out of a random source with known and fixed probability distribution.

3.2.13 Theorem *(Source Coding Theorem — Shannon) Let us have the best possible, uniquely decodable mapping $C : \Omega \to \{0,1\}^+$ for a source of messages $\omega \in \Omega$ with distribution $\Pr_\Omega$. The code C must be self-delimiting — it must be possible to determine the end of the codeword from the codeword alone, even if it is followed by arbitrary extra bits. By 'best' it is meant that $\mathcal{E}_\Omega(N)$, the expected length of the output codeword, is minimal, where $N(\omega) := l(C(\omega))$ is the length of the code of message ω. Then*

$$H_{\langle\Omega,\Pr_\Omega\rangle} \leq \mathcal{E}_\Omega(N) < H_{\langle\Omega,\Pr_\Omega\rangle} + 2 \tag{90}$$

Proof To prove $\mathcal{E}_\Omega(N) < H_{\langle\Omega,\Pr_\Omega\rangle} + 2$, just consider C to be an arithmetic coding. The lower bound can be proved by contradiction. Let us assume that for certain fixed $\langle\Omega, \Pr_\Omega\rangle$ we would have a code $C : \Omega \to \{0,1\}^+$ for which $\mathcal{E}_\Omega(N) = H_{\langle\Omega,\Pr_\Omega\rangle} - \varepsilon$, where $\varepsilon > 0$. Since random variable C is one-to-one mapping we have $H_\Omega(C) = H_{\langle\Omega,\Pr_\Omega\rangle}$. Let us define space $\langle\Gamma, \Pr_\Gamma\rangle$, for suitable n (which will be determined later) such that $\Gamma := \Omega^n$ and

$$\Pr_\Gamma\left(\{\langle\omega_1 \ldots \omega_n\rangle\}\right) := \prod_{k=1}^{n} \Pr_\Omega\left(\{\omega_i\}\right) \tag{91}$$

This can be thought of as using the code n times to send n randomly and independently selected messages from Ω. As C leads to self-delimiting messages, this is possible by simple concatenation of the respective bit-strings. The resulting code can be described by random variable $D = C_1 \times \cdots \times C_n$, where C_i is code C acting on i-th component of the message $\gamma \in \Gamma$:

$$D(\langle\omega_1 \ldots \omega_n\rangle) = C(\omega_1)\cdots C(\omega_n) \tag{92}$$

Let $M(\gamma) := l(D(\gamma)) = \sum_k l(C(\omega_k))$ be a random variable assigning length of code $D(\gamma)$ to the event γ. Due to the concatenation and linearity of expectation we have $\mathcal{E}_\Gamma(M) = n\,\mathcal{E}_\Omega(N)$. From (80) it follows that $H_\Gamma(D) = nH_\Gamma(C_1) = nH_\Omega(C) =$

[11] Since $\lceil x\rceil < x + 1$.

$n\,\mathcal{E}_\Omega(N) + n\varepsilon$. On the other hand $H_\Gamma(D) = H_\Gamma(D, M) = H_\Gamma(M) + H_\Gamma(D|M)$, as
48 ◁ follows from 3.2.5. Using 3.2.9, we can upper-bound $H_\Gamma(D \mid \text{“}M = m\text{”})$ by m, as there are at most 2^m distinct m-bit strings. This implies the following upper-bound on $H_\Gamma(D \mid M)$.

$$\sum_m \Pr_\Gamma(\text{“}M{=}m\text{”}) H_\Gamma(D \mid \text{“}M{=}m\text{”}) \le \sum_m m \Pr_\Gamma(\text{“}M{=}m\text{”}) = \mathcal{E}_\Gamma(M) \tag{93}$$

Combining these together we get

$$n\,\mathcal{E}_\Omega(N) + n\varepsilon = H_\Gamma(D) \le H_\Gamma(M) + \mathcal{E}_\Gamma(M) = H_\Gamma(M) + n\,\mathcal{E}_\Omega(N) \tag{94}$$

which leads to $n\varepsilon \le H_\Gamma(M)$. From 3.2.11 we have $H_\Gamma(M) \le 1 + 2\log_2 \mathcal{E}_\Gamma(M) = 1 + 2\log_2 n\,\mathcal{E}_\Omega(N)$, thus $n\varepsilon \le c + 2\log_2 n$, where c is constant. Taking n large enough, we obtain a contradiction. Q.E.D.

3.2.13 Note It is even possible to prove that $\mathcal{E}_\Omega(N) < H_{\Pr_\Omega} + 1$, using more sophisticated coding technique [39, 19].

Since code strings are self-delimiting, the arithmetic coding can be used repeatedly for encoding consecutive messages that may be even drawn from different distributions, provided that both communication ends agreed on when to use which distribution. The obvious method of code concatenation, used in the proof of 3.2.13 is not the best one, however. It is causes a 2-bit overhead, the arithmetic code has over the entropy — with each message up to 2 bits could be wasted. Fortunately, there is a better way. Instead of using the encoder separately for each message we could prepare joint distribution by embedding intervals of the second message in all intervals of the first one, as shown in fig. 10 for case of $\Omega := \{A, B, C\}$. Obviously, possible third message would be embedded in all intervals of the second one, and so on. Then, we would use our encoder just once, obtaining codeword for a sequence of N messages with overhead of just $\frac{2}{N}$ bits per message.

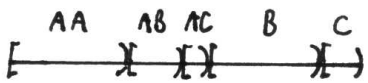


Fig. 10. *Arithmetic coding run twice. Interval embedding is drawn just for pairs starting by A to improve readability.*

Arithmetic coding, as it has been presented so far, transforms whole messages at the time. This is highly impractical, especially if we consider sending series of messages — sometimes we cannot afford to wait until the last message is formulated. Fortunately, arithmetic coding can be implemented to work incrementally, outputting the leading bits of the codeword as soon as their value becomes fixed. This may be accomplished by tracking the lower (l) and upper (u) point of the message interval.

With each input message these points would be updated to reflect nesting of message intervals. Then, l and $u{-}2^{-n}$, where n is taken to be higher than the bit-precision of numbers representing interval lengths, would be expressed as binary numbers. Leading bits that would be identical in these two numbers cannot change later, since new message intervals always fall inside the old ones. Therefore, this leading segment can be safely sent to the output. Note that the bits which were emitted as a result of processing the message may not suffice to recover it completely. Only the bits of orders 2^{-1} to 2^{-p+1}, where p is the smallest integer such that $\exists k \in \mathbb{N} : l < k2^{-p} < u$, could be emitted by the encoder after it has processed the first message. For instance, in

case of $l = 0.49$ and $u = 0.51$ this means that $p = 1$ and no bit can be emitted before following messages allow the encoder to decide whether the resulting codeword should begin with binary one or zero. Unfortunately, there is no limit on the number of input messages that would be possibly needed to get out of this dead-point.

As this is unacceptable, practical implementations try to side-step this problem by various ways. For instance, it is possible to count the number of retained messages, dynamically changing problematic interval in — say — the fourth message after the one being blocked. The encoder would simply shrink the interval the way it would begin or end at $k2^{-p}$. This, in fact, would also shrink intervals of the following messages for which reason it has to be accounted for in the decoder. This might be done by simulating what the encoder would do with the messages just decoded, accordingly correcting intervals the decoder expected, when the interval change should occur.

As the reader might have already guessed, practical implementations of the arithmetic coding do not work with messages from infinite space Ω directly. Instead, the space is described by means of strings over certain finite alphabet A. Employing the chain rule (3), we can compute the probability of any element of Ω as if it was a $\diamond$-terminated string over A. Then we are able to encode it incrementally, letter by letter. ▷ 18
Note that nothing is lost in comparision with theoretical ‘single-shot’ approach because the final intervals exactly correspond to the intervals of $\Pr_\Omega(\{\omega_i\})$, after the intervals of all the letters have been embedded. Therefore the resulting code will be of the same length, if we neglect little overhead caused by the dead-point problem resolution. Also note that when no other message from Ω is being awaited, the codeword should be ‘closed’ by shrinking the last interval to the closest power of 2 (the way we did in fig. 9), which causes the final 2 bit overhead. Otherwise, u and l may remain open, but the first message would be fully decodable only after the next message entered the encoder and enough[12] of its symbols have been processed.

The finite alphabet also makes possible to use fixed precision arithmetics — if we would represent probabilities of intervals as, say, 20-bit numbers (which is harmless as we can hardly hope for any better estimate based on real data), we could perform all the calculations within 64-bit register, provided that we defined suitable dead-point resolution rule. Matching high bits of l and $u{-}2^{-64}$ can be shifted to the output, while at the same time, shifting l and u to the left, effectively expands the interval, so that intervals of the next letter can be embedded into it without noticeable loss of precision.

A special case of binary alphabet $A = \{0, 1\}$ is interesting as it does not need to perform multiplication. It gets by with addition and shifting, which may be faster on some CPUs. The disadvantage is that for encoding a single `ASCII` character we have to invoke the encoder 8 times, so it may not pay-off after all. Also, we cannot directly use $\diamond$ as the alphabet would become ternary. The messages have to be ended by an agreed-on sequence (which would be allowed to appear only at the end of the string) instead. The only requirement is that (3) would result in a correct probability distribution. For instance we might use *‘8 consecutive zeroes, beginning on position divisible by 8’* instead of $\diamond$ symbol (this is exactly what the C language uses to terminate strings).

Finally, as all the conditional probabilities involved in (3) are typically unknown, N-gram model (4) is used to approximate the distribution of the next letter. The ▷ 19

[12] Exact number depends on probability distribution and on dead-point resolution details.

bigram case ($N=2$, that is a conditioning on previous letter) reveals an interesting connection with conditional entropy 3.2.4. Let X represent the current letter and Y the previous one. Using $P(X=x \mid Y=y)$ to determine intervals in the arithmetical coder leads to letter-codewords approximately $\log_2 P(X=x \mid Y=y)$ bits long. Total length of the resulting bit string is thus:

$$\sum_{n=1}^{\#\vec{t}-1} \log_2 P(X=t_n \mid Y=t_{n-1}) + \alpha \;=\; \sum_{x,y\,\in\, A} c(x,y)\log_2 P(X=x \mid Y=y) + \alpha \qquad (95)$$

where $\alpha \in [0,2)$. Due to the *large number theorem*, the value of $(\#\vec{t}-1)P(X=x,Y=y)$ goes to $c(x,y)$. Therefore the average code length per letter converges to $H(X|Y)$.

More broadly (returning to the 'single-shot' encoding of long messages X and Y for a while), $H(X|Y)$ may be interpreted as an average number of bits we need to add to $H(Y)$-many bits we already spent for buying message Y, in order to discover text of the message X, provided that we used optimal encoding. Note that after discovering X we still know Y so it comes at no surprise that $H(X,Y) = H(Y) + H(X|Y)$. This informally explains the chain rule 3.2.5 as well as the fact that incremental arithmetic coding yields the same efficiency as its single-shot version.

3.2.15 Relative Entropy and Mutual Information

A question of how many bits will be needed on average to encode the message, when the encoder uses imprecise probability distribution $Q(\omega)$ instead of the correct one $P(\omega)$ leads to the notion of *cross entropy*. As the codeword for ω will be of length $-\log_2 Q(\omega)$ but will appear with probability $P(\omega)$, the following definition gives the answer.

3.2.16 Definition *Cross Entropy*

For a space $\langle \Omega, \Pr_\Omega \rangle$, $P(\omega) := \Pr_\Omega(\{\omega\})$, and $Q : \Omega \to \mathbb{R}$ s.t. $Q(\omega) \geq 0$ and $Q(\bullet) = 1$ the *cross entropy* is defined as follows.

$$H_\Omega(P \,||\, Q) := -\sum_{\omega\,\in\,\Omega} P(\omega)\log_2 Q(\omega) \qquad (96)$$

3.2.17 Definition *Relative Entropy*

Let $P(\omega) := \Pr_\Omega(\{\omega\})$ for space $\langle \Omega, \Pr_\Omega \rangle$ and $Q : \Omega \to \mathbb{R}$ be s.t. $Q(\omega) \geq 0$ and $Q(\bullet) = 1$. The *relative entropy* or *Kullback-Leibner divergence* is closely related to the cross entropy, meaning the number of bits wasted in the encoder that is using an imperfect distribution Q instead of the correct one P.

$$D_\Omega(P \,||\, Q) := \sum_{\omega\,\in\,\Omega} P(\omega)\log_2 \frac{P(\omega)}{Q(\omega)} = H_\Omega(P \,||\, Q) - \underbrace{H_\Omega(P \,||\, P)}_{H_{\langle \Omega, \Pr_\Omega \rangle}} \qquad (97)$$

3.2.18 Observation $D(P \,||\, Q) \geq 0$.

Proof Using 3.1.18 we have (in accordance with "$0 \cdot \infty = 0$"-rule we skip those ω_i for which $P(\omega_i) = 0$, in $\mathcal{E}$):

$$D(P \,||\, Q) = \mathcal{E}\left(-\log_2 \circ \frac{Q}{P}\right) \geq -\log_2 \mathcal{E}\left(\frac{Q}{P}\right) = -\log_2 Q(\bullet) = 0 \qquad (98)$$

Q.E.D.

3.2.19 Definition *(Conditional) Mutual Information*

Mutual information between random variables X and Y is defined the following way.

$$\begin{aligned} I_\Omega(X;Y) &:= H_\Omega(X) + H_\Omega(Y) - H_\Omega(X,Y) = H_\Omega(X) - H_\Omega(X|Y) \\ &= \sum_{x,y} \Pr_\Omega(\text{“}X=x \;\&\; Y=y\text{”}) \log_2 \frac{\Pr_\Omega(\text{“}X=x \;\&\; Y=y\text{”})}{\Pr_\Omega(\text{“}X=x\text{”})\Pr_\Omega(\text{“}Y=y\text{”})} \qquad (99) \\ &= D_{\mathbb{X}\times\mathbb{Y}}\Big(\Pr_\Omega\left((X\times Y)^{-1}[\{\langle x,y\rangle\}]\right) \;||\; \Pr_\Omega\left(X^{-1}[\{x\}]\right)\Pr_\Omega\left(Y^{-1}[\{y\}]\right)\Big) \end{aligned}$$

When optionally conditioned by a random variable Z, it becomes:

$$\begin{aligned} I_\Omega(X;Y|Z) &:= H_\Omega(X|Z) + H_\Omega(Y|Z) - H_\Omega(X,Y|Z) \;=\; \sum_{z\,\in\, Z[\Omega]} \Pr_\Omega(\overbrace{Z^{-1}[z]}^{\mathcal{Z}})\cdot \qquad (100) \\ &D_{\mathbb{X}\times\mathbb{Y}}\Big(\Pr_\Omega\left((X\times Y)^{-1}[\{\langle x,y\rangle\}]\big|\mathcal{Z}\right) \;\Big|\Big|\; \Pr_\Omega\left(X^{-1}[\{x\}]\big|\mathcal{Z}\right)\Pr_\Omega\left(Y^{-1}[\{y\}]\big|\mathcal{Z}\right)\Big) \end{aligned}$$

Mutual information intuitively means the number of bits that can be spared in the process of encoding message X by employing our knowledge of Y which we assume is available to the decoder, too. Interestingly, it is symmetric (because its definition formula is symmetric), so we spare exactly the same amount if we used Y to help encoding X or if it was the other way round. From the definition it is clear that $I(X;Y|Z) = 0$ iff $X \perp\!\!\!\perp Y|Z$ and from 3.2.18 it follows that $I(X;Y|Z) \geq 0$. Hence we can understand mutual information to be a measure of dependence between two random variables, having maximum $H(X) = I(X;Y)$ for $X = Y$. For this reason the entropy is sometimes called the *self-information*.

3.2.20 Note From $I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$ it follows that $H(X|Y,Z) \leq H(X|Z)$ which means that additional information Y can only decrease our uncertainty about X. This inequality also implies that the right side of (80) constitutes an upper bound on joint entropy even if the variables are not independent.

3.2.21 Observation *Mutual information satisfies the following chain rule:*

$$I(X,Y;Z \mid W) = I(X;Z \mid W) + I(Y;Z \mid X,W) \qquad (101)$$

Proof $I(X,Y;Z \mid W) = H(X,Y|W) - H(X,Y|Z,W) = H(X|W) + H(Y|X,W) - H(X|Z,W) - H(Y|X,Z,W) = H(X|W) - H(X|Z,W) + H(Y|X,W) - H(Y|X,Z,W) = I(X;Z \mid W) + I(Y;Z \mid X,W)$ Q.E.D.

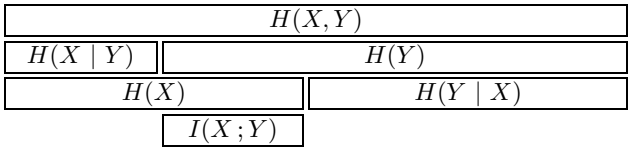
3.2.22 Note Some authors call $\log_2 \frac{P(x,y)}{P(x)P(y)}$ the *point-wise information*. Note however that this formula lacks properties of mutual information, such as non-negativity. Nevertheless, when averaged over data sample $\vec{x}, \vec{y}$, we obtain

$$\frac{1}{\#x}\sum_{k=0}^{\#x-1} \log_2 \frac{P(x_k,y_k)}{P(x_k)P(y_k)} \qquad (102)$$

which is equal to $I_\Omega(X;Y)$, provided that distribution $\Pr_\Omega\left((X\times Y)^{-1}[\{\langle x,y\rangle\}]\right) = P(x,y)$ was obtained from the data $\vec{x},\vec{y}$ by raw counting. In case we used probability smoothing in its estimation, or if it was estimated from different data set than the one over which we evaluate (102), we would obtain so called *cross mutual information* which again is not true mutual information as it may become negative, in extreme case.

3.2.23 Note Mutual information is non-negative, symmetric, but in general it does not satisfy the triangle inequality. So it is not a distance.

3.2.24 Summary The following diagram taken from [21] may be used as a reminder of basic properties of entropy and mutual information.



3.2.25 Data Processing Inequality

This subsection treats fundamental theoretical limitation on the amount of information that can be obtained from measurement as well as how it might be seemingly overcome
47◁ in practice. It may be regarded as a generalization of 3.2.8.

3.2.26 Definition *Markov Chain*

Random variables X, Y and Z over $\langle \Omega, \Pr_\Omega \rangle$ are said to form the *Markov chain* (denoted by $X \to Y \to Z$) iff for all $x \in X[\Omega]$, $y \in Y[\Omega]$ and $z \in Z[\Omega]$, s.t. $p(y) > 0$:

$$p(x, y, z) = p(z|y)p(y, x) \tag{103}$$

where $p(x)$ denotes $\Pr_\Omega(\text{“}X = x\text{”})$ for sake of brevity — analogically for y and z. Alternatively, (103) can be written as $p(z|x, y) = p(z|y)$ for all x, y, z s.t. $p(x, y) > 0$.

3.2.27 Note For x satisfying $p(x) > 0$ we can write (103) as $p(z|y)p(y|x)p(x)$. The ‘chain’ from x to z is clearly apparent in this form.

3.2.28 Observation $X \to Y \to Z$ *iff* $X \perp\!\!\!\perp Z \mid Y$.

Proof $X \to Y \to Z$ iff $p(x, y, z) = p(z|y)p(y, x)$ for all x, y, z except y s.t. $p(y) = 0$. Dividing both sides by $p(y)$ we get equivalent formula $p(x, z|y) = p(z|y)p(x|y)$ which is the definition of $X \perp\!\!\!\perp Z \mid Y$. Q.E.D.

3.2.29 Note Since $X \perp\!\!\!\perp Z \mid Y$ iff $Z \perp\!\!\!\perp X \mid Y$, the chain is always formed in both directions for which reason it is sometimes denoted as $X \leftrightarrow Y \leftrightarrow Z$.

3.2.30 Note Whenever $Z = f(Y)$ where $f : \mathbb{Y} \to \mathbb{Z}$ is a deterministic function, the variables Y and Z form Markov chain $X \to Y \to Z$ with any X. This is because
35◁ $p(z|y, x) = \Pr(\text{“}f(Y) = z\text{”} | \text{“}Y = y, X = x\text{”}) = (z = f(y))?1 : 0$ does not depend[20] on X.

3.2.31 Theorem *(Data Processing Inequality) If the following random variables form a Markov chain $A \to B \to C$, then $I(A\,;B) \geq I(A\,;C)$.*

Proof The chain rule 3.2.21 can be applied in two different ways, giving:

$$I(C\,;A) + I(B\,;A|C) = I(C, B\,;A) = I(B\,;A) + I(C\,;A|B) \tag{104}$$

Since $A \to B \to C$, we have $A \perp\!\!\!\perp C \mid B$ which implies $I(A\,;C|B) = 0$. As $I(B; A|C) \geq 0$ we obtain the desired $I(C\,;A) \leq I(B\,;A)$. Q.E.D.

3.2.32 Consequence $I(f(B)\,;A) \leq I(B\,;A)$ *for any* $f : \mathbb{B} \to \mathbb{C}$.

Data processing inequality states that no matter how clever information processing we invent (deterministic or random) we never obtain more information about A — the actual object of our interest (an image of a far galaxy, let’s say) — than is already contained in raw data B received from telescope’s camera. Any computer program, although it can increase entropy of B while converting it into C, can only decrease information that C will contain about A.

But what about all those ‘image restoration’ programs? Does it mean they should not work? Not necessarily. These programs work by combining many measurements B_i of A into single output image C. Therefore, the data processing inequality applies to $B := B_1 \times \cdots \times B_n$. And although it may happen that $I(A\,;B_i) < I(A\,;C)$, the information that C carries about A is still bounded by $I(A\,;B)$ from above.

To be precise, the measurements B_i reflect different states A_i of our object but we *assume* that it changes slowly so that we can practically regard all A_is as equal. This can be true even if individual B_is are very different of one another. Imagine that they represent a movie of passing-by car and that A we are interested in is a text on its numberplate. Then, although B_i change due to geometric projection sweep, A remains constant.

Another situation which seemingly overcomes 3.2.31 is when we would obtain higher resolution image from a page of text by using our knowledge of font outline curves. Let the whole page be printed in single font. Then, as long as the resolution of B is sufficient to recognize individual letters we are seemingly getting additional information about A from our knowledge of font shapes. But this is not so. As we have forbidden other fonts and generally other non-textual images to appear in A we caused that font shape could not be used to carry information. In another words, as font shape could not be used to convey information in this setup, it would not be counted in $H(A)$. If, on the other hand, different fonts were possible in A but the resolution would be so poor that they would be indistinguishable, then our program would sometimes fail to determine the font correctly, just to make 3.2.31 true.

3.2.33 Theorem *If $A \to B \to C$, then $I(A\,;B|C) \leq I(A\,;B)$*

Proof In the proof of 3.2.31 we can just as well point out that $I(C\,;A) \geq 0$, picking up $I(B\,;A|C)$. Q.E.D.

3.2.34 Note This shows that conditioning by a ‘downstream’ variable in the Markov chain can only decrease dependence of A and B. Note that reverse inequality is possible if random variables do not form a Markov chain.

3.2.35 Note Markov chain can defined more generally, allowing arbitrarily long chains $A_1 \to A_2 \to \cdots \to A_n$ by requiring

$$p(a_1, a_2, \ldots, a_n) = p(a_1) \prod_{k=2}^{n} p(a_k \mid a_{k-1}) \tag{105}$$

This can be shown to be equivalent with the requirement $p(a_k \mid a_{k-1}, \ldots, a_1) = p(a_k \mid a_{k-1})$ for all k, which in turn is equivalent[13] with $A_k \perp\!\!\!\perp A_1 \ldots A_{k-2} \mid A_{k-1}$.

[13] Because $A \perp\!\!\!\perp C|B$ iff $p(a|b, c) = p(a|b)$ for all a, b, c s.t. $p(b, c) > 0$.

Data processing inequality holds here, too — information can only decrease each time we traverse the arrow. Formally, for $x \le k < l \le y:\ I(A_x\,;A_y) \le I(A_k\,;A_l)$.

3.3 Noisy Channel

Imagine we have a channel which can be used for sending binary digits with only limited probability of success. Namely, if `0` was sent then it can be changed into `1` during transmission with probability p. Likewise `1` can become `0` with probability q. Apart from that, we will assume that digits cannot be lost, only misrecognized, and that subsequent uses of the channel are well separated so that digits sent in different times don't get confused and that probabilities p and q do not depend on past or future digits. Action of such a channel is symbolically drawn in fig. 11. From now on, let us assume that $p = q$. This special case is called *memoryless symmetric binary channel.*

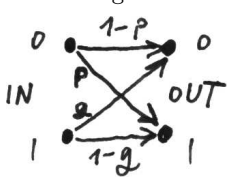


Fig. 11. *Binary channel*

If we would like to send single binary digit with less probability of error than the channel provides, we could send it $2n+1$-times, regarding $n+1$ or more received ones as `1`. As this method cannot correct more than n channel errors, probability of bit transmission error is:

$$p_e = \sum_{k=n+1}^{2n+1} \binom{2n+1}{k} p^k (1-p)^{2n+1-k} = p^{2n+1} \sum_{k=0}^{n} \binom{2n+1}{k} \left(\frac{1-p}{p}\right)^k \qquad (106)$$

Obviously, the probability of first $n+1$ bits being wrong (ignoring the rest of them), which is p^{n+1}, is a lower bound on p_e. Noting that $\sum_{k=0}^{2n+1} \binom{2n+1}{k} = 2^{2n+1}$ and using its symmetry to obtain $\sum_{k=0}^{n} \binom{2n+1}{k} = 4^n$ we get the following bounds on p_e.

$$p^{n+1} \max\left(1, 4^n p^n\right) \le p_e \le p^{2n+1} 4^n \left(\frac{1-p}{p}\right)^n = p^{n+1} 4^n (1-p)^n \qquad (107)$$

As can be seen from the last formula, in order to push p_e close to zero we have to grow the number of repetitions indefinitely. Specifically $\lim_{p_e \to 0} n(p_e) = \infty$, therefore as we strive for more reliable codes, the transfer rate falls to zero.

Before Shannon's breakthrough paper [19], it was generally believed that it must always be so. In case of one bit messages we have been sending so far it is even true[14]. Shannon's unexpected result however showed that if we took long enough block of bits to be encoded together, we could communicate with arbitrarily low probability of error at transfer rate being arbitrarily close to certain constant C — so called *channel capacity*, which depends only on properties of the channel (only on p in our elementary case). To many, this sounded counter-intuitively at that time. Remember that it was in 1948 and people were not yet so widely used to get something for no price at all. It was hence expected that better reliability must be paid by a sacrificed transfer rate.

[14] No matter how we would represent `1` and `0`, if $2n+1$ channel bits have to be used for it, up to $2n+1$ errors could always make `0` from `1`. As we want both symbols to be equally protected we again end up with at most n errors turning `0` to something the decoder will still take as zero. Therefore there is no better code than the repetition code, if it comes to sending of individual source bits.

Nevertheless, Shannon proved that for any channel there is a constant $C \ge 0$ such that for any $\varepsilon > 0$ there is a block length N and invertible coding rule transforming N-bit input x into M-bit channel code y, such that the rate of the code $R := M/N$ is higher than $C - \varepsilon$, while at the same time the probability of channel output z being decoded to erroneous $\widehat{x} \neq x$ is less than ε, given any x.

This may be intuitively justified by the following geometric argument: Imagine we group every k bits to represent single input bit by the repetition code. Then, N input bits transform into $M = Nk$ channel bits. Codewords of this channel encoding occupy 2^N points in $\{0,1\}^M$ space. The protection is achieved by their distance — two codewords are at least k bit-flips apart[15]. Inefficiency of this encoding stems from the fact that there are unnecessarily long distances (up to Nk) in diagonal directions. By encoding N bits at a time we are allowed to use whole Nk-dimensional space freely, and therefore more efficiently. We may tessellate it in such a way that each cell would contain inner point at least $\lceil \frac{k}{2} \rceil$ bits apart from its boundary. These inner points would serve as our codewords. Shannon's theorem in fact claims that by making M large enough it is possible to tessellate $\{0,1\}^M$ into so many codewords that the rate of the channel can be kept constant for any probability of error chosen beforehand.

Dependence of M on the probability of error was clarified by Gallager in 1968 [21]. He showed that the probability of block error depends on the block length M exponentially, namely that if we fix desired code rate R to certain number, less than the capacity C, then there exists a code with block length M, such that the probability of its block error p_B is

$$e^{-ME_s} \le p_B \le e^{-ME_r} \qquad \text{leading to} \qquad M \approx \frac{1}{E} \log \frac{1}{p_B} \qquad (108)$$

where E_s and E_r are constants depending only on the channel and selected rate R. As R approaches C, both constants go to zero, which makes it necessary to take M larger and larger to keep the probability of error constant. Moreover, for fixed channel, both E_s and E_r are decreasing, positive convex functions of code's rate R for $R \in [0, C)$ and $E_r(C) = 0$. This result shows that, in agreement with commonsense wisdom concerning *'get something for free'*-offers, we have to pay for reliability after all. Only not with a transmission speed but with a transmission delay (packet length).

3.3.1 Definition *Discrete Channel*

Discrete channel is a triple $\langle \mathbb{X},\, p(y|x),\, \mathbb{Y} \rangle$, where $\mathbb{X}$ and $\mathbb{Y}$ are finite sets and conditional probability $p(y|x)$ is non-negative function $\mathbb{X} \times \mathbb{Y} \to \mathbb{R}$ such that $p(\bullet|x) = 1$ for each $x \in \mathbb{X}$. It can be *instantiated* in probability space $\langle \Omega, \Pr_\Omega \rangle$ by two random variables $X : \Omega \to \mathbb{X}$ and $Y : \Omega \to \mathbb{Y}$ which satisfy $\Pr_\Omega(\text{“}Y = y\text{”} \mid \text{“}X = x\text{”}) = p(y|x)$ for all x, y for which $\Pr_\Omega(\text{“}X = x\text{”}) \neq 0$. X represents input and Y represents output.

3.3.2 Definition *n-th Memoryless Extension of Discrete Channel*

For channel $\langle \mathbb{X},\, p(y|x),\, \mathbb{Y} \rangle$, its *n-th memoryless extension* is channel $\langle \mathbb{X}^n,\, q(\vec{y}\,|\vec{x}),\, \mathbb{Y}^n \rangle$, where q is defined as follows.

$$q(\vec{y} \mid \vec{x}) := \prod_{k=0}^{n-1} p(y_k \mid x_k) \qquad (109)$$

[15] Minimal number of bit-flips by which one codeword can be transformed into another is called the *Hamming distance* of the two codewords.

It represents n independent uses of the original channel. Often the original channel is said to be memoryless if we consider only its memoryless extensions[16].

3.3.3 Notation The previous definition gives rise to n copies of same random variable $X : \Omega \to \mathbb{X}$. The new variables $X_1, \ldots, X_n$ act on Ω^n and are defined in an obvious way such that X_k is a copy of X on k-th Ω-coordinate of Ω^n. These variables occur quite frequently in statistics under a name *independent identically distributed (i.i.d.) random variables*. There is a common misuse of notation in this case. By X^n we do not mean set theoretical $(\Omega \times \mathbb{X})^n$ not even a function returning $X(X(\cdots X(\omega)))$ for $\omega \in \Omega$ but a function $\Omega^n \to \mathbb{X}^n$ such that $X^n(\langle \omega_1 \ldots \omega_n \rangle) := \langle X_1(\omega_1), \ldots X_n(\omega_n) \rangle$. Also, if there is a set $\mathbb{Z}$ having a one-to-one correspondence with $\mathbb{X}^n$ we sometimes identify
50 ◁ their elements without prior notice — this was used in (92). These formal lapses are common and in fact welcome as they help to keep the text readable, if used wisely.

3.3.4 Definition *Capacity of the Channel*

The *capacity* of $\langle \mathbb{X},\, p(y|x),\, \mathbb{Y} \rangle$ channel is defined the following way:

$$C := \max_{\Pr_\Omega(X)} I_\Omega(X;Y) \tag{110}$$

Where $\Omega := \mathbb{X} \times \mathbb{Y}$ and random variables X, Y are projections to their respective ranges $\mathbb{X}, \mathbb{Y}$. The maximization is meant to search thru any distribution $\Pr_\Omega$ which instantiates the channel in $\langle \Omega, \Pr_\Omega \rangle$. This is equivalent with searching over all input distributions $p(x)$, keeping $p(x,y) = p(y|x)p(x)$. Since all distributions on finite alphabet $\mathbb{X}$ form a high-dimensional simplex (that is a bounded and closed set) and mutual information is continuous function of $p(x,y)$, the maximum indeed exists (but may not be unique).

3.3.5 Note We can think about (110) as of shaping the source distribution such that it would have maximal influence on channel's output (this is somewhat similar with impedance matching commonly used in electronics, where the impedance of source and load have to be equal to deliver maximal power to the load).

3.3.6 Note For a channel with capacity C, its n-th memoryless extension will have a capacity of nC, because *no memory* means that individual uses are independent, hence $I(X^n;Y^n) = nI(X;Y)$.

3.3.7 Definition *Block Code, its Errors and its Rate*

For discrete memoryless channel $\langle \mathbb{X}, p(y|x), \mathbb{Y} \rangle$ we define a *block code of type* (Q, M) as a mapping $\mathcal{C} : \{1, 2, \ldots, Q\} \to \mathbb{X}^M$ and associated decoding function $\mathcal{D} : \mathbb{Y}^M \to \{1, 2, \ldots, Q\}$, which estimates what was sent thru the channel. The set $\mathrm{Rng}(\mathcal{C})$ is called a *codebook*. Only these sequences can be actually sent into the channel. So called *block error* occurs whenever $\mathcal{D}(\vec{y}) \neq \vec{x}$, where $\vec{x}$ was sent and $\vec{y}$ received. Probability of block error on input $\vec{x}$, in a probability space that instantiates the channel is:

$$p_B(\vec{x}) := \Pr_\Omega(\text{“}\mathcal{D}(Y^n) \neq \vec{x}\text{”} \mid \text{“}X^n = \vec{x}\text{”}) \tag{111}$$

[16] This can be done if physics of the channel guarantees that consecutive symbols remain separated in time during transmission, that is if the channel has only negligible echoes, wave dispersion, etc.

We are more interested in *maximal probability of block error*, defined as follows.

$$p_B := \max_{\vec{x} \in \mathrm{Rng}(\mathcal{C})} p_B(\vec{x}) \tag{112}$$

The *rate of the code* is defined as $R := M^{-1} \log_2 Q$. Note that it might not be integer, whereas Q is always an integer. It measures how many input bits (on average) are conveyed by single channel symbol.

3.3.8 Theorem *(Channel Coding Theorem: Reliable Communication — Shannon) For discrete memoryless channel with capacity C and for any $\varepsilon > 0$ and any $R \in (0, C)$ there exist block size M and appropriate (Q, M) block code with rate R and probability of block error $p_B \leq \varepsilon$.*

Proof Only an idea will be sketched here. Lookup real proof in [59] or [21]. As it is very hard to construct a code satisfying the theorem, in fact for most channels, explicit code is unknown, the proof calculates average error over *all* possible codebooks, showing that it is small. Then, there *must* exist at least one codebook for which the error is also small.

First, we fix $p(x)$. We can do this because for long enough block lengths M, it is under our control. That is because we can design a codebook $\vec{x}_1, \vec{x}_2 \ldots \vec{x}_Q$, whose codes would mimic chosen distribution. We can do it by selecting codes' components $x_{ij} \in \mathbb{X}$ randomly, in agreement with distribution $p(x)$. The encoder $\mathcal{C}$ works by selecting appropriate codeword from this codebook, that is $\mathcal{C}(a) := \vec{x}_a$, for which reason I will not make distinction between the encoder and the codebook in the sequel. As the codewords were generated by independent random tosses, each codeword has a probability: $\Pr_\Omega(\vec{x}_k) = \prod_{l=0}^{M-1} p(x_{kl})$ and the probability of using particular encoder is:

$$\Pr(\mathcal{C}) = \prod_{k=1}^{Q} \prod_{l=0}^{M-1} p(x_{kl}) \tag{113}$$

Now it is possible to develop a decoding strategy $\mathcal{D}$ based on so called *jointly typical sequences*. I will skip this step, only using its results — see [59] for technical details. So we have an input number a, which the encoder changes into $\vec{x}$ that gets randomly transformed by the channel into $\vec{y}$ from which the decoder estimates what was sent ($\widehat{a}$). This can be symbolically depicted in the following diagram.

$$a \xrightarrow{\mathcal{C}} \vec{x} \xrightarrow{p(y|x)} \vec{y} \xrightarrow{\mathcal{D}} \widehat{a} \tag{114}$$

We can compute probability of error for randomly selected codebook $\mathcal{C}$ and (uniformly) randomly selected message a to be

$$\pi_B := \sum_{\mathcal{C}} \sum_{a=1}^{Q} \Pr(\widehat{a} \neq a \mid \mathcal{C}, a) \frac{\Pr(\mathcal{C})}{Q} = \sum_{\mathcal{C}} \Pr(\mathcal{C}) \Pr(\widehat{a} \neq 1 \mid \mathcal{C}, 1) \tag{115}$$

Where the last equality comes from symmetry caused by averaging over all codebooks. Analyzing the decoder $\mathcal{D}$ based on jointly typical set decoding rule, it is possible to show that for any $\delta > 0$ there exists M_0 s.t. for any $M \geq M_0$:

$$\pi_B < \delta + Q\, 2^{-MI(X;Y)+3\delta M} = \delta + \underbrace{2^{-M(I(X;Y)-R-3\delta)}}_{\star} \tag{116}$$

We can control R by choosing Q. Hence, for $R < I(X;Y) - 3\delta$ we can always make M large enough to bound $\star$ by δ, thus getting $\pi_B < 2\delta$. As π_B is a weighted average of average unreliability of the respective codebooks, there must[17] be at least one codebook $\mathcal{C}_0$ whose average error rate is at most π_B. By tampering with $p(x)$ it is possible to maximize $I(X;Y)$, obtaining C instead of it. Now, we still have a code whose *maximal* probability of error may be enormous. But since $\sum_{a=1}^{Q} \Pr(\widehat{a} \neq a \mid \mathcal{C}_0, a) < 2\delta Q$, at least half of the codewords are already good, having probability of error less than 4δ — if it was not so, the sum would exceed $2\delta Q$. By sieving-out bad codewords we obtain codebook with slightly worse rate (namely $R - 1/M$), whose worst probability of block error p_B is less than $\varepsilon := 4\delta$. Q.E.D.

The above theorem justifies mutual information as a limit on signaling rate by showing that reliable communication is possible up to this limit. Unfortunately it does not provide practical way to construct the codebook. The enumeration method used in the proof is practically infeasible due to enormous spaces involved. Even the resulting codebook would be too large (exponential in M) to be of any use. Therefore, we need the code not only close to the channel capacity but also simple, namely we would like the encoder and decoder to be short and fast (preferably $\mathcal{O}(M)$) algorithms. It was only recently (1993) when practical codes (so called *Turbo Codes*) approaching the channel capacity for variety of channels were discovered [21].

57 ◁ **3.3.9 Example** Let us calculate capacity of symmetric binary channel (of fig. 11). Having $p(\mathtt{0}|\mathtt{1}) = p(\mathtt{1}|\mathtt{0}) = p = q$ and $p(\mathtt{0}|\mathtt{0}) = p(\mathtt{1}|\mathtt{1}) = 1-p$, mutual information $I(X;Y)$ is

$$H(X) - H(X|Y) = H(X) - \sum_y p(y) H(X|Y=y) = H(X) - p(\bullet) \underbrace{H(X|Y=\mathtt{0})}_{H(p)} \quad (117)$$

Where the last equality follows from symmetry. By setting $p(x) = 1/2$ we obtain $C = 1 - H(p)$.

3.3.10 Example For general binary channel with $p \neq q$ it is possible to obtain the following formula for its capacity, after modest calculus exercise:

$$C = H(p)\frac{q - \frac{1}{1+2^{-\alpha}}}{1-p-q} + H(q)\frac{p - \frac{1}{1+2^{\alpha}}}{1-p-q} + H\left(\frac{1}{1+2^{\alpha}}\right) \quad \text{where} \quad \alpha = \frac{H(q) - H(p)}{1-p-q} \quad (118)$$

As can be seen, the capacity seems to be rather complicated[18] even for this elementary channel. In fact, this is typical and exact analytic formula is known only for very few channels. As (118) is quite long it is often more practical to use the following lower bound:

$$C_L = 1 - H\left(\frac{p+q}{2}\right) \quad (119)$$

[17] This argument works even for weighted average: For $\lambda_i \geq 0$ and $\lambda_\bullet = 1$ there is always k s.t. $x_k \leq \sum_i x_i \lambda_i$. If it was not so we would have $x_k > \sum_i x_i \lambda_i$ for all k and could average it one more time, getting $\sum_k x_k \lambda_k > \lambda_\bullet \sum_i x_i \lambda_i$, which would be a contradiction.

[18] Yes, it is indeed symmetric in p and q since $\frac{1}{1+2^{-\alpha}} + \frac{1}{1+2^{\alpha}} = 1$.

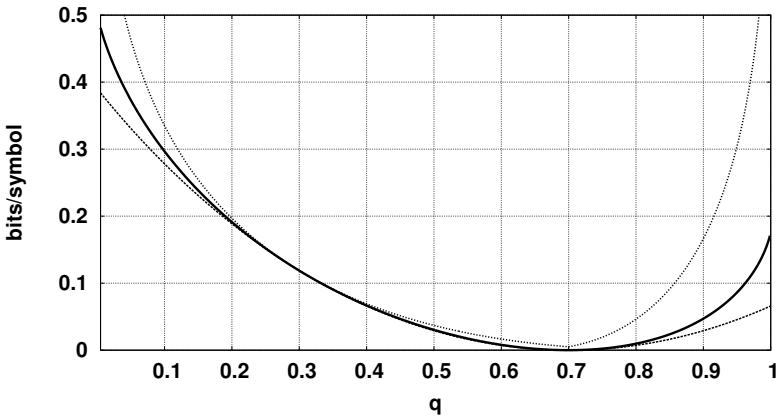


Fig. 12. *Capacity of non-symmetric binary channel $C(p,q)$ (solid line), drawn with its upper and lower bounds C_U and C_L (dotted lines) for fixed probability $p = 0.3$.*

This lower bound can be justified by operating the channel in a way which makes it effectively symmetric. To the original channel $X \to Y$ with crossover probabilities $p(y|x)$, there would be connected a scrambler (performing[19] $X := A \oplus Z$) and unscrambler doing $B := Y \oplus Z$, where $Z : \Omega \to \{0,1\}$ would be random variable independent of input A (i.e. $Z \perp\!\!\!\perp A$) such that $\Pr_\Omega(Z = \mathtt{0}) = 1/2$. Also, Z must not interfere with the working of the channel, that is $(Y \times X) \perp\!\!\!\perp Z$. This way, we would obtain new channel $A \to B$ with crossover probabilities

$$r(b \mid a) = \sum_{x,y,z} \Pr(X=x, Y=y, Z=z, B=b \mid A=a) = \sum_z \Pr(Y = b \oplus z \mid X = a \oplus z, Z = z, A = a) \Pr(Z = z) = \frac{1}{2} \sum_z p(b \oplus z \mid a \oplus z) \quad (120)$$

whose capacity is C_L of (119), which follows from 3.3.9, since $\frac{1}{2}(p+q) = r(\mathtt{1}|\mathtt{0}) = r(\mathtt{0}|\mathtt{1})$. It remains to show what seems intuitively clear — that we cannot push more information thru the pipeline than the capacity of its middle section allows, that is $I(A;B) \leq I(X;Y)$. Realizing that $(Y \times X) \perp\!\!\!\perp Z$ implies $A \perp\!\!\!\perp B \mid Y$, we have Markov chain $A \to Y \to B$ and the data processing inequality gives $I(A;B) \leq I(A;Y)$. Similarly we have $A \perp\!\!\!\perp Y \mid X$, i.e. $A \to X \to Y$, leading to $I(A;Y) \leq I(X;Y)$. Therefore $C_L = \max_{\Pr(A)} I(A;B) \leq \max_{\Pr(X)} I(X;Y) = C$. Q.E.D.

The capacity is shown in fig. 12 along with its lower and upper bound. However, the upper bound computed according to the following formula is only my conjecture.

$$C_U = 1 - H\left((p+q<1)\,?\,\sqrt{pq} : \sqrt{1-p}\sqrt{1-q}\right) \quad (121)$$

[19] $\bigoplus$ denotes the `xor` operation (addition modulo 2).

I have no proof for it although I have checked it numerically on a grid with step size of 10^{-5}. Perhaps AM-GM inequality[20] might be helpful in the proof, as it implies that $C_U \geq C_L$.

3.3.11 Theorem *(Fano's Inequality) Let random variables $A \to B \xrightarrow{\mathcal{D}} \widehat{A}$ form a Markov chain, where $\mathcal{D}$ is a decoding function (deterministic or random) which tries to recover value of A from the observation B. Moreover, let $\mathbb{A} := \mathrm{Rng}(A)$ be finite. Let P_e denote the probability of recognition error, that is, $P_e := \Pr_\Omega(A \neq \mathcal{D}(B))$. Then*

$$P_e \geq \frac{H(A) - I(A;B) - H(P_e)}{\log_2 \#\mathbb{A}} \geq \frac{H(A \mid B) - 1}{\log_2 \#\mathbb{A}} \tag{123}$$

Proof Let us define random variable $C : \Omega \to \mathbb{A} \cup \{\perp\}$ which would correct the errors made by the decoder in the following way (new symbol $\perp \notin \mathbb{A}$ means 'no correction').

$$C(\omega) := \begin{cases} \perp & \text{iff } \widehat{A}(\omega) = A(\omega) \\ A(w) & \text{else} \end{cases} \tag{124}$$

Another random variable $E : \Omega \to \{0,1\}$ would serve as an error indicator. Obviously $E(\omega) = (C(\omega) = \perp)\ ?\ 0 : 1$. Note that $P_e = \Pr_\Omega(\text{“}E = 1\text{”})$. As C allows for perfect
54 ◁ recovery of A from $\widehat{A}$ we have that $H(\widehat{A}, A) = H(\widehat{A}, C)$. It follows from 3.2.20 that $H(\widehat{A}, C) \leq H(\widehat{A}) + H(C)$, hence $H(A|\widehat{A}) \leq H(C)$. From data processing inequality we have that $H(A) - I(A;B) = H(A|B) \leq H(A|\widehat{A}) = H(A) - I(A;\widehat{A})$, hence

$$H(A|B) \leq H(C) \tag{125}$$

This is not surprising as it means that in order to correct $\widehat{A}$, we have to add at least the information about A that was missing in B. $H(C) = H(C,E)$ since all information about E is already in C. Now, $H(C,E) = H(E) + H(C|E) = H(E) + H(C|\text{“}E=1\text{”})P_e + H(C|\text{“}E=0\text{”})(1-P_e)$. Moreover, $H(C|\text{“}E=0\text{”}) = 0$ and $H(C|\text{“}E=1\text{”}) < \log_2 \#A$ which leads to $H(A|B) \leq H(E) + P_e \log_2 \#A \leq 1 + P_e \log_2 \#A$. Q.E.D.

3.3.12 Theorem *(Channel Coding Theorem: Unreliable Communication) It is possible to communicate at rate*

$$R(p_b) = \frac{C}{1 - H(p_b)} = \frac{C}{1 + p_b \log_2 p_b + (1-p_b)\log_2(1-p_b)} \tag{126}$$

thru discrete memoryless channel with probability of bit error[21] *less than $p_b + \varepsilon$ for any $\varepsilon > 0$, which determines the block length M.*

Proof See [21] or try yourself. All required machinery dwells in the theorems proved so far. Q.E.D.

[20] So called AM-GM inequality states that for $\alpha_i \geq 0$ and $S := \alpha_\bullet$ the following holds.

$$\frac{1}{S}\sum_i \alpha_i x_i \geq \left(\prod_i x_i^{\alpha_i}\right)^{1/S} \tag{122}$$

43 ◁ It follows from $\log \sum_i (\alpha_i/S) x_i \geq \sum_i (\alpha_i/S) \log x_i$, which is an instance of Jensen's inequality 3.1.18.

[21] The input number a is treated as if it was written in binary representation and p_b is probability that its randomly selected bit will not be decoded correctly. In case of communication below channel's
60 ◁ capacity 3.3.8 we have $p_B \leq p_b \leq M p_B$.

3.3.13 Theorem *(Channel Coding Theorem: Impossible Communication) For any $p_b \geq 0$ communication at rate higher than $C/(1-H(p_b))$ is not possible.*

Proof The encoder, channel and the decoder form a Markov chain (114). From data ▷ 60
processing inequality 3.2.31, we have (M is code's block length): ▷ 55

$$MC \geq I(X^M; Y^M) \geq I(A; Y^M) \geq I(A; \widehat{A}) \tag{127}$$

On the other hand, being able to send data at rate R with probability of bit error p_b means that $MR(1 - H(p_b)) \leq MRC_B \leq I(A;\widehat{A})$ because we could equivalently send MR bits thru a noise-less channel and then send each received bit thru a binary channel with crossover probabilities p and q such that $p + q = 2p_b$. According to (119) the capacity of this thought binary channel C_B is bounded by $1 - H(p_b)$ from below. Therefore, at least $MR(1-H(p_b))$ bits get thru. It may be more for correlated bit errors. Anyway, $MR(1-H(p_b))$ constitutes a lower bound on $I(A;\widehat{A})$. As $R > C/(1-H(p_b))$ we have $I(A;\widehat{A}) > MC$, which is in contradiction with (127). Q.E.D.

Channel coding theorems can be further generalized, especially, the channel may have memory and it may be continuous valued (with noise, making its capacity finite). In all these cases we can communicate arbitrarily reliably up to a certain rate. Moreover, the codewords tend to look as noise, which stems from their random construction.

When transmitting data that came from a source of entropy lower than maximal we can first compress them (by arithmetic coding, for instance), which can be done regardless of properties of the channel and then send them to the other side using our best channel code, which in turn does not depend on probability distribution of the source. Famous *source/channel separation theorem* states that this procedure is optimal in a sense that any encoder (including those that would be optimized specifically for this source/channel pair) would not work any better (than by arbitrarily small $\varepsilon > 0$). This theorem is a real blessing for engineers who like to develop highly optimized black-boxes, connecting them together later.

How different all of this is from human language, which by no means looks as random noise! It obeys Zipf's law at many levels[22] and its complicated multilevel redundancy protects it against transmission errors. Some might suspect that we, the humans, are inferior in this respect but there may be a good reason why we do not communicate by noise bursts — the learning. From noise-like signals which hardly ever repeat exactly (owing to the source compression), showing no apparent inner structure, it would be quite hard to learn the language without prior knowledge of the code. On the other hand, we can start learning natural language before we can fully decode (and understand) its messages. As it is open at so many levels we can start with statistical properties of syllables then combine them to words, use words to discover even larger structures, etc.

[22] So called Mandelbrot-Zipf's law states that words in a lexicon, sorted according to their probability of occurrence in the text obey the following probability distribution

$$\Pr(w_k) = a(b+k)^{-c} \tag{128}$$

where $a > 0,\ b \geq 0,\ c > 1$ are suitable constants and k is rank of the word (starting by 1). This law holds on different scales simultaneously — for instance, syllables or longer phrases obey it too. This contrasts with compressed communication which tends to uniform distribution.

3.3.14 Viterbi Codes

The Viterbi Algorithm was originally developed in 1967, as a decoding strategy for *convolutional error correcting codes* [15]. It worths saying a few words about them so that we could better understand the background we are using for speech recognition. The idea will be demonstrated on the code that was actually used to send data from the Voyager space probes[23] down to the Earth.

The computer of the spacecraft emits messages encoded in binary alphabet. Let's assume that these have been already compressed and so the probability of `0` and `1` is close to 0.5 and adjacent zeroes and ones are nearly uncorrelated. The individual bits
62 ◁ will be denoted as x_n. The encoder computes[19]

$$a_n := 1 \oplus \bigoplus_{k=0}^{6} \alpha_k x_{n-k} \quad \text{and} \quad b_n := \bigoplus_{k=0}^{6} \beta_k x_{n-k} \tag{129}$$

where the sequences α and β are defined as follows, according to [11].

$$\alpha := \langle 1,0,1,1,0,1,1\rangle \quad \text{and} \quad \beta := \langle 1,1,1,1,0,0,1\rangle \tag{130}$$

The output stream $\langle a_0, b_0, a_1, b_1, \ldots\rangle$ is obtained by interleaving streams a and b. Hence for each input bit there will be two output bits. The operation (129) is called a *convolution*, hence the name convolutional codes. Formula (129) can be easily implemented in hardware, using a 7 bit shift register and several xor gates. The code is obviously redundant and it is this redundancy what allows to correct bit flipping errors occasionally occurring in radio transmission. This code is referred to as $(7, 1/2)$-code, where the first number means number of state bits in the encoder (so in our case the encoder has 2^7 distinct states) and the second number is rate of the code.

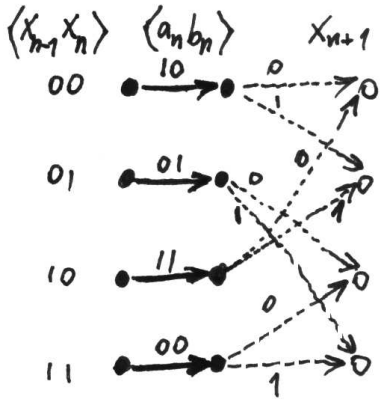


Fig. 13. *$(2, 1/2)$-code HMM. The outline circles on the right side should be considered to be identical with the filled circles on the left (imagine it drawn on a cylinder). Also note that null-arrows are superfluous as they could be composed with regular arrows. It would halve the number of states. Even though it is preferable in implementation it would hide the principle so I decided for more readable variant.*

The encoder can also be represented by a graph of fig. 13. For sake of simplicity it is drawn for $(2, 1/2)$-code with $\alpha = \langle 1, 0\rangle$ and $\beta = \langle 1, 1\rangle$. By walking along the arrows, writing out the labels above the solid arrows and using the labels above the dashed arrows to navigate, we obtain output that the Viterbi encoder would produce. This strongly resembles HMM and it comes at no surprise that the trellis of this graph can be used for decoding. We just have to modify the graph a little bit. First we have to delete labels above null arrows because these labels only designate the input which we treat as random anyway. We set the probability of the two null arrows emanating from each state to 0.5 as we consider the input to be already compressed.

[23] Launched in 1977 and still operational at time of this writing — Voyager 1 being about 15.34 light-hours from the Earth, Voyager 2 being 12.44 light-hours away from us [8].

If we tried to use our HMM for decoding as it is now, it would fail to decode messages with corrupted bits. This is because the solid-line arrows do not yet permit incorrect outputs, therefore the sequence passing thru correct HMM states might not exist for noise-corrupted bit strings. This is somewhat akin to the reasons why we needed to smooth probabilities in the language model. But here, the reason comes from the noise in the channel not from the fact that any word sequence is possible in principle, as was the case in human language[24]. So we have to introduce a noise model into the HMM.

It is educational to describe the Viterbi encoder in terms of fig. 1. There, $P(W)$ ▷ 15 would assign probability of 0.5 to individual bits of the input stream treating them as independent and $P(A \mid W)$ would correspond to the HMM of fig. 13 composed with a model of noise introduced by the channel. In the most simple case this would be a memoryless discrete channel of fig. 11. Since in the radio transmission which uses ▷ 57 phase modulation there is no fundamental difference between transmitting 0 and 1 we may regard it as symmetric channel with probability of bit flipping being equal to p. Then, it is easy to modify the HMM by adding parallel solid-line arrows for all other combinations the two output bits, setting their probability to $(1-p)^2$ for the correct symbol, to $p(1-p)$ in case of one bit error and to p^2 for symbols with two flipped bits.

If we set probability p to match that of the channel[25], we would obtain optimal[26] decoder by running a Viterbi algorithm on this modified HMM. Note that the leading bits of the message can be output only after the Viterbi has finished, which can happen only after the entire message (or at least a packet if we send the file divided into packets) has been received. This is because the winning path may change in the last stage of the trellis.

So we have an optimal decoder for a discrete channel. But we can go beyond this. The radio channel is in fact analogue and we may want to take advantage of it. Real noise can be closely approximated by a Gaussian distribution — sending the bits across this channel changes the mean value of the noise signal, leaving its variance unaffected. Probability density of receiving voltages $\langle x, y\rangle \in \mathbb{R}^2$ of the two signal samples, given that digital signal $\langle a, b\rangle \in \{0,1\}^2$ has been sent is then:

$$p_{ab}(x,y) = \frac{1}{\pi\sigma^2} \exp\left(-\frac{(x-a)^2 + (y-b)^2}{2\sigma^2}\right) \tag{131}$$

The Viterbi algorithm would use $-\log p_{ab}(x,y)$ as a cost of traversing that arrow, otherwise it would be unmodified.

This method of decoding really improves performance. According to [11], we can afford 2 dB worse signal to noise ratio to achieve same number of uncorrected bit errors

[24] Those idealists who think that certain word sequences are impossible due to language structure should consider that we would like our ASR to have a chance to succeed even when they utter their words of wisdom: *"Sequence 'all your base are belong to us' can never appear in an English sentence."*.

[25] This can be measured quite precisely, provided that it does not change in time.

[26] In the sense that it would minimize the probability of receiving a corrupted message. It does not, however, take the number of corrupted bits into an account so it might not be optimal if we would like minimize $\mathcal{E}$(*"the number of bad bits remaining in the decoded message"*) $= p_b M$.

as the Viterbi code with discrete channel model would have achieved. Moreover if we are satisfied with only 1.5 dB improvement, we can attain that even with discrete channel if we quantize each input bit into 8 (suitably chosen) levels. Note that there would be 64 arrows per code state in the graph of fig. 13 instead of 4 but this is of no concern because only one remains after the trellis has been pruned. Even storing this number of arrows is not a problem as they can be represented by a formula. The only thing that needs to be stored are 8 numbers representing the probability distribution (16 if the channel was not symmetric).

Another possible improvement concerns time varying noise. Imagine that we are using radio transmission during a thunderstorm. Each nearby discharge renders the bits just received useless. If only few bits were damaged the code will work as expected. But we could improve its performance by telling it which bits have been corrupted so that they would be more or less ignored during trellis decoding. This would stop diffusion of erroneous information down the trellis, thus improving the performance. In case of Gaussian noise model (131), this could be achieved by temporarily boosting the
14 ◁ variance σ during the discharge. In fact this is what (1) demands to attain minimal probability of incorrectly decoded message. It demands that the probability distribution of the channel should match the reality. If the reality has time dependent σ so should have our model. Strictly speaking, during the discharge the distribution may lose its Gaussianity and even symmetry due to possible saturation of input circuits. So it is quite possible to receive all zeroes during the discharge, but these are implementation issues, going well beyond the scope of this book.

The only trouble left is that we need to know precise instants of discharges. But this is not very complicated because they cause wideband radiofrequency noise, which can be detected by listening on several silent bands which are not used for communication. If we observe sudden increase of power in all these bands we almost surely have an interference in the data band as well, and should increase σ. The amount of how much we should do it could be calculated from the power we observed in auxiliary bands and calibration table we measured before, which would tells us how this power relates with noise power observed in the data band. It should be mentioned that rising σ longer than for 7 channel bit periods (in case of Voyager code) is ineffective because the code state, which makes the error correction possible, becomes lost. Of course, the interference may last for more than 7 bits if it is weaker in terms of σ because the state would be destroyed only partially in such a case.

I am not aware if this trick is actually being used in radio communications. As far as I know, it is not a standard technique. Also I am not sure if anything similar has ever been tried in speech recognition area. There are noise robust systems but those I have encountered were trained on a speech corrupted by (an automobile) noise of various intensity. During recognition, the actual model was interpolated from the trained models to match the noise intensity estimated from the signal. Although it is a step in a right direction it should be noted that noises are not so simple in case of speech recognition. First the 'noise' added by a channel involves not only background sounds but also reverberations. Secondly there are many types of background sounds, most of them being transient. While the aforementioned recognizer can work well in a car with steadily humming motor, it is likely to experience problems with non-stationary noises like door slams.

Before finishing this subsection let us return to the Voyager, for sake of completeness. Not all choices of α and β lead to equal protection against errors. It is hard to tell analytically how good certain code will be. In case of the Voyager program, the code length 7 was chosen so that the computers of that time would be able to decode it in reasonable time and α and β were found by an exhaustive search over all 2^{14} possible choices by simulating performance of the respective codes over the (simulated) Gaussian channel and selecting α and β achieving best performance.

The Viterbi code was not the only code used in the Voyager space probe. In fact it was the last code before the data were passed to the modulator and antenna. The Viterbi encoder was fed with the data that already passed thru the Solomon Reed encoder. The *Solomon-Reed Code* is designed to correct burst of errors[27]. It is there to correct the errors that the Viterbi decoder leaves uncorrected, as these tend to appear in clusters[28].

The final problem of receiving Voyager signals is proper synchronization. Fortunately, the bit rate is quite regular (and known to the receiver) so we only have to find middles of bits then determine which one is a_n and which one is b_n. This can be done by running 6 Viterbi decoders each one being shifted by 1/3 of bit duration from the preceding decoder. The one with the right position will experience highest probability of the winning path. This is done only at the signal beginning and it may be even enhanced by specially designed codewords sent as a signal's header. Once the correct synchronization is found it can be easily maintained by computing correlation with idealized two level signal (obtained from so far decoded bits) with the received noisy analogue signal. Position of the highest peak in the correlation function would then be used to adjust timing of the next sampling point, so that slow changes of the bit rate would be followed.

Unfortunately, the speech signal is far from being synchronously timed. Different people use different speaking rates and rate changes are commonly used to emotionally emphasize certain words, etc. It is remarkable that good codes (and decoding strategies) for channels which introduce timing errors in addition to ordinary signal noise are still an open research problem [21]. It turns out that using Viterbi decoding is suboptimal, regarding the timing. Note that a loop in an HMM can only describe exponential distribution of dwelling in that state. This causes a problem if we for instance need to model steady state of a vowel by a single state with a loop. Exponential distribution gives highest probability to the shortest realization of that vowel. But we intuitively feel that this is wrong as there would be certain 'typical' duration and shorter and longer vowels should be less probable than that.

Although it would be possible to model more complicated distributions using more states and more arrows (having dedicated paths instead of loops to model the distribution as we prescribed it) some of the advanced ASR systems took another direction, using explicit modeling of probability distributions of the time it takes to traverse the outputting transition.

[27] For which reason it is also used on compact discs, since scratches lead to error bursts.

[28] After the Viterbi decoder loses its state it is likely to output errors until it collects enough data to reduce entropy about the state in which the encoder might have been when it transmitted the current symbol. That is why the uncorrected errors stick together.

3.3.15 Other Uses of Viterbi Decoding

Interesting application of Viterbi Decoding is used in modern hard disk drives, known as PRML (*Partial Response – Maximum Likelihood*). Spatial density of bits is so high that individual bits influence each other's readout signal because the reading head slightly reacts to the magnetic field of the nearby bits, too.

This means that the analogue signal y as seen by the reading head can be written as a convolution of some weighting coefficients a_k (determined by head geometry) with magnetically recorded bits $x_0, x_1, x_2, \ldots$, where $x_n \in \{-1, 1\}$, in the following way:

$$y_n = \varepsilon_n + \sum_{k=-N}^{N} a_k x_{n-k} \tag{132}$$

65 ◁ where ε_n is additional thermal noise. This strongly resembles (129), only the operation 65 ◁ are carried out in $\mathbb{R}$ instead of $\mathbb{Z}_2$. But this is only a minor difference which still allows us to use the same decoding technique. Figure 13 is still valid HMM for the decoder if used with 2^{2N+1} code states and emission probabilities precomputed according to 66 ◁ (132) with Gaussian noise ε. Although continuous model similar to that of fig. (131) could be used in theory, the practical implementation will rather use discrete channel with multiple quantization levels[29]

In fact, not all $2N$+1-bit combinations are allowed on the disk because of physical limitations on maximal number of consecutive bits of same polarity and requirement of approximately zero DC level. To satisfy these needs and also to provide necessary synchronization bits, the data bits are intermixed with auxiliary bits. Usually *8 to 10 code* is used for that purpose, mapping each byte to a 10 bit sequence. This code is taken into account in the Viterbi decoder as it limits number of states and it also restricts transitions (most states have unequal probabilities of 0 and 1 arrows, some of them even having single output arrow). These restrictions actually help the decoder as they limit the number of high probable competing paths.

It should be noted that formula (132) could also describe propagation of sound 83 ◁ in a reverberant environment (see 4.3.1). If x_n were bits transmitted by an ideal loudspeaker in a room, then y_n would be what we would measure with perfectly linear microphone, provided that the room would not change its shape, nothing would move inside it (including the speaker and microphone) and the samples y_n would be perfectly synchronized with x_n. So, in principle, we could use the Viterbi decoder to invert (132). Although it requires exact synchronization and assumes that nothing moves, it is reasonable to expect that it will work to some extent even if these assumptions would be partially violated. This would explain why current ASR systems work acceptably even in somewhat reverberant environment, such as a room, despite the fact that typical 152 ◁ front end conceals only very short echos (10 ms) via the CMS (see 5.1.1).

If the room stays invariant during training then the acoustic model will simply learn it together with phonemes. Later, when the room would be different on testing,

[29] There has to be an A/D converter anyway, so in the end, we have nothing like real-valued voltage. Additionally, the number of quantization levels will likely be small in this case because the bit speed of modern discs is enormous and finer quantization would be cost-prohibitive. Finally, as noted in [11], 8 quantization levels are equivalent to 1.5 dB SNR improvement over 2 quantization levels, while whatever detailed quantization can only give additional 0.5 dB over these 8 quantization levels.

the performance of the system might drop considerably. If the training takes place in variety of rooms, a mixture of reverberant characteristics would be learned. It is then reasonable to expect that the recognition will work in nearly any room to some extent.

Unfortunately, the HMM for words of fig. 4, will be quite reluctant to learn any ▷ 21 echo-related features because the echo-relevant HMM state is lost on phone boundaries. That is why standard ASR systems (practically all commercial ones) use more compli- cated form of HMM than the one described in section 2.4. It is called *triphone model* ▷ 20 and is described throughly in [39]. The idea is that for each phoneme we maintain many of its variants conditioned on context (the preceding and following phoneme). The HMM of the phoneme is the same as we had before (fig. 3), only the tying is ▷ 21 changed and word beginnings and ends has to be split accordingly, so that the ap- propriate triphone could be used, based on the last phoneme of the preceding word. Needless to say, main problem with training such a beast is data sparseness. However it could be overcome with clever methods of clustering and smoothing. Current IBM rec- ognizer (ViaVoice) is already using septaphones, which means that each phoneme has a context made of 3 preceding and 3 succeeding phonemes. Triphones were originally introduced to cope with coarticulation[30]. The fact that septaphone system outper- forms a triphone one, such that it worths to use it in commercial recognizer despite the complications it brings, suggest that it perhaps captures echo as well as coarticulation.

3.4 My Contribution

My contribution consists of the idea of tracking instantaneous SNR during Viterbi de- coding as described in 3.3.14, and of my suspicion that septaphones partially counteract ▷ 65 the echo, as stated in 3.3.15.

[30] An effect when adjacent phonemes influence each others' sound, as explained in chapter 5. ▷ 147

4 Signal Processing Theory

In this chapter, I summarize some well known facts about digital signal processing. It does not contain many original results, although the way how I present the theory is original. Experts are strongly advised to skip this chapter entirely, and use it solely as reference of the notation.

Also, this chapter should not be expected to cover all the topics treated in standard textbooks on signal processing. Mostly just the things that will be needed in the sequel are included. Specifically, continuous-time signal theory will not be presented at all. Although the statements of its theorems are rather similar to their discrete-time counterparts, they are not exactly same, and more importantly they need more advanced calculus to be proven. Even to define what we mean by the continuous-time signal the theory of distributions would be needed. But, unfortunately, it is rather abstract and lengthy and, for our purposes, it definitely would not pay-off since we would need it only to describe continuous-time wave before it was sampled into the computer. To side-step continuous-time theory I will treat this single case by approximating continuous signal with discrete-time signal of very high (say 10^{50} Hz) sampling frequency. This should not make much physical difference because the air pressure is, after all, generated by the molecules impacting microphone's membrane, which is a discrete process, provided that the sampling frequency is so high, that it will not miss molecule impacts between consecutive measurements. Either there will be a hit (possibly from several molecules at once), or there will be nothing at all within whole sampling period[1]. This does not mean that the continuous-time description is useless. It is very convenient for analysis of networks of elements described by integral or differential equations, such as capacitors or inductors, in which case discrete-time method would be cumbersome. But as we neglect these elements in the analog path and the description of reverberation we gets by with delays and sums, inclusion of this theory would be an overkill.

Main topics studied here will be how to sample the physical signal at reasonably slow rate such that it would be possible to reconstruct the original signal with good enough precision, how the signal changes if it is passed thru a linear system (such as a room causing a reverb) and what is signal's spectrum and why it is important. Also, several special topics such as the Hilbert transform will be discussed, as they will be used in the front end.

4.1 Discrete Periodic Signals

Before we define general signal, let us first explore easier case of periodic signals. As they are periodic and discrete we only need finitely many values to represent them — one period is enough to specify whole signal going from minus infinity to infinity

[1] According to quantum physics, time intervals shorter than Planck's time ($\approx 5.391 \cdot 10^{-44}$ s) should be indistinguishable. If the molecule could bounce off the membrane in shorter time it would violate this principle (by passing thru two distinguishable places in space during Planck's time we would be able to distinguish even shorter time interval, which is considered impossible).

in time. Therefore we can treat them as N-dimensional vectors over $\mathbb{C}$, where N is determined by the signal's period. To avoid confusion with later defined signals, I will refer to discrete periodic signals as to vectors, from now on.

The reason why $\mathbb{C}$ was chosen instead of $\mathbb{R}$ as the field over which we develop the theory is rather practical. Many useful operations with signals turn out easier to state in $\mathbb{C}$. Nevertheless, the input signal will be real valued in most applications of the theory.

Elements of the vector will be indexed from 0 to $N-1$, where N is the vector's dimension. Matrices will also be indexed starting by 0. The vectors are understood as column vectors and formally behave as $1 \times N$ matrices. So when I write Ax, I mean the matrix A multiplied by a column vector x. Identity matrix will be denoted by I (sometimes by I_N to explicitly state that it is $N \times N$ matrix), all-zero matrix will be denoted by O. The complex conjugate of matrix A will be written as $\overline{A}$. Transposition as A^{T} and Hermite conjugate as $A^{\mathrm{H}} = \overline{A}^{\mathrm{T}}$. As I treat vectors as special case of matrices, the same symbolics holds for them, too. Using this notation, the (complex) dot product of x and y would be written as $x^{\mathrm{H}}y$. Vector norm will be written as $||x||$, being defined this way: $||x|| := \sqrt{x^{\mathrm{H}}x}$. Individual matrix elements are denoted using indices, for instance A_{ij} refers to the element in i-th row and j-th column of A.

4.1.1 Definition *Discrete Fourier Transform*

For N-dimensional vector a, its discrete Fourier transform is defined as $F_N \cdot a$, where $N \times N$ matrix F_N is defined as follows:

$$(F_N)_{yx} := \frac{1}{\sqrt{N}} e^{-\frac{2\pi i y x}{N}} \tag{133}$$

If there is no danger of confusion we will write just F instead of F_N, when referring to that matrix.

4.1.2 Observation *F_N is unitary matrix*

Proof We have to show that $F_N F_N^{\mathrm{H}} = I = F_N^{\mathrm{H}} F_N$. Expanding the first product, we get

$$\frac{1}{N} \sum_{p=0}^{N-1} e^{-\frac{2\pi i y p}{N}} e^{\frac{2\pi i p x}{N}} = \frac{1}{N} \sum_{p=0}^{N-1} e^{\frac{2\pi i p}{N}(x-y)} \tag{134}$$

Now, if $x - y = kN$ for integer k the argument of the exponential will be 0 therefore the sum will be N and the result will be 1. Because x and y are both ≥ 0 and less than N, this only happens for $x = y$. In other cases it is easy to show (geometric sum) that the sum is zero. So we have that $F_N F_N^{\mathrm{H}} = I$. The other product follows from the first one and the fact that F_N is symmetric ($F_N = F_N^{\mathrm{T}}$). So the first product is in fact $I = F_N \overline{F_N}$. And the second one is $I = \overline{I} = \overline{F_N \overline{F_N}} = \overline{F_N} F_N$. Q.E.D.

4.1.3 Consequence *(Parseval's equality)* $||x||^2 = ||F_N x||^2$

Proof $||F_N x||^2 = (F_N x)^{\mathrm{H}}(F_N x) = x^{\mathrm{H}} F_N^{\mathrm{H}} F_N x = x^{\mathrm{H}} x = ||x||^2$ Q.E.D.

4.1.4 Lemma *(Cyclic Correlation Lemma)*

$$\sum_{k=0}^{N-1} a_k b_{(n+k)\%N} = \frac{1}{N}\sum_{f=0}^{N-1}\left(\sum_{k=0}^{N-1} a_k e^{-\frac{2\pi i f k}{N}} \sum_{k=0}^{N-1} b_k e^{\frac{2\pi i f k}{N}}\right) e^{-\frac{2\pi i f n}{N}} \tag{135}$$

Proof

$$c_n = \sum_{k=0}^{N-1} a_k b_{(k+n)\%N} \tag{136}$$

$$\sum_{n=0}^{N-1} c_n e^{\frac{2\pi i f n}{N}} = \sum_{n=0}^{N-1}\sum_{k=0}^{N-1} a_k b_{(k+n)\%N} e^{-\frac{2\pi i f k}{N}} e^{\frac{2\pi i f (k+n)}{N}} = \underbrace{\sum_{k=0}^{N-1} a_k e^{-\frac{2\pi i f k}{N}} \sum_{k=0}^{N-1} b_k e^{\frac{2\pi i f k}{N}}}_{h_f} \tag{137}$$

Now, applying Fourier transform to both sides and using observation 4.1.2 we get:

$$\sqrt{N} c_p = \frac{1}{\sqrt{N}} \sum_{f=0}^{N-1} h_f e^{-\frac{2\pi i}{N} f p} \tag{138}$$

Q.E.D.

4.1.5 Definition *Cyclic Convolution*

For N-dimensional vectors a and b, the cyclic convolution $a * b$ (which is also N-dimensional vector) is defined as follows[2]:

$$(a * b)_n := \sum_{k=0}^{N-1} a_{(n-k)\%N} b_k \tag{139}$$

4.1.6 Lemma *(Cyclic Convolution Lemma)*

$$\sum_{k=0}^{N-1} a_k b_{(n-k)\%N} = \frac{1}{N}\sum_{f=0}^{N-1}\left(\sum_{k=0}^{N-1} a_k e^{\frac{2\pi i f k}{N}} \sum_{k=0}^{N-1} b_k e^{\frac{2\pi i f k}{N}}\right) e^{-\frac{2\pi i f n}{N}} \tag{140}$$

Proof

$$c_n = \sum_{k=0}^{N-1} a_k b_{(n-k)\%N} \tag{141}$$

$$\sum_{n=0}^{N-1} c_n e^{\frac{2\pi i f n}{N}} = \sum_{n=0}^{N-1}\sum_{k=0}^{N-1} a_k b_{(n-k)\%N} e^{\frac{2\pi i f (n-k)}{N}} e^{\frac{2\pi i f k}{N}} = \sum_{k=0}^{N-1} a_k e^{\frac{2\pi i f k}{N}} \sum_{k=0}^{N-1} b_k e^{\frac{2\pi i f k}{N}}$$

The rest follows by use of the discrete Fourier transform. Q.E.D.

[2] % stands for the modulo operation defined as follows: $a\%b := a - b\cdot\lfloor a/b\rfloor$.

4.1.7 Note It follows from the above lemma that cyclic convolution is commutative, associative and distributive. This is because the lemma translates convolution into point wise multiplication which has these properties. For sake of clarity, I will show it for associativity. The above lemma states that $a * b = \sqrt{N} F(F^{\mathrm{H}} a \odot F^{\mathrm{H}} b)$, where $\odot$ denotes element-by-element multiplication of the vectors[3]. Now, let us have three N-dimensional vectors, a, b and c. Then

$$a*(b*c) = \sqrt{N}\cdot F(F^{\mathrm{H}} a \odot F^{\mathrm{H}}(\sqrt{N}\cdot F(F^{\mathrm{H}} b \odot F^{\mathrm{H}} c))) = N\cdot F(F^{\mathrm{H}} a \odot (F^{\mathrm{H}} b \odot F^{\mathrm{H}} c)) \tag{142}$$

Because of linearity of F and distributivity of $\odot$-multiplication, we get

$$a*(b*c) = N\cdot F((F^{\mathrm{H}} a \odot F^{\mathrm{H}} b)\odot F^{\mathrm{H}} c) = N\cdot F(F^{\mathrm{H}}(F(F^{\mathrm{H}} a \odot F^{\mathrm{H}} b))\odot F^{\mathrm{H}} c) = (a*b)*c \tag{143}$$

Q.E.D.

Often, we want to compute 'acyclic convolution', and the following lemma tells us how much zero padding is needed so that we could harness our cyclic convolution machinery.

4.1.8 Lemma *(Convolution Lemma) Let a_k be zero for $k \notin [0, M_a]$ and b_k be zero for $k \notin [0, M_b]$. Then for any $N > M_a + M_b$:*

$$\sum_{k=0}^{n} a_k b_{n-k} = \frac{1}{N}\sum_{f=0}^{N-1}\left(\sum_{k=0}^{N-1} a_k e^{\frac{2\pi i f k}{N}} \sum_{k=0}^{N-1} b_k e^{\frac{2\pi i f k}{N}}\right) e^{-\frac{2\pi i f n}{N}} \tag{144}$$

Proof Since $N > M_a + M_b$, we have $\sum_{k=0}^{n} a_k b_{n-k} = \sum_{k=0}^{N-1} a_k b_{(n-k)\%N}$, and the rest follows directly from the cyclic convolution lemma. Q.E.D.

4.1.9 Fast Fourier Transform (FFT)

The above lemma can be used to rapidly compute convolution using the Fast Fourier Transform, which is a fast algorithm for computing the discrete Fourier transform. It can be concisely written as appropriate factorization of the F_N matrix. Let N be power of 2, and let the $N \times N$ matrix P and $\frac{N}{2} \times \frac{N}{2}$ matrix Q be defined as follows:

$$P := \begin{pmatrix} 1 & 0 & & & & & & \\ & & 1 & 0 & & & & \\ & & & & 1 & 0 & & \\ & & & & & & \ddots & \\ 0 & 1 & & & & & & \\ & & 0 & 1 & & & & \\ & & & & 0 & 1 & & \\ & & & & & & \ddots & \end{pmatrix}, \quad Q := \begin{pmatrix} 1 & & & & \\ & e^{-2\frac{\pi i}{N}} & & & \\ & & e^{-4\frac{\pi i}{N}} & & \\ & & & e^{-6\frac{\pi i}{N}} & \\ & & & & \ddots \end{pmatrix} \tag{145}$$

Px permutes input vector x such that its odd and even indexed elements become grouped together. It is obvious that $P^{-1} = P^{\mathrm{T}}$. Now,

$$F_N = F_N \cdot P^{\mathrm{T}} \cdot P = \frac{1}{\sqrt{N}}\begin{pmatrix} \left(e^{-\frac{2\pi i}{N} y 2x}\right)_{yx} & \left(e^{-\frac{2\pi i}{N} y(2x+1)}\right)_{yx} \end{pmatrix} P = \tag{146}$$

[3] In the following, $\odot$ will be considered to have lower priority than matrix multiplication and higher priority than addition. So $a + b \odot C \cdot d$ should be read as $a + (b \odot (C \cdot d))$

$$= \frac{1}{\sqrt{N}} \begin{pmatrix} \sqrt{\frac{N}{2}} F_{\frac{N}{2}} & \sqrt{\frac{N}{2}} Q F_{\frac{N}{2}} \\ \sqrt{\frac{N}{2}} F_{\frac{N}{2}} & -\sqrt{\frac{N}{2}} Q F_{\frac{N}{2}} \end{pmatrix} P = \frac{1}{\sqrt{2}} \begin{pmatrix} I & I \\ I & -I \end{pmatrix} \begin{pmatrix} I & \\ & Q \end{pmatrix} \begin{pmatrix} F_{\frac{N}{2}} & \\ & F_{\frac{N}{2}} \end{pmatrix} P \quad (147)$$

so we have expressed F_N by means of two invocations of $F_{\frac{N}{2}}$, $N/2$ complex multiplications, N complex additions and some data reordering. We can continue recursively and use (147) on $F_{\frac{N}{2}}$, $F_{\frac{N}{4}}$ until we reach $F_1 = I_1$. This way we get rid of Fs in the formula and only multiplications by Q and additions survive. As there were $\log_2 N$ recursion steps, the overall complexity is $\mathcal{O}(N \log_2 N)$, which is great improvement over $\mathcal{O}(N^2)$ operations required by general multiplication by matrix F_N. Moreover, the product of permutation matrices P can be precomputed, and interestingly, it comes out as a permutation which sends index i to an index obtained from its binary representation by writing it in reversed order. This can be observed from the fact that P swaps the least and the most significant bits of the index. The recursively invoked Ps do the same thing with the remaining bits.

Formula (147) is sometimes called *'decimation in time'-FFT*. There also exist *'decimation in frequency'-FFT*, which can be obtained from (147) in the following way:

$$\begin{aligned} F_N = F_N^{\mathrm{T}} &= \left(\begin{pmatrix} I & I \\ I & -I \end{pmatrix} \begin{pmatrix} I & \\ & Q \end{pmatrix} \begin{pmatrix} F_{\frac{N}{2}} & \\ & F_{\frac{N}{2}} \end{pmatrix} \frac{P}{\sqrt{2}} \right)^{\mathrm{T}} \\ &= \frac{1}{\sqrt{2}} P^{\mathrm{T}} \begin{pmatrix} F_{\frac{N}{2}} & \\ & F_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I & \\ & Q \end{pmatrix} \begin{pmatrix} I & I \\ I & -I \end{pmatrix} \end{aligned} \quad (148)$$

The two 'decimations' are useful for convolution computation, because they allow not to perform multiplication by P at all[4]. In (144) we first have to do an inverse transform on a and b (by multiplying it by $\overline{F_N}$ from the left), then multiply the results point-by-point. Finally we have to perform forward transform and $\sqrt{N}$-scaling to obtain the result. As the point-by-point multiplication of the spectra does not depend on the ordering in which the points are stored, we can omit permutations altogether, provided that the forward transform will be able to read data in that order. This is exactly what decimation in frequency followed by a decimation in time does, since $PP^{\mathrm{T}} = I$.

The FFT, as it has been described, only works for vectors whose dimension is power of two. Similar decomposition can be carried out for other powers of primes and in case of compound numbers, these can be mixed. Practically it only pays-off for small numbers, like 3 and 5. To compute FFT of a vector of length of some larger prime number, the following trick can be done:

$$\begin{aligned} \sqrt{N} c_f &= \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nf} = \sum_{n=0}^{N-1} x_n e^{\frac{\pi i}{N}\left((f-n)^2 - f^2 - n^2\right)} \\ &= e^{-\frac{\pi i}{N} f^2} \sum_{n=0}^{N-1} \underbrace{\left(x_n e^{-\frac{\pi i}{N} n^2}\right)}_{y_n} \underbrace{e^{\frac{\pi i}{N}(f-n)^2}}_{a_{f-n}} \end{aligned} \quad (149)$$

The last formula is clearly a convolution, where the index n goes from 0 to $N-1$ and the index $f-n$ from $-N+1$ to $N-1$. Let us take $M \geq 2N-1$ and define y and a to be M-dimensional vectors, zero padded outside the above mentioned ranges (negative indices in case of a has to be wrapped over M, such that $-i$ would be stored at $M-i$ for $i > 0$). Now, we can compute (149) using cyclic convolution lemma, with the Fourier transforms implemented as FFT if we took $M := 2^m$. This way it is possible to compute the Fourier transform of any M-dimensional vector in time $\mathcal{O}(N \log_2 N)$ at the expense of invoking power-of-two FFT two times[5] on vectors at most twice as long. Note that the precision also slightly suffers by longer computation but this is not a concern for majority of practical applications.

[4] Today's CPUs are not particularly fast in wild data reordering, so this can really improve speed.

4.2 Fourier Series

The previous periodic approach becomes cumbersome once we want to work with signals of different periods simultaneously as we must take the period large enough to be divisible by periods of all the signals involved. Moreover, there are natural phenomena which cannot be described even in this manner. For instance, voltage measured on a discharging capacitor goes to zero like e^{-t}, so it will never really reach zero and the signal would therefore be infinite in time[6]. For this reason, we would like to have infinite aperiodic signals with analogical operations (addition, $\odot$-multiplication and convolution) as in the periodic case. It is obvious that not every $\mathbb{Z} \to \mathbb{C}$ function can be considered as a signal in this sense. For instance, constant 1 convolved[7] with itself would lead to a divergent sum. That is why we will require the signals to fall to zero as time goes to plus or minus infinity. In fact, we will need more than that to assure convergence. The exact condition will emerge from this section, where the discrete Fourier transform will be generalized such that it could 'process' certain infinite sequences. Particularly, these will include all finite sequences (zero padded to infinity). To get initial felling of what this section is all about, let us imagine that we have a vector and we will zero-pad it to length N, watching its discrete Fourier transform, while going with N towards infinity. To allow the spectrum to stabilize, we will watch $\sqrt{N} F_N x$ instead of plain $F_N x$. Also, imagine the resulting vector to be plotted onto the interval $[0, 1)$. As N goes to infinity, it is reasonable to expect that the shape stabilizes and in the limit it becomes continuous $[0, 1) \to \mathbb{C}$ function. This section formulates this idea precisely and more generally as the signal x will be allowed to be truly infinite, not just zero-padded.

I will extensively use complex numbers, especially integral from complex function of a real variable. It is understood formally as follows.

$$\int f(x) + ig(x)\ dx := \int f(x)\ dx + i \int g(x)\ dx \quad (150)$$

Derivative of $\mathbb{R} \to \mathbb{C}$ functions will be treated similarly. Note that it is different from true complex derivative, which operates on $\mathbb{C} \to \mathbb{C}$ functions. Before really beginning, let us remind some definitions.

[5] It is assumed that FFT of a has been precomputed as it is constant for any given N.

[6] In the real world this signal falls below the level of thermal noise quite soon, so it becomes practically indistinguishable from its finite portion (noise-padded to infinity). But we want to work with ideal unquantized and noise free signals, at this level of abstraction. It is easier to first introduce ideal noise-free signals and add the noise afterwards.

[7] Convolution of infinite signals will be defined in 4.3.1

▷ 83

4.2.1 Definition *Piecewise Continuous Function*

$f : [a,b] \to \mathbb{C}$ will be called piecewise continuous on $[a,b]$ iff it is continuous on $[a,b]$ except for finite number of points in which finite one-sided limits exist (both limits have to exist in inner points, the relevant one is enough in a and b).

4.2.2 Definition *Rectified Function*

For function $f : \mathbb{R} \to \mathbb{C}$ the following symbols will be used (if the limit is defined)

$$f^+(x) := \lim_{t \to x^+} f(t) \qquad f^-(x) := \lim_{t \to x^-} f(t) \qquad f^\pm(x) := \lim_{t \to x} f(t) \tag{151}$$

4.2.3 Observation *Any piecewise continuous function $f : [a,b] \to \mathbb{C}$ is bounded.*

Proof Let us have all its discontinuity points $a = x_1 < x_2 < \cdots < x_{n-1} < x_n = b$. Then, since the one-sided limits must exist in them, we know that[8] the function $(f \restriction [x_i, x_{i+1}])^\pm$ is continuous on compact set $[x_i, x_{i+1}]$, therefore it is bounded there and that is why f itself is bounded on (x_i, x_{i+1}). Finally, the values of $f(x_i)$ do not spoil the boundedness, since there are only finitely many of them. Q.E.D.

4.2.4 Lemma *Let the function g be piecewise continuous on $[a,b]$. Then*

$$\lim_{y \to \infty} \int_a^b g(x) \sin(yx)\ dx = 0 \tag{152}$$

Proof g is bounded by 4.2.3, so $\int_a^b = \sum_i \int_{x_i}^{x_{i+1}}$, where the function we integrate is continuous inside each interval, and it has the limits at its endpoints. As the value of such an integral does not depend on the value of g in the endpoints themselves, we can assume that g was continuous. Moreover it is enough to analyze what happens in one such interval, so without the loss of generality we will only examine continuous function g on interval $[0,1]$. Because $[0,1]$ is compact, g is also uniformly continuous. This implies that for any $\varepsilon > 0$ there is large enough n such that $\forall k \in \{0, \ldots, n-1\} : \forall x \in \left[\frac{k}{n}, \frac{k+1}{n}\right] : \left|g(x) - g\left(\frac{k}{n}\right)\right| < \varepsilon$. Now, let us take large enough y, such that

$$y > \frac{1}{\varepsilon} \underbrace{\sum_{i=0}^{n-1} \left| g\left(\frac{k}{n}\right) \right|}_{S} \tag{153}$$

We have

$$\left| \int_0^1 g(x) \sin(yx)\ dx \right| = \left| \sum_{k=0}^{n-1} \int_{k/n}^{(k+1)/n} g(x) \sin(yx)\ dx \right| = \tag{154}$$

$$= \left| \sum_{k=0}^{n-1} \left(\int_{k/n}^{(k+1)/n} \left(g(x) - g\left(\frac{k}{n}\right) \right) \sin(yx)\ dx + g\left(\frac{k}{n}\right) \cdot \int_{k/n}^{(k+1)/n} \sin(yx)\ dx \right) \right| \leq$$

[8] $f \restriction A$ denotes function restricted to domain A, that is $f \restriction A := \{\langle x, y \rangle \in f \mid x \in A\}$

$$\leq \sum_{k=0}^{n-1} \int_{k/n}^{(k+1)/n} \varepsilon\, dx + \sum_{k=0}^{n-1} \left|g(k/n)\right| \cdot \left| \int_{k/n}^{(k+1)/n} \sin(yx)\ dx \right| \leq \tag{155}$$

$$\varepsilon + \sum_{k=0}^{n-1} \left|g(k/n)\right| \cdot \left| \frac{\cos(yk/n) - \cos(y(k+1)/n)}{y} \right| \leq \varepsilon + \sum_{k=0}^{n-1} \left|g(k/n)\right| \cdot \frac{2}{y} \leq \varepsilon + \frac{2S}{y} \leq 3\varepsilon$$

Hence the limit is 0. Q.E.D.

4.2.5 Definition *Piecewise Differentiable Function*

$f : [a,b] \to \mathbb{C}$ will be called piecewise differentiable on $[a,b]$ iff it is piecewise continuous and it has first derivative on $[a,b]$ except for finite number of points. In these points, finite derivatives from the left and from the right of $f^-(x)$ and $f^+(x)$, respectively, have to exist (in a from the right and in b from the left is enough).

4.2.6 Definition *Periodic Function*

An $\mathbb{R} \to \mathbb{C}$ function f will be called *T-periodic* iff $p(x) = p(x+T)$.

4.2.7 Theorem *(Fourier Series) Let C be 1-periodic piecewise differentiable function. Let us define*

$$c_k := \int_{-1/2}^{1/2} C(f)\, e^{2\pi i f k}\ df \qquad \text{and} \qquad C_n(x) := \sum_{k=-n}^{n} c_k e^{-2\pi i k x} \tag{156}$$

Then

$$\frac{C^+(x) + C^-(x)}{2} = \lim_{n \to \infty} C_n(x) \tag{157}$$

and the limit on the right side exists for all $x \in \mathbb{R}$.

Proof

$$C_n(x) = \sum_{k=-n}^{n} \int_{-1/2}^{1/2} C(f) e^{2\pi i k (f-x)}\ df = \int_{-1/2}^{1/2} C(f) \underbrace{\sum_{k=-n}^{n} e^{2\pi i k (f-x)}}_{D_n(f-x)}\, df \tag{158}$$

Function D_n is continuous (because it is a finite sum of continuous functions) and $D_n(0) = 2n+1$. For $x \neq 0$ we have the following geometric sum:

$$D_n(x) = \sum_{k=-n}^{n} \left(e^{2\pi i x}\right)^k = e^{-2\pi i x n} \cdot \sum_{k=0}^{2n} \left(e^{2\pi i x}\right)^k = \frac{e^{2\pi i x (n+1)} - e^{-2\pi i x n}}{e^{2\pi i x} - 1} \cdot \frac{e^{-\pi i x}}{e^{-\pi i x}} =$$

$$= \frac{e^{2\pi i x (n+\frac{1}{2})} - e^{-2\pi i x (n-\frac{1}{2})}}{e^{\pi i x} - e^{-\pi i x}} = \frac{\sin(2\pi x (n+\frac{1}{2}))}{\sin(\pi x)} \tag{159}$$

As C and D are periodic (with period 1), they can be shifted

$$C_n(x) = \int_{-1/2}^{1/2} C(f)D_n(f-x)\ df = \int_{-1/2}^{1/2} C(f+x)D_n(f)\ df = \tag{160}$$
$$= \int_{0}^{1/2} C(f+x)D_n(f)\ df + \int_{-1/2}^{0} C(f+x)D_n(f)\ df$$

By substituting $g=-f$, using the fact that $D_n(f)=D_n(-f)$ and by swapping the integration limits we obtain

$$C_n(x) = \int_0^{1/2} C(x+f)D_n(f)\ df + \int_0^{1/2} C(x-f)D_n(f)\ df = \int_0^{1/2} \big(C(x+f)+C(x-f)\big)D_n(f)\ df \tag{161}$$

We will evaluate the limit now.

$$\lim_{n\to\infty} C_n(x) = \lim_{n\to\infty} \int_0^{1/2} \big(C(x+f)+C(x-f)\big)D_n(f)\ df =$$
$$\lim_{n\to\infty} \int_0^{1/2} \big(C(x+f)-C^+(x)+C(x-f)-C^-(x)+C^+(x)+C^-(x)\big)D_n(f)\ df =$$
$$\lim_{n\to\infty} \left(\int_0^{1/2} \big(C(x+f)-C^+(x)+C(x-f)-C^-(x)\big)D_n(f)\ df \quad + \right.$$
$$\left. + \quad \big(C^+(x)+C^-(x)\big)\cdot \int_0^{1/2} D_n(f)\ df \right) \tag{162}$$

42 ◁ Now, for[4] $C=\lambda f.1$ we have $c_k=0$ except for c_0, which is 1 by (156). So, for $C=\lambda f.1$ we have $C_n(x)=1$ for all $n\in\mathbb{N}$ and from (161) we see, that $\int_0^{1/2} D_n(x)\ dx = 1/2$. Hence

$$\lim_{n\to\infty} C_n(x) = \frac{1}{2}\big(C^+(x)+C^-(x)\big) \quad + \tag{163}$$

$$\lim_{n\to\infty} \int_0^{1/2} \underbrace{\left(\frac{C(x+f)-C^+(x)}{f} + \frac{C(x-f)-C^-(x)}{f}\right)\frac{f}{\sin(\pi f)}}_{g(f)} \sin\left(2\pi f\left(n+\frac{1}{2}\right)\right)\ df$$

To prove that the last limit goes to zero we have to show that $g(f)$ is piecewise continuous, which enables us to use lemma 4.2.4. The only difficult point is 0, but there, $f/\sin(\pi f)$ has finite limit and $\frac{C(x+f)-C^+(x)}{f}$ goes to the derivative from the right of C in x, while $\frac{C(x-f)-C^-(x)}{f}$ goes to the derivative from the left of $-C$ in x. Both are defined and finite because C was piecewise differentiable. Q.E.D.

4.2.8 Theorem *(Uniqueness of the Fourier coefficients) Let us have c_k such that*

$$C(x) := \lim_{n\to\infty} \sum_{k=-n}^{n} c_k e^{-2\pi ikx} \tag{164}$$

converges everywhere and is a piecewise differentiable function. Then

$$c_p = \int_{-1/2}^{1/2} C(x)\, e^{2\pi ifp}\ dx \tag{165}$$

Proof Since $C(x)$ is bounded (as it is piecewise differentiable), we can swap the limit with the integral and get

$$\int_{-1/2}^{1/2} \left(\lim_{n\to\infty} \sum_{k=-n}^{n} c_k e^{-2\pi ikx}\right) e^{2\pi ifp}\, dx = \lim_{n\to\infty} \sum_{k=-n}^{n} c_k \int_{-1/2}^{1/2} e^{2\pi i(p-k)x}\, dx \tag{166}$$

where the integral is 1 iff $p=k$, and 0 otherwise. So we obtain c_p as a result. Q.E.D.

4.2.9 Note The above two theorems can be stated for general period p by simple change of variables as follows.

$$c_k := \frac{1}{p}\int_{-p/2}^{p/2} C(f)\, e^{2\pi ifk/p}\ df \qquad \text{then} \quad \frac{C^+(x)+C^-(x)}{2} = \sum_{k=-\infty}^{\infty} c_k e^{-2\pi ikx/p} \tag{167}$$

where C is p-periodic piecewise differentiable function and the coefficients c_k are determined uniquely for the C given.

4.2.10 Lemma *For a 1-periodic piecewise differentiable function C, the following holds (and the limit exists):*

$$\frac{C^+(x)+C^-(x)}{2} = \lim_{n\to\infty} \int_{-1/2}^{1/2} C(f)D_n(f-x)\ df \tag{168}$$

Proof Contained in proof of 4.2.7, starting by equation (160). Q.E.D.

4.2.11 Definition *Signal, σ, σ_F*

Let us have a function $c:\mathbb{Z}\to\mathbb{C}$ and let us define $C(f) := \lim_{n\to\infty}\sum_{k=-n}^{n} c_k e^{-2\pi ikf}$. Then, c will be called a *signal* iff $C(f)$ converges $\forall f$ and is piecewise differentiable. The set of all signals will be denoted as σ. Let $\sigma_F \subset \sigma$ stand for the set of all finite signals, that is signals with only finitely many non-zero elements c_k.

78◁ 78◁ **4.2.12 Note** According to 4.2.7, any sequence obtained from 1-periodic piecewise differentiable function by integral (156) is a signal.

4.2.13 Definition *$\mathcal{F}$, Spectrum, $\mathcal{S}$*

For signal $a\in\sigma$, $A(f) := \sum_{k=-\infty}^{\infty} a_k e^{-2\pi ikf}$ will be called its *spectrum* and abbreviated as $\mathcal{F}(a)$. In fact, $\mathcal{F}$ may be understood as a mapping from σ to the set of piecewise differentiable 1-periodic functions. Let the set of all spectra $\{\mathcal{F}(s) \mid s\in\sigma\}$ be denoted by $\mathcal{S}$.

4.2.14 Note $\mathcal{S}$ is a subset of the set of all 1-periodic piecewise differentiable function. By 4.2.7 we know that $\forall s\in\sigma : \mathcal{F}(s) = \frac{\mathcal{F}(s)^+ + \mathcal{F}(s)^-}{2}$. Therefore,

$$\mathcal{S} = \{C \mid C \text{ is 1-periodic piecewise differentiable function s.t. } 2C = C^+ + C^-\} \quad (169)$$

The reason to constrain ourselves to $\mathcal{S}$ is to have unique spectrum for a given signal.

4.2.15 Observation *σ and $\mathcal{S}$ are infinite-dimensional vector spaces, and $\mathcal{F}$ is a linear mapping $\sigma \to \mathcal{S}$.*

Proof Defining $(a+b)_k = a_k + b_k$ and $(\alpha a)_k = \alpha a_k$ for $a, b\in\sigma$ and $(A+B)(f) = A(f)+B(f)$ and $(\alpha A)(f) = \alpha A(f)$ for $A, B\in\mathcal{S}$ we see that all vector space axioms hold, provided that the results are also from σ and $\mathcal{S}$, respectively. To show this, let us begin by realizing that scaling by α is clear in both cases. The sum of two functions from $\mathcal{S}$ is also from $\mathcal{S}$ because it is piecewise differentiable and the property (169) holds, too. Having $a, b\in\sigma$ we know that $A = \lim s(a)_n$ and $B = \lim s(b)_n$ exist. Then, $\lim s(c)_n$ also exists (for $c = a+b$), being equal to $A+B$. Hence $c\in\sigma$. As for $\mathcal{F}$, we have to show that $\mathcal{F}(a+\alpha b) = \mathcal{F}(a) + \alpha\mathcal{F}(b)$. As we now know that $a+\alpha b$ is also $\in\sigma$ (which means that the limit in $\mathcal{F}(a+\alpha b)$ converges) the equality follows straightforwardly from the fact that $\lim(x_n + y_n) = \lim x_n + \lim y_n$. Q.E.D.

4.2.16 Definition *Zero and Unit-Impulse Signals*

Let the signal z such that $\forall k : z_k = 0$ be denoted by $\vec{0}$. Let the signal u such that $u_0 = 1$ and $\forall k \neq 0 : u_k = 0$ be denoted by $\vec{1}$.

4.2.17 Note $\mathcal{F}(\vec{0}) = 0$ and $\mathcal{F}(\vec{1}) = 1$.

4.2.18 Lemma *$\mathcal{F}$ is one-to-one mapping* — $\forall a, b\in\sigma, a\neq b : \mathcal{F}(a) \neq \mathcal{F}(b)$.

Proof Let us have $a, b\in\sigma$ such that $\mathcal{F}(a) = \mathcal{F}(b)$. Using linearity of $\mathcal{F}$, we have that $\mathcal{F}(a-b) = 0$. Using uniqueness 4.2.8 we know that $a - b = \vec{0}$, hence $a = b$. Q.E.D.

4.2.19 Definition $\mathcal{F}^{-1}$

Now it is valid to define inverse $\mathcal{F}^{-1} : \mathcal{S} \to \sigma$ in the following way:

$$\mathcal{F}^{-1}(A) := \lambda k. \int_{-1/2}^{1/2} A(f)\, e^{2\pi ifk}\, df \quad (170)$$

4.2.20 Theorem *(Parseval's equality) Let $c\in\sigma$ and $\mathcal{F}(c) = C$. Then*

$$\lim_{n\to\infty} \sum_{k=-n}^{n} ||c_k||^2 = \int_{-1/2}^{1/2} ||C(f)||^2\, df \quad (171)$$

Proof

$$\sum_{k=-\infty}^{\infty} c_k\overline{c_k} = \sum_{k=-\infty}^{\infty} \int_{-1/2}^{1/2} C(f)e^{2\pi ikf}\, df \int_{-1/2}^{1/2} \overline{C(f)}e^{-2\pi ikf}\, df$$
$$= \lim_{n\to\infty} \sum_{k=-n}^{n} \int_{-1/2}^{1/2} C(f) \int_{-1/2}^{1/2} \overline{C(g)}e^{2\pi ik(f-g)}\, dg\, df \quad (172)$$

The sum is finite so it can be propagated inside the formula and because the function inside the outer integral is bounded, the limit can be moved inside.

$$\int_{-1/2}^{1/2} C(f) \lim_{n\to\infty} \int_{-1/2}^{1/2} \overline{C(g)} \sum_{k=-n}^{n} e^{2\pi ik(f-g)}\, dg\, df = \int_{-1/2}^{1/2} C(f) \lim_{n\to\infty} \int_{-1/2}^{1/2} \overline{C(g)} D_n(f-g)\, dg\, df \quad (173)$$

By lemma 4.2.10 we have

$$\sum_{k=-\infty}^{\infty} c_k\overline{c_k} = \int_{-1/2}^{1/2} C(f)\overline{C(f)} df \quad (174)$$

Q.E.D.

4.2.21 Note Parseval's equality ensures that $||s||^2 = \lim_{n\to\infty} \sum_{k=-n}^{n} s_k\overline{s_k}$ converges for every $s\in\sigma$. Note that the converse is not true. For instance $s_n := \frac{1}{|n|+1}$ has $||s||^2$ finite but $\mathcal{F}(s)(0) = \infty$, hence $s\notin\sigma$. The physical meaning of $||s||^2$ is total energy of the signal. To justify this, let us imagine that s_k is the input voltage at time point k. To maintain this voltage, signal source has to provide power $P_k = s_k^2/R_{in}$, where R_{in} is the input resistance of preamplifier. So, during each sampling period[9] T, work P_kT has to be carried out. Therefore, $||s||^2$ is proportional to the signal's energy (by factor T/R_{in}). Parseval's equality states that the energy of the signal is conserved by $\mathcal{F}$ transformation (if we also change $\sum$ to $\int$ in the formula for energy).

4.2.22 Historical Remark Fourier series predate discrete Fourier transform, and they were originally formulated in real numbers. In this formulation the function $C : \mathbb{R}\to\mathbb{R}$ meant continuous-time 1-periodic signal. Its spectrum consisted of a_k and b_k, where in our notation, $a_k + ib_k = c_k$. Note that for real C we have $c_{-k} = \overline{c_k}$, thus it is enough to specify a_k and b_k for $k\geq 0$ only. The series can than be considered as a way of expressing function C as an infinite sum of suitably weighted sines and cosines

$$\frac{C^+(t) + C^-(t)}{2} = a_0 + 2\sum_{n=1}^{\infty} (a_k \cos(2\pi nt) + b_k \sin(2\pi nt)) \quad (175)$$

[9] Imagine that we still work with extremely high sampling frequency

or if we use the identity

$$A_n \sin(2\pi nt + \varphi_n) = \underbrace{A_n \cos\varphi_n}_{2b_n} \sin(2\pi nt) + \underbrace{A_n \sin\varphi_n}_{2a_n} \cos(2\pi nt) \tag{176}$$

it may be understood as a decomposition of C into shifted and scaled sines, so called harmonic components. Note that we are using the series the other way round. We have discrete time and continuous frequencies. So, what used to be considered as spectrum we understand as a signal today, and vice-versa. Also, note that even if all c_ks of our signal were real, C could still be complex (and $C(f) = \overline{C(-f)}$ would hold).

4.3 Signals and Operations with them

80 ◁ Let us remind that σ denotes the set of all signals (4.2.11), $\mathcal{S}$ denotes the set of all spectra and $\mathcal{F}$ stands for one-to-one correspondence between the two sets. Sequences not in σ will not be considered as signals. From the practical point of view this is not a restriction because σ includes all finite-duration signals (zero padded to infinity), that is all physically meaningful signals. We have already seen that it also makes definition of total energy of any signal correct (and finite). In this section operation with signals will be introduced and shown to be well defined. Their basic properties will be proved, too.

In 4.2.15 we have seen that σ and $\mathcal{S}$ are vector spaces. Hence we have addition and scaling operations already. Definitions of other operations together with their effect in $\mathcal{S}$-space will follow. For the rest of this section, let us have $a, b \in \sigma$, $A = \mathcal{F}(a)$ and $B = \mathcal{F}(b)$.

4.3.1 Definition *Convolution*

$$(a * b)_n := \lim_{n\to\infty} \sum_{k=-n}^{n} a_{n-k} \cdot b_k \tag{177}$$

Proof To demonstrate correctness of the above definition, we have to show that $c := a * b \in \sigma$. To do this, we will show that $\mathcal{F}(a * b)$ converges to $C \in \mathcal{S}$. Let us have 78 ◁ 80 ◁ $C = \mathcal{F}(a) \cdot \mathcal{F}(b)$. Clearly $C \in \mathcal{S}$. Using 4.2.7 and 4.2.8 we can reconstruct c in the following way

$$c_p = \int_{-1/2}^{1/2} A(f)B(f)e^{2\pi ifp}\, df = \int_{-1/2}^{1/2} \lim_{n\to\infty} \left(\sum_{k=-n}^{n} a_k e^{-2\pi ifk}\right)\left(\sum_{k=-n}^{n} b_k e^{-2\pi ifk}\right) e^{2\pi ifp}\, df \tag{178}$$

Since $A(f)B(f)$ is bounded, we can swap the limit with the integral to obtain[10]

$$\lim_{n\to\infty} \sum_{k=-n}^{n}\sum_{l=-n}^{n} a_k b_l \int_{-1/2}^{1/2} e^{-2\pi if(k+l-p)}\, df = \lim_{n\to\infty} \sum_{k=-n}^{n}\sum_{l=-n}^{n} a_k b_l \delta_{k+l-p} =$$

[10] δ is Kronecker's delta. That is $\delta_x = 0$ except for $\delta_0 = 1$.

$$= \lim_{n\to\infty} \sum_{l=-n}^{n} a_{p-l} b_l \tag{179}$$

Now, from 4.2.7 we see that c is well defined. Moreover, $\mathcal{F}(a * b) = \mathcal{F}(a)\mathcal{F}(b)$ which implies that the convolution is commutative, associative (recall (142)) and because of linearity of $\mathcal{F}$ it is also distributive with addition. Q.E.D. ▷ 74

4.3.2 Definition *Spectral Convolution*

For $A, B \in \mathcal{S}$, $A * B$ will be defined as follows

$$A * B := \lambda f. \int_{-1/2}^{1/2} A(x)B(f-x)\, dx \tag{180}$$

Proof It is straightforward that $A * B \in \mathcal{S}$, since the integration cannot spoil piecewise differentiability (integration smooths out discontinuities hence (169) automatically holds). Q.E.D.

4.3.3 Definition *Multiplication*

$$(a \odot b)_n := a_n \cdot b_n \tag{181}$$

Proof To show the correctness, we will show that $\mathcal{F}(a \odot b) = A * B$. We know that $A * B \in \mathcal{S}$, from previous paragraph. Therefore it is in $\mathrm{Rng}(\mathcal{F})$ and it is valid to compute its inverse.

$$\mathcal{F}^{-1}(A * B)_n = \int_{-1/2}^{1/2} (A * B)(f) e^{2\pi ifn}\, df = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} A(x)B(f-x)\, dx\, e^{2\pi ifn}\, df \tag{182}$$

Now, we exchange the integrals using Fubini's theorem, which is valid since $A * B$ is piecewise continuous.

$$\int_{-1/2}^{1/2}\int_{-1/2}^{1/2} A(x)B(\underbrace{f-x}_{y})e^{2\pi ifn}\, df\, dx = \int_{-1/2}^{1/2}\int_{-1/2-x}^{1/2-x} A(x)B(y)e^{2\pi i(x+y)n}\, dy\, dx \tag{183}$$

Using periodicity and uniqueness 4.2.8, we obtain

$$\int_{-1/2}^{1/2}\int_{-1/2}^{1/2} A(x)e^{2\pi ixn}B(y)e^{2\pi iyn}\, dy\, dx = \int_{-1/2}^{1/2} A(x)e^{2\pi ixn}\, dx \cdot \int_{-1/2}^{1/2} B(y)e^{2\pi iyn}\, dy = a_n \cdot b_n \tag{184}$$

Therefore the definition is valid and $\mathcal{F}(a \odot b) = \mathcal{F}(a) * \mathcal{F}(b)$. Q.E.D.

4.3.4 Definition *Time Shift*

Time-shifted signal $a[k]$ is a signal a advanced[11] by k sampling periods, or

$$a[k]_n := a_{k+n} \tag{185}$$

Proof $\vec{1}[k] \in \sigma$ for any k, since it is a finite signal and we already know that σ contains all finite signals. For general a we have $a[k] = a * (\vec{1}[k])$, thus the definition is valid. Note that $\mathcal{F}(a[k]) = e^{2\pi i f k}\mathcal{F}(a)$. Q.E.D.

4.3.5 Definition *Time Reversal*

The symbol ρa stands for a 'played' backwards, that is

$$(\rho a)_n := a_{-n} \tag{186}$$

Proof It is obvious that time reversal cannot make the limit of $\mathcal{F}$ diverge. Therefore, ρ is well defined. But for sake of completeness, I will show how it manifests in the spectrum.

$$\mathcal{F}(\rho a)(f) = \sum_{n=-\infty}^{\infty} a_{-n}e^{-2\pi i n f} = \sum_{n=-\infty}^{\infty} a_n e^{2\pi i n f} = \mathcal{F}(a)(-f) \tag{187}$$

So, defining $(\rho A)(x) := A(-x)$, we can write $\mathcal{F}(\rho a) = \rho\mathcal{F}(a)$. Q.E.D.

4.3.6 Definition *Complex Conjugate*

$$(\overline{a})_n := \overline{a_n} \tag{188}$$

Proof It is clear that $\overline{a} \in \sigma$. For its spectrum, we have

$$\mathcal{F}(\overline{a})(f) = \sum_{n=-\infty}^{\infty} \overline{a_n}e^{-2\pi i n f} = \sum_{n=-\infty}^{\infty} \overline{a_n e^{2\pi i n f}} = \overline{\mathcal{F}(a)(-f)} \tag{189}$$

So $\mathcal{F}(\overline{a}) = \overline{\rho\mathcal{F}(a)}$, or equivalently, $\overline{\mathcal{F}(a)} = \mathcal{F}(\rho\overline{a})$. Q.E.D.

4.3.7 Definition *Function Application*

For a polynomial $f : \mathbb{C} \to \mathbb{C}$ such that $f(0) = 0$ we define

$$(f(a))_n := f(a_n) \tag{190}$$

Proof For $(f(a))_n = \sum_{k=1}^{N} \alpha_k a_n^k$, we have

$$\mathcal{F}(f(a)) = \sum_{k=1}^{N} \alpha_k \underbrace{(\mathcal{F}(a) * \cdots * \mathcal{F}(a))}_{k-\text{times}} \tag{191}$$

therefore the signal $f(a)$ is in σ. Q.E.D.

[11] Shifted to the left

4.3.8 Note Non-linear function applied on a band limited signal spreads its spectrum, as can be seen from (191). Each spectral convolution broadens the resulting band by the bandwidth of the input signal. High enough sampling rate (or narrow enough band of the input signal) has to be used, to avoid wrap-around aliasing (4.10.1). ▷ 115

4.3.9 Definition *Inverse, σ_I*

For a such that $\forall f \in \mathbb{R} : \mathcal{F}(a)(f) \neq 0$, we define its *inverse* as

$$a^{-1} := \mathcal{F}^{-1}(1/\mathcal{F}(a)) \tag{192}$$

The set of all invertible signals will be denoted as σ_I.

Proof To prove the correctness we have to show that $1/\mathcal{F}(a) \in \mathcal{S}$. Having $\mathcal{F}(a)$ non-zero we get $|\mathcal{F}(a)| \geq \varepsilon_0 > 0$ owing to its uniform continuity. Hence $1/\mathcal{F}(a)$ is bounded and its derivative is also well defined. Thus it is a piecewise differentiable function and the property (169) also holds. Q.E.D. ▷ 81

4.3.10 Observation *For $a \in \sigma_I$ we have $a^{-1} * a = \vec{1}$ and $\forall b \in \sigma : (b * a = \vec{1}$ implies $b = a^{-1})$*

Proof $a^{-1} * a = \mathcal{F}^{-1}\mathcal{F}(a^{-1} * a) = \mathcal{F}^{-1}\left(\mathcal{F}(a^{-1})\mathcal{F}(a)\right) = \mathcal{F}^{-1}\left((1/\mathcal{F}(a))\mathcal{F}(a)\right) = \mathcal{F}^{-1}(1) = \vec{1}$. Let us have b such that $b * a = \vec{1}$. Then by distributivity we get $(b - a^{-1}) * a = \vec{0}$, that is $\mathcal{F}(b - a^{-1})\mathcal{F}(a) = 0$. Since $\mathcal{F}(a)$ is non-zero, $\mathcal{F}(b - a^{-1})$ must be zero everywhere, therefore by uniqueness 4.2.8 we know that $b - a^{-1} = \vec{0}$, hence $b = a^{-1}$. Q.E.D. ▷ 80

4.3.11 Note It is clear that periodic signals are not in σ because the sum inside $\mathcal{F}$ would be non-convergent. This means that it is not easy to analyze a system involving both aperiodic and periodic signals because these two have incompatible formalizations. Fortunately, we will not need it because every practical signal is finite, after all. Periodic signals may be approximated by their long enough finite portion, zero-padded to infinity.

4.4 Linear Time-Invariant (LTI) Systems

Many physical phenomena concerning sound may be approximated by integral and differential equations. In our discrete model, the derivatives could be approximated as differences — in the most primitive way by $x_n - x_{n-1}$. More advanced methods could use multipoint approximations, which are more resistant to noise. Higher order differentiations are also possible, for instance 2nd derivative can be approximated by $x_{n-1} - 2x_n + x_{n+1}$. System integrating its input could be modeled using the following recursive formula

$$y_n = x_n + \alpha y_{n-1} \tag{193}$$

where $x \in \sigma$ is the input and y is the output. For example, this could be a model of a voltage on a capacitor being fed by a current x. Generalizing the above examples we get so called *digital filter* [28].

$$y_n = \sum_{k=0}^{N} c_k x_{n-k} - \sum_{k=1}^{M} d_k y_{n-k} \tag{194}$$

The above formula could be used to model all the RLC networks and amplifiers contained in the analog path (provided that they are linear), it could be used to model sound reverberations in the room and it could be even (mis)used[12] to describe speech production.

Any linear model is necessarily an approximation, however. Even the sound propagation thru the air is a non-linear process (fortunately, this effect is practically negligible at non-harmful loudness levels). More importantly, there can be non-linearity in the channel (such as the non-linearity of low cost microphones) and usually there is some kind of non-linearity in the sound source itself. On the other hand, the non-linearity can often be separated so that the system could be described by an LTI model (194) whose output is subsequently processed by a non-linear function. According to (191), the result is complicatedly inter-modulated version of signal coming out of the LTI model. If the non-linearity was weak, the LTI's output would dominate the spectrum, leaving quadratic, cubic and higher order components of spectral convolution (191) small and spread over whole spectrum. It is often practical to use linear model to describe such non-linear system. As the intermodulation components will be small, hopefully comparable to noise, this may even lead to better behavior than that of more precise non-linear model, which would be prone to fit the noise as if it were intermodulation components of some signal. Besides, linearity and time invariance leads to relatively simple mathematics and that is why LTI systems are so widely used.

Nevertheless, it should be noted that when there is strong enough non-linearity in the feedback of otherwise linear system, very complicated behavior can occur, including the onset of chaos. Once this happens, the following theory is of little use.

4.4.1 Definition *Linear Time-Invariant System*

LTI system can be formally defined as a mapping $\mathcal{A}:\sigma\to\sigma$, such that for all $x,y\in\sigma$, $\alpha\in\mathbb{C}$ and $k\in\mathbb{Z}$ the following properties hold:

$$\begin{array}{rl} \text{linearity} & \mathcal{A}(x+\alpha y)=\mathcal{A}(x)+\alpha\mathcal{A}(y) \\ \text{time invariance} & \mathcal{A}(x[k])=\mathcal{A}(x)[k] \\ \text{boundedness} & \exists K_{\mathcal{A}}\in\mathbb{R}:\forall x\in\sigma:||\mathcal{A}(x)||\le K_{\mathcal{A}}||x|| \end{array} \tag{195}$$

82 ◁ where $||x||$ is the *2-norm* introduced in 4.2.21.

4.4.2 Theorem *Any LTI system $\mathcal{A}$ can be written as a convolution. Formally, for all $x\in\sigma$: $\mathcal{A}(x)=\mathcal{A}(\vec{1})*x$. We call the $\mathcal{A}(\vec{1})$ an impulse response.*

Proof Let $x\in\sigma_F$, then for suitable N

$$\mathcal{A}(x)=\mathcal{A}\left(\sum_{k=-N}^{N}x_k\vec{1}[-k]\right)=\sum_{k=-N}^{N}x_k\mathcal{A}(\vec{1})[-k]=x*\mathcal{A}(\vec{1}) \tag{196}$$

[12] Our vocal cords generate impulse trains (for voiced sounds) or noise (for unvoiced sounds). This sound then propagates up, thru our vocal tract, being partially reflected back and forth. All these paths finally add up, forming the output sound. These reflections and delays could be directly captured by formula (194), provided that they were time invariant. Unfortunately, this is not true in case of the vocal tract, changing its shape during articulation. But as the rate of this change is relatively slow (when compared with the speed of sound), the coefficients c_k and d_k can be understood as 'locally nearly constant'. As such, they can by updated once every 20 ms, for instance. This is used in LPC
153 ◁ analysis/synthesis method, which is in use in GSM telephony and in some front ends (see 5.2).

Now we want to prove that for general $y\in\sigma$ and every $n\in\mathbb{Z}$

$$\mathcal{A}(y)_n=\lim_{N\to\infty}\sum_{k=-N}^{N}y_k\mathcal{A}(\vec{1})_{n-k} \tag{197}$$

Let us be given an $\varepsilon>0$. Let us write y as $x+z$, where $x\in\sigma_F$ and $z\in\sigma$ such that z is zero in the interval $[-N,N]$ and let's take N such that $||z||^2<\varepsilon^2/K_{\mathcal{A}}^2$.

$$\left|\mathcal{A}(y)_n-\sum_{k=-N}^{N}y_k\mathcal{A}(\vec{1})_{n-k}\right|=\left|\mathcal{A}(y)_n-\sum_{k=-N}^{N}x_k\mathcal{A}(\vec{1})_{n-k}\right|=|\mathcal{A}(z)_n| \tag{198}$$

Using boundedness, we get

$$|\mathcal{A}(z)_n|^2\le||\mathcal{A}(z)||^2\le K_{\mathcal{A}}^2||z||^2<\varepsilon^2 \tag{199}$$

Hence $|\mathcal{A}(z)_n|<\varepsilon$, which implies that $\mathcal{A}(y)_n=(y*\mathcal{A}(\vec{1}))_n$. Q.E.D.

4.4.3 Consequence *Composition of LTI systems is commutative and associative.*

4.4.4 Example *(Brick-Wall Low Pass Filter)* LTI system which removes all frequencies above $f_0<1/2$ from the signal, while leaving everything below f_0 intact, has the following impulse response for $k\neq0$ (thanks to 4.3.1 and 4.4.2): ▷ 83

$$h_k=\int_{-1/2}^{1/2}H(f)e^{2\pi ifk}df=\int_{-f_0}^{f_0}e^{2\pi ifk}df=\frac{e^{2\pi if_0k}-e^{-2\pi if_0k}}{2\pi ik}=\frac{\sin(2\pi f_0k)}{\pi k} \tag{200}$$

where $h_0=2f_0$. Note that brick-wall filter is not a filter according to definition (194) because summation over h_k is infinite. Also note that although $\sum_k h_k$ converges it does not converge absolutely.

4.4.5 Generalized Digital Filter and its Solution

Let us analyze how the digital filter (194) transforms x to y. Before doing that, we'll generalize it even more into the following equation ($x\in\sigma$ is input, y is output).

$$\sum_{k=M}^{N}d_ky[-k]=\sum_{k=P}^{Q}c_kx[-k] \tag{201}$$

Practical disadvantage of this formula is that it no longer specifies how to compute new y_n from the old values, as formula (194) did. Whole input signal must be known in advance and (201) solved to obtain y. It turns out that this solution is unique, if we are interested in $y\in\sigma$ only. By zero padding d_k and c_k to infinity, we obtain signals d and c, so we can write (201) as

$$d*y=c*x \tag{202}$$

The convolution on the left side is used correctly because as we are searching for a signal solution, we already treat y as certain, yet, unknown element of σ. Now, from 4.3.10 we see that

$$y = (d^{-1} * c) * x = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(c)}{\mathcal{F}(d)}\right) * x \tag{203}$$

and that the solution is unique, provided that $d \in \sigma_I$. In case of $d \in \sigma_F \setminus \sigma_I$, the solution may or may not exist and in case it exists it may be non-unique. As this is an uninteresting case from the practical point of view, let us only consider $d \in \sigma_I \cap \sigma_F$ henceforth.

Note that $y \in \sigma$ does not follow from $x \in \sigma$ automatically. This means, that there may be other (non-signal) solutions y of (201) for a given input signal x. This can happen even in case of the original 'algorithmic' formulation (194) as will be demonstrated now. Let us have $c = \vec{1}$ and $d = \vec{1} - 2 \cdot \vec{1}[-1]$, leading to recurrence

$$y_n = x_n + 2y_{n-1} \qquad \text{or} \qquad y_{n-1} = \frac{y_n - x_n}{2} \tag{204}$$

Feeding it with $x = \vec{1}$, we could obtain two solutions depending on the evaluation direction. For left-to-right direction, we get $y_n = 2^n$ for $n \geq 0$ and zero otherwise, for right-to-left, the outcome is $y_n = -2^n$ for $n < 0$ and zero otherwise. The second solution is from σ, while the first was not[13]. So it seems from this example that the direction in which the filter is being evaluated is crucial for the result to be in σ. However, for more complicated filters (having more d_ks non-zero) it can happen that their direct evaluation in any direction would lead to exponentially growing results. Nevertheless, from (203) we know that solution in σ exists and it is possible to use this formula to compute any point of it, provided that $x \in \sigma_F$. More practical method will be developed now. Let us write (203) in frequency domain in the following way.

$$\mathcal{F}(y)(f) = \left(\frac{\mathcal{F}(c)(f)}{\mathcal{F}(d)(f)}\right) \cdot \mathcal{F}(x)(f) = H(f) \cdot \mathcal{F}(x)(f) \tag{205}$$

Then, substituting[14] $z := e^{2\pi i f}$ into it, we obtain (assuming $d_M \neq 0$, since $d \in \sigma_I$):

$$H(f) := \left(\frac{\mathcal{F}(y)}{\mathcal{F}(x)}\right)(f) = \left(\frac{\mathcal{F}(c)}{\mathcal{F}(d)}\right)(f) = \frac{\sum_{k=P}^{Q} c_k z^{R-k}}{\sum_{k=M}^{N} d_k z^{R-k}} = \frac{\sum_{k=R-Q}^{R-P} c_{(R-k)} z^k}{\sum_{k=R-N}^{R-M} d_{(R-k)} z^k} \tag{206}$$

where $R = \max(Q, N)$. Using the fundamental theorem of algebra, we can factor the polynomials and get

$$H(f) = \frac{\sum_{k=0}^{R-P} c_{(R-k)} z^k}{\sum_{k=0}^{R-M} d_{(R-k)} z^k} = \frac{c_P}{d_M} \cdot \frac{\prod_{k=1}^{R-P}(z - z_k)}{\prod_{k=1}^{R-M}(z - p_k)} = \frac{c_P}{d_M} \cdot \frac{\prod_{k=1}^{R-P}(e^{2\pi i f} - z_k)}{\prod_{k=1}^{R-M}(e^{2\pi i f} - p_k)} \tag{207}$$

[13] There are other non-σ solution — any affine combination of the above solutions is also a solution.

[14] Herein, I assume this substitution to be only formal. But it is possible to treat formulas with z as $\mathbb{C} \to \mathbb{C}$ functions (in variable z) and it is even possible to substitute z into $\mathcal{F}$ this way, which is then called the z-transform and, in fact, it is a mapping between sequences and complex functions expressed as Laurent's series. Due to the lack of space and because we would not need it after all, I will not go into any details, here. Variable z should be considered only as a lexical shortcut for $e^{2\pi i f}$. In figures, z^{-1} will be used as a symbol for delaying the signal by one sampling period, because this is what $e^{-2\pi i f}$ does with the signal, when multiplied with its spectrum.

where $z_1 \ldots z_{R-P}$ and $p_1 \ldots p_{R-M}$ are roots of the polynomials in nominator and denominator, respectively. In accordance with complex analysis, z_is are called *zeroes* and p_is are called *poles*. Hence we see, that filter (201) is fully described by its poles, zeroes and the amplification factor c_P/d_M. This description is more convenient than original coefficients c_k and d_k because it leads to decomposition (208), which is closely related to implementation. Using the fact that the convolution is commutative and associative and that it manifests itself in spectral domain as multiplication of the spectra, we can write $h := \mathcal{F}^{-1}(H)$ as a convolution of individual zero/pole terms, that is

$$h = \frac{c_P}{d_M}\mathcal{F}^{-1}(z - z_1) * \cdots * \mathcal{F}^{-1}\left(z - z_{(R-P)}\right) * \mathcal{F}^{-1}\left(\frac{1}{z - p_1}\right) * \cdots * \mathcal{F}^{-1}\left(\frac{1}{z - p_{(R-M)}}\right) \tag{208}$$

which can be read either as a formula for obtaining h or as an algorithm of filtering the input signal x, if we use associativity in $h * x$. This way, it is possible to first convolve $x \cdot c_P/d_M$ with $\mathcal{F}^{-1}(z - z_1)$ then, convolve the result with $\mathcal{F}^{-1}(z - z_2)$ and so on. We may think of it as of a serial connection of single pole or single zero filters. Moreover, the order in which they are connected is irrelevant because the convolution is commutative and associative. This allows us to first convolve x with all $\mathcal{F}^{-1}(z - z_i)$ then continue with those $\mathcal{F}^{-1}\left((z - p_i)^{-1}\right)$ that can be evaluated from the left to the right and finally finish with those $\mathcal{F}^{-1}\left((z - p_i)^{-1}\right)$ which lead to a signal solution only when evaluated from the right to the left. This will be worked out in the next subsection. Let's finish this one with few definitions.

4.4.6 Definition *ZP-Canceled Digital Filter*

Zero/pole specification of a digital filter (207) will be called *zp-canceled* iff $\forall i, j : z_i \neq p_j$ and $\forall i : z_i \neq 0$ and $\forall i : p_i \neq 0$.

The first condition asserts that there are no redundant zero/pole pairs, while the second one forbids uninteresting poles or zeroes which would only shift the impulse response to the left or to the right. Such delaying or advancing is better to be excluded from the filter itself because it simplifies algebra. It is still fully general, provided that the input signal has been shifted accordingly, before it was fed into the filter. Therefore, the generalization (201) does not bring anything new over the original formulation (194), except for the insight gained from more symmetric formula. A filter

$$y_n = \sum_{k=0}^{N} c_k x_{n-k+N-M} - \sum_{k=1}^{M} d_k y_{n-k} \tag{209}$$

is described by

$$H(f) = z^{N-M} \frac{\sum_{k=0}^{N} c_k z^{-k}}{\sum_{k=0}^{M} d_k z^{-k}} = \frac{\sum_{k=0}^{N} c_{N-k} z^k}{\sum_{k=0}^{M} d_{M-k} z^k} = c_0 \frac{\prod_{k=1}^{N}(e^{2\pi i f} - z_k)}{\prod_{k=1}^{M}(e^{2\pi i f} - p_k)} \tag{210}$$

in frequency domain and it reads data only from x_{n-M} to x_{n+N-M} when computing value of y_n. Notice that the meaning of M and N is that of eq. (194), not (201). Also note that $c_0 \neq 0$ and $d_0 = 1$ correspond to the highest powers of the polynomials, while c_N and d_M are their constant terms. These are both non-zero for a zp-canceled filter.

4.4.7 Definition *One-Way and Causal Filters*

A filter is said to be *one-way* iff it can be evaluated from left to right, which happens iff all of its poles lie inside the unit circle (that is $\forall i : |p_i| < 1$). Moreover, if its impulse response h_k is zero for $k < 0$, it is called *causal*. This means that it does not need to know future inputs to be able to work. Clearly, any causal filter is one-way and any one-way filter can be suitably shifted to become causal. For instance, for h being an impulse response of (209), $h[M-N]$ would be causal, provided that h was one way.

4.4.8 Definition *FIR and IIR Filters*

Filter with impulse response $h \in \sigma_F$ will be called a *Finite Impulse Response* or *FIR*. It will be called an *Infinite Impulse Response* or *IIR* otherwise. Note that zp-canceled filter is FIR iff it does not have any poles.

4.5 Practical Digital Filtering

Let's work out how to convolve x with $\mathcal{F}^{-1}(z - z_0)$ and $\mathcal{F}^{-1}\left(\frac{1}{z-p_0}\right)$. Filter with frequency response $H(f) = z - z_0$ can be evaluated as[15]

$$y_n = -z_0 \cdot x_n + x_{n+1} \tag{211}$$

while filter with $H(f) = (z - p_0)^{-1}$ leads to[16]

$$-p_0 \cdot y_n + y_{n+1} = x_n \tag{212}$$

Thus, if $|p_0| < 1$, it should be evaluated from left to the right:

$$y_n = x_{n-1} + p_0 \cdot y_{n-1} \tag{213}$$

while for $|p_0| > 1$, it should be evaluated from right to the left[17]:

$$y_n = \frac{1}{p_0} \cdot (-x_n + y_{n+1}) \tag{214}$$

These directions of evaluation assure that y will be from σ if x was[18]. Let us now assume that x was a finite signal. To be specific, let us have $x_n = 0$ for all $n \in \mathbb{Z} \setminus [0, L]$. If all poles were inside the unit circle the filter is one-way. First, we can run (213) for all $(z-p_i)^{-1}$ terms, followed by (211) for all $z-z_i$ terms. As $x_n = 0$ for $\forall n < 0$, we can start (213) at $n = 1$ with initial condition $y_0 = 0$ to obtain correct solution, consistent with (203). This is the most common case of digital filtering. It has an advantage over

[15] Which follows from $y_n = \sum_k c_k x_{n-k}$ and $H(f) = \sum_k c_k e^{-2\pi i f k} = e^{-2\pi i f(-1)} - z_0 = z - z_0$.

[16] Because $\sum_k d_k y_{n-k} = x_n$ and $1/H(f) = \sum_k d_k e^{-2\pi i f k} = e^{-2\pi i f(-1)} - p_0 = z - p_0$.

[17] Note that $|p_0|$ cannot be 1 since $\mathcal{F}(d)(f) \neq 0$ because $d \in \sigma_I$.

[18] This is apparent from the fact that the impulse response of recurrence (212) falls exponentially towards zero if evaluated in correct direction, hence $\mathcal{F}$ is convergent on it and so the impulse response is in σ. Running (212) on a signal is equivalent to convolving it with its impulse response. As both operands of $*$ are in σ so must be result.

(203) that we are able to start outputting y_ns before we have read whole input x. In fact x_ns can be translated into y_ns with just some fixed delay, as can be easily seen from above equations.

4.5.1 Two-Way Filters

Nonetheless, were there poles outside the unit circle[19], we would have to use backward pass (214) for them. Such filters are called *two-way filters* in [28] as they also need forward pass if there are poles inside the unit circle, which is typical. Unfortunately, there is a problem with initial conditions for backward passes as these cannot be zero anymore. To illustrate the problem, let us have x, the input signal already filtered by all non-recursive passes (211) and multiplied by amplification factor c_0. As the input was finite in time, so will be x. Suppose we first run the forward passes (213) on x for all poles inside the unit circle, producing intermediate result $w \in \sigma$. Then, we would like to run backward passes (214) on w to obtain y, the desired output of the filter. However, as w extends indefinitely to the future time, there is a trouble in finding suitable starting point for (214). Note that zero initial conditions, although also leading to a signal from σ, would not give the true solution (203). To make this clear, let's imagine that a filter with only a single pole p_1 outside the unit circle is run backwardly in the following way, starting at $n = N$.

$$y_n = p_1^{-1} \cdot (-w_n + y_{n+1}) \tag{215}$$

It is clear that y_{N+1} must already agree with the general solution (203) once we want to use it as an initial condition in (215) to determine y_N consistent with (203). The formula (215), however, does not specify it in any way. Theoretical solution to this problem would be to compute this single value using slow but general equation (203). Then we could start (215) to do the rest. Would there be more out-of-unit-circle poles $(p_1, \ldots, p_L)$, the very same method could be used in several steps. First, we would compute signal y^1 using only the first pole, with initial condition y^1_{N+1} set to $\left(x * \mathcal{F}^{-1}\left(\left(\prod_{k=2}^{L}\left(e^{2\pi i f} - p_k\right)\right)/\mathcal{F}(d)\right)\right)_{N+1}$. Then, y^1 would be used as the input w of (215) to produce y^2, using $\left(x * \mathcal{F}^{-1}\left(\left(\prod_{k=3}^{L}\left(e^{2\pi i f} - p_k\right)\right)/\mathcal{F}(d)\right)\right)_{N+1}$ as an initial condition y^2_{N+1}. Continuing along these lines, we would finally end up with correct output y, after L steps.

Another, more practical, method of two-way filtering gives up on precise result and only computes an approximation of it. It will be shown that the numerical error could be affordably kept close to machine precision. Hence, the difference between precise and approximative solution would be practically indistinguishable. To compute this approximation, we would start backward recurrence (215) farther in time, say at $n = N+K$, with initial condition $y_{N+K+1} = 0$. Because of linearity, the obtained result y could be understood as a sum $y = \widehat{y} + \check{y}$, where $\widehat{y}$ is the output of (215) fed with w, started at $N + K$ with initial condition $\widehat{y}_{N+K+1}$ set in accordance with (203), and $\check{y}$ would represent the error-term, being an output of (215) processing $\vec{1}[-N-K]\widehat{y}_{N+K+1}$

[19] For instance, if we would like to construct an inverse to general FIR filter we would need poles both outside and inside the unit circle. This is because zeroes of the original filter were unrestricted and the inverse filter has to be constructed by making poles from zeroes, according to 4.3.10.

▷ 86

as its input, with initial condition $\check{y}_{N+K+1} = 0$. It is easy to check that the sum of these two filters yields to y. The second filter essentially outputs its own impulse response (shifted by $N+K$ and magnified by $\widehat{y}_{N+K+1}$), which falls exponentially towards zero. Hence, for affordable values of K, this error signal can be kept negligible. For filters with more poles outside the unit circle, this method can be easily generalized at the expense of slightly larger values of K.

Note that the approximative solution allows us to start processing the signal x before it is known it in its totality. Only K samples ahead of current position are needed once we can upper-bound the output $|y_n|$ (which we usually can). However, for acceptable effectiveness at least $2K$ samples ahead of current position would be reasonable. Using this method both ways, it is possible to compute approximation of
89 ◁ (203) even for infinite x, which is otherwise impossible to be done exactly in general, because of infinite summation involved in (203).

4.5.2 Time Reversed Filter

Two-Way filters are closely related with time reversal. Namely, the backward pass can be viewed as a forward pass on a time reversed signal using a time reversed impulse response, or formally

$$x * h = \rho(\rho x * \rho h) \tag{216}$$

Here, I will make clear what happens with zeroes and poles of a zp-canceled filter with impulse response h, when reversed into ρh. We have

$$\rho h = \rho\mathcal{F}^{-1}(H(f)) = \mathcal{F}^{-1}(H(-f)) \tag{217}$$

90 ◁ As the filter can be shifted by increasing M or N we can always describe it by (210):

$$H(-f) = \frac{\sum_{k=0}^{N} c_{N-k}e^{-2\pi ifk}}{\sum_{k=0}^{M} d_{M-k}e^{-2\pi ifk}} = c_0 \cdot \frac{\prod_{k=1}^{N}(e^{-2\pi if} - z_k)}{\prod_{k=1}^{M}(e^{-2\pi if} - p_k)} \tag{218}$$

Rewriting $e^{-2\pi if} - z_k$ as $(e^{-2\pi if}z_k)(z_k^{-1} - e^{2\pi if})$ we get

$$H(-f) = \left(c_0 \cdot (-1)^{(N-M)} \cdot \frac{\prod_{k=1}^{N} z_k}{\prod_{k=1}^{M} p_k}\right) \cdot \frac{\prod_{k=1}^{N}(e^{2\pi if} - 1/z_k)}{\prod_{k=1}^{M}(e^{2\pi if} - 1/p_k)} \cdot e^{2\pi if(M-N)} \tag{219}$$

Hence, the time reversal changes poles and zeroes of the filter to their respective reciprocal values (that is $z_k = Z_k e^{i\zeta_k}$ changes into $Z_k^{-1}\overline{e^{i\zeta_k}}$). The amplification factor of the filter is changed, too. The last factor performs time shift by $M - N$ samples.

4.6 Transfer Function and Group Delay

Any digital filter is fully specified by its impulse response h. Practically, however, it is more instructive to think about its *frequency response* $H(f) = \mathcal{F}(h)(f)$, also known as its *transfer function*. As H is an $\mathbb{R} \to \mathbb{C}$ function it worths to rewrite it using *amplitude* and *phase*, to gain further insight:

$$H(f) = |H(f)| \cdot \underbrace{\frac{H(f)}{|H(f)|}}_{\Phi(f)} = |H(f)| \cdot \underbrace{\Phi(f)}_{e^{i\varphi(f)}} = |H(f)| \cdot e^{i\varphi(f)} \tag{220}$$

Let us see now how it relates to the impulse response. Let us take (non-signal) sequence $x_n = e^{2\pi ifn}$ and let us convolve it with h to get $y = h * x$. Then

$$y_p = \lim_{n\to\infty} \sum_{k=-n}^{n} h_k e^{2\pi if(p-k)} = e^{2\pi ifp} \lim_{n\to\infty} \sum_{k=-n}^{n} h_k e^{-2\pi ifk} = (\mathcal{F}(h)(f))e^{2\pi ifp} \tag{221}$$

from which we see that $y = H(f)x$. This means that the exponential x acts as an eigenvector of linear operator $\mathcal{A}(x) = h * x$, where $H(f)$ is the respective eigenvalue. Using (220) we can write the output as $|H(f)|e^{i\varphi(f)}e^{2\pi ifn} = |H(f)|e^{2\pi ifn+i\varphi(f)}$, which can be interpreted as if x was amplified by $|H(f)|$ and than phase shifted by $\varphi(f)$. Due to linearity, this happens separately to every single frequency f which is eventually present in the input signal[20]. To clarify it further, let us explore this phenomenon in usual case of real valued signals convolved with real valued impulse responses. In this case, we are only interested in the real part of $e^{2\pi ifn}$, that is in $\widehat{x}_n = \cos(2\pi fn)$. As h is real, the convolution itself would not mix real and imaginary parts, therefore we can just take $\mathrm{Re}(h * x)$ to the determine $h * \widehat{x}$ we are after — these two are equal. Hence $\mathrm{Re}\left(|H(f)|e^{2\pi ifn+i\varphi(f)}\right) = |H(f)|\cos(2\pi fn + \varphi(f))$. Again, the signal was amplified by $|H(f)|$ and phase-shifted by $\varphi(f)$.

Electrical engineers found out that it is a good idea to plot $|H(f)|$ on a *log-log paper* as depicted in the upper panel of fig. 26. Although the rationale for doing so is more applicable in continuous time case, we will use it here, too. The log-log plot[21] means that as we linearly move along the paper, frequency changes exponentially and so does the amplitude. In fig. 26 the amplitude axis seems to be of linear scale, but this is because $|H(f)|$ is plotted in decibels, which means that instead of $|H(f)|$, we plot:
▷ 150

$$H_{dB}(f) = 20\log_{10}|H(f)| \tag{222}$$

$H_{dB}(f)$ is often referred to as the *magnitude response*, while $\varphi(f)$ is called the *phase response*. The magnitude response tells us how the signal's energy will be amplified or attenuated (that is filtered) at different frequencies. The phase response, unfortunately, does not have as vivid interpretation, apart from telling us how much will a pure sine be shifted would it appear on the filter's input. To get deeper understanding of it, it is useful to define so called group delay and a similar notion of wave delay.

4.6.1 Definition *Group Delay*

For a piecewise differentiable transfer function (220), the *group delay* at frequency f is defined as

$$\delta(f) := -\frac{1}{2\pi} \cdot \varphi'(f) \qquad \left[\frac{\mathrm{rad}}{2\pi\mathrm{Hz}} = \mathrm{s}\right] \tag{223}$$

where φ is meant to be as smooth as possible by jumping among neighboring branches of complex logarithm in $\varphi(f) = -i\log(H(f)/|H(f)|)$. Or, in another words, if we would like to compute $\varphi(f)$ using $\mathrm{atg}\left(\frac{\mathrm{Im}(H(f))}{\mathrm{Re}(H(f))}\right)$, we would have to add or subtract $\frac{\pi}{2}$ at certain points, when moving along f, to maintain continuity. The group delay is to be measured in units of seconds.

[20] This is mostly an informal explanation, the real reason why $\mathcal{F}(h * x) = |H|\Phi\mathcal{F}(x)$ is (179). ▷ 84

[21] Often called the *Bode Plot* after its inventor Hendrik Wade Bode.

4.6.2 Note $\delta(f)$ may remain undefined for finitely many points for general $H(f)$ even if proper branches of logarithm were taken. This is because $H(f)$ itself can be discontinuous in finitely many points. Fortunately, for digital filter (202), $H(f)$ is smooth (because $c\in\sigma_F$ and $d\in\sigma_F\cap\sigma_I$), which implies smoothness of φ'. For that reason, it can be computed the following way

$$\varphi'(f)=\lim_{g\to f}\left(\frac{\partial}{\partial g}\operatorname{atg}\left(\frac{\operatorname{Im}(H(g))}{\operatorname{Re}(H(g))}\right)\right) \tag{224}$$

in this case. Note that it does not matter that $\frac{\operatorname{Im}(H(g))}{\operatorname{Re}(H(g))}$ is undefined in finitely many points (due to division by zero). As long as we know that true φ' will be smooth anyway (and therefore continuous), the limit will recover the correct value.

The above definition is useful in theory. For numerical computation, however, it turns out to be cumbersome, owing to the continuity problems. The following theorem states an equivalent formula, better suited for numerical evaluation.

4.6.3 Lemma *For transfer function (220) the following holds*

$$\delta(f)=-\frac{\Phi'(f)}{2\pi i\Phi(f)} \tag{225}$$

Proof We have $\Phi'(f)=\frac{\partial}{\partial f}e^{i\varphi(f)}=i\varphi'(f)e^{i\varphi(f)}=i\varphi'(f)\Phi(f)$, where φ was as smooth as possible. Hence $\varphi'(f)=\frac{\Phi'(f)}{i\Phi(f)}$, therefore $\delta(f)=-\frac{\Phi'(f)}{2\pi i\Phi(f)}$. Q.E.D.

Sometimes it is more convenient to plot group delay using number of delayed waves as a unit, instead of seconds. This comes in handy when we want to plot group delay of two resonators, one being physically scaled version of the other, onto a single paper. As the larger one can have group delay orders of magnitude longer, it might get challenging to fit both of them on a single sheet. On the other hand, the number of delayed waves is often invariant to physical scaling, which makes it suitable for this purpose.

4.6.4 Definition *Wave Delay*

For a piecewise differentiable transfer function (220), the *wave delay* at frequency $f\in[-f_s/2,f_s/2)$ is

$$w(f):=\delta(f)\cdot|f| \tag{226}$$

To intuitively grasp the notion of group delay, let us imagine a filter with transfer function $H(f)=\exp(i\varphi(\alpha,\beta,f_0,f))$, where $\varphi(\alpha,\beta,f_0,f)=-2\pi(\beta f-(\beta-\alpha)f_0)$. Clearly, $\delta(f)=\beta$. Let f_s be the sampling frequency[22] and let us investigate what happens with signal $s=\mathcal{F}^{-1}(S)$, when filtered by $H(f)$. The output signal z will be

$$\begin{aligned}z_n&=f_s^{-1}\int_{-f_s/2}^{f_s/2}S(f)H(f)e^{2\pi ifn/f_s}df=f_s^{-1}\int_{-f_s/2}^{f_s/2}S(f)e^{2\pi i(fn/f_s-\beta f+(\beta-\alpha)f_0)}df\\&=\frac{e^{2\pi i(\beta-\alpha)f_0}}{f_s}\int_{-f_s/2}^{f_s/2}S(f)e^{2\pi if(n/f_s-\beta)}df\end{aligned} \tag{227}$$

For β being an integer multiple of sampling period $T_s=1/f_s$, the integral becomes a variant of $\mathcal{F}^{-1}$, delayed (see 4.3.4) by β/T_s samples, that is by β seconds. For general β, the integral can be taken as a definition of non-integer time shift, as will be shown in section 4.10.

If we imagine the signal as a three-dimensional object (two dimensions for real and imaginary part, one for the time — see fig. 19) we could see that it has been delayed by β seconds and rotated by an angle $2\pi i(\beta-\alpha)f_0$ along the time-axis, due to the factor $e^{2\pi i(\beta-\alpha)f_0}$. Another interpretation of this factor can be revealed by considering a narrow-band signal $w\in\sigma$ (having $W(f):=\mathcal{F}(w)(f)=0$ for $\forall f\in[f_1,f_s)$) modulated by a carrier $c_n=e^{2\pi inf_0/f_s}$, that is $s:=w\odot c$. Note that although $c\notin\sigma$, we have $s\in\sigma$ and $S(f)=W(f-f_0)$. Using (227) and periodicity of $S(f)$, we obtain

$$\begin{aligned}z_n&=\frac{e^{2\pi i(\beta-\alpha)f_0}}{f_s}\int_{-f_s/2}^{f_s/2}S(f)e^{2\pi if(\frac{n}{f_s}-\beta)}df=\frac{e^{2\pi i(\beta-\alpha)f_0}}{f_s}\int_{-f_s/2}^{f_s/2}W(\underbrace{f-f_0}_{g})e^{2\pi if(\frac{n}{f_s}-\beta)}df\\&=\frac{e^{2\pi i(\beta-\alpha)f_0}}{f_s}\int_{-f_s/2}^{f_s/2}W(g)e^{2\pi i(g+f_0)(\frac{n}{f_s}-\beta)}dg=\frac{e^{2\pi if_0(\frac{n}{f_s}-\alpha)}}{f_s}\int_{-f_s/2}^{f_s/2}W(f)e^{2\pi if(\frac{n}{f_s}-\beta)}df\end{aligned} \tag{228}$$

So, for α and β being a multiple of sampling period, the result is $z=w[-\beta f_s]\odot c[-\alpha f_s]$. This means that the narrow band signal has been delayed by β seconds, while the carrier wave by α seconds. As the carrier was periodic, α only caused its phase shift, for which reason it is sometimes called a *phase delay*.

General filter $H(f)$ can be expressed as $A(f)e^{i\varphi(f)}$, according to (220). The phase response $\varphi(f)$ can then be approximated by a segmented line, using $\varphi(\alpha_k,\beta_k,f_k,f)$ with appropriate values of α_k,β_k and f_k for each segment. Finally the input signal can be understood as a sum of narrow band signals each spanning frequencies from f_k to f_{k+1}. Due to linearity of filtering, we can process each group of frequencies $[f_k,f_{k+1})$ separately summing the results afterwards. As the groups were constructed such that f_ks in the signals match f_ks of the filter, the filtering obeys (227). Therefore, in each band, the signal would be delayed by β_k seconds, while its carrier would be delayed by α_k seconds. Usually, we are more interested in group delay than in phase delay because the phase delay only refers to the carrier wave, which does not convey any information. The group delay can be roughly understood as a time by which the information encoded in a particular band has been delayed by the filter.

[22] Until now, we thought of it as being very high, while at the same time we essentially used $f_s=1$ in formulas (this is called a *normalized frequency scale*). It is preferable in theory as it leads to shorter formulas. It is meant that the user will imagine numbers on an x-axis in a transfer function plot as if they were multiplied with f_s of his choice.

However, for demonstration that units of $\delta(f)$ are indeed seconds, it is better to work with concrete value of f_s at the expense of longer formalism. In that case, $\mathcal{F}$ must be adapted to account for general f_s, according to (167). The spectrum is then f_s-periodic, where in the time domain, samples s_n of the signal s are to be taken every $1/f_s$ seconds (but we still use $n\in\mathbb{Z}$ to index them).

In the sequel, I will use $\mathcal{F}$ freely in both senses. It will be apparent from the context which one I mean. Usually, I will stay with normalized frequency approach in theory, using true f_s mainly in plots.

88 ◁ 80 ◁ ▷ 85 ▷ 115 ▷ 125

The above analysis has been carried out for complex numbers but it also applies to a real-valued case. Let's see how. We'll start with a narrow band real-valued signal w and its spectrum $W(f)$. For a real-valued signal w and real-valued filter h we have $W(f) = \overline{W(-f)}$ and $H(f) = \overline{H(-f)}$. The signal is considered to be band-limited from $-f_1$ to f_1. We will modulate that signal by $d_n := 2\cos(2\pi f n f_0/f_s) = c_n + \overline{c_n}$, obtaining $w \odot d$, which is a signal actually entering the filter H. Because modulation by c effectively shifts spectrum by f_0 to the right and we assume that $f_1 < f_0$ the spectrum of $w \odot c$ will not interfere with spectrum of $w \odot \overline{c}$. In fact the first one is non-zero only for positive frequencies while the second one only for negative ones. Now, we reuse complex-valued formula (228) by defining

$$H(f) := \begin{cases} \exp(i\varphi(\alpha, \beta, f_0 - \frac{f_1}{2}, f)) & \text{for } f > 0 \\ \exp(-i\varphi(\alpha, \beta, f_0 - \frac{f_1}{2}, -f)) & \text{for } f < 0 \end{cases} \tag{229}$$

where $f \in [-f_s/2, f_s/2)$. Note that $H(f) = \overline{H(-f)}$ and if we defined $H_1(f)$ to be the first case of (229) and zero otherwise we could have written $H(f)$ as $H_1(f) + \overline{H_1(-f)}$
85 ◁ or in the time domain as $h = h_1 + \overline{h_1}$ according to (189), leading to

$$z = h * s = h * (w \odot d) = h * (w \odot (c + \overline{c})) = h_1 * (w \odot c) + \overline{h_1 * (w \odot c)} \tag{230}$$

By (228), it implies that

$$z = w[-\beta f_s] \odot c[-\alpha f_s] + \overline{w[-\beta f_s] \odot c[-\alpha f_s]} = w[-\beta f_s] \odot d[-\alpha f_s] \tag{231}$$

Again, the information content of the signal was delayed by β seconds while the carrier wave was delayed by α seconds. In '3D representation' of the signal we have to imagine the real signal to be a sum of two mutually conjugated complex signals, each of which being delayed and rotated counter the other one by the filter, which maintains cancellation of the imaginary components.

4.6.5 Negative Group Delay

Now, as we might think we understood group delay, let us look at the following filter:

$$y_n = x_n - 0.95\, x_{n-1} \tag{232}$$

150 ◁ whose transfer function and group delay are shown in fig. 26. Evidently, there is a region with small negative group delay. It makes more than 1 ms below 60 Hz. In principle, this delay can be made arbitrarily large by cascading the filter (232) and amplifying the result to compensate for the 26 dB attenuation at low frequencies. Does it mean that the output information concerning low frequency signals arrives before the input has entered the filter? Obviously not. This can be seen from the formula (232) which only uses data from the past.

But if we perform numerical simulation shown in fig. 14, we might get an impression that we are really sending information from the present to the past. Employing optical phenomenon called *anomalous dispersion*, it is even possible to build real device (as they did in [45]) which seems to be able to send light pulses at superluminal velocities or even to the past — if light detectors are mounted at the input and at the output of that device and accordingly band-limited pulse is sent into it, the input detector peaks *after* the output detector has peaked.

However, our joy of having invented a time-machine quickly vanishes if we consider what is happening with information here. First of all, the material which exhibits anomalous dispersion can be sliced into many sections each of which can be modeled as an LTI system followed by a delay the light would need to travel that section's length in the vacuum. This is acceptable approximation for sufficiently small amplitudes which do not yet invoke non-linear phenomena in the material. Hence the material can be viewed as a complicated digital filter. Negative group delay at certain frequencies comes as direct consequence of anomalous dispersion, then. So it makes sense to concentrate on much simpler filter (232) to understand what is really happening here.

The brick-wall filter used in fig. 14 removes everything above 60 Hz, leaving everything below intact. It has infinite impulse response and requires the input signal to be entirely known before the filtering could begin. As a result the original sharp impulse becomes smeared, with ripples coming towards infinity and minus infinity. It turns out that the ripples already contain enough information from which we can recover the signal advanced (or delayed) in time. In our example, the cascade of N filters (232) only needs samples from $t - N$ to t in order to create t-th output sample. Each section in the cascade advances the signal by about 1 ms and there is no theoretical limit on the number of stages as long as we used brick-wall filter — practically we are limited by round-off noise, though.

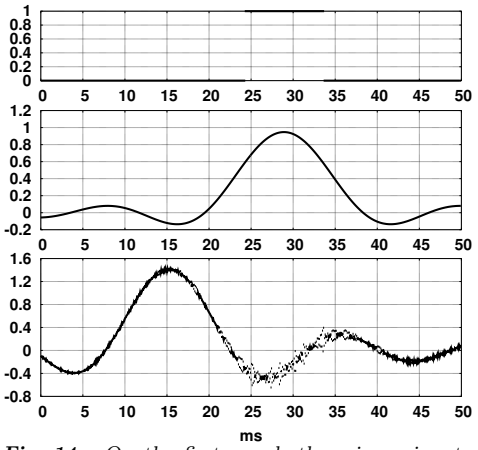


Fig. 14. *On the first panel, there is an input impulse sampled at 16 kHz. Below, there is the same impulse low-pass filtered by a brick-wall filter so that only frequencies up to 60 Hz remained. The last panel shows the middle signal after it has passed thru 12 stages of (232), what advanced it in time by about 14 ms. Notice slight deformation of the impulse shape which is caused by non-constant gain and group delay in low 60 Hz range. The noise is due to round-off errors which are magnified by cascade's gain which is about $12 \times 26 = 312$ dB at high frequencies. In fact, 12 stages were the maximum I was able to achieve with 80-bit floating point numbers before the noise took over the signal completely.*

So we have to conclude that it was the low pass filtering which made future information available in the past. In this light it comes at no surprise that the brick-wall filter needed to know entire input before it could begin its work.

As for the superluminal device described in [45], we cannot hope to have brick wall filter but as we want to advance pulses only by limited amount of time, much simpler low pass filter can be used. It is enough to attenuate unwanted frequencies such that they remain negligible after the light passes thru the machine. In case of fig. 14, this would mean that the low pass filter with 400 dB attenuation in the stop-band is already good enough to be used with 12 stage filter (232). The low pass filter can be shifted such that it would use samples from the past only — as a result, the pulse in the middle panel of fig. 14 moves to the right and so does the output pulse. Naturally, the last pulse could only be advanced to the level of the unfiltered input impulse because the impulse response of the low pass filter convolved with impulse response of the cascade

still uses data from the past only. So there is no communication to the past, since the output always appears after the unfiltered input. Only the middle signal has been delayed and fig. 14 just demonstrates that the delay can be compensated at the expense of noise and slight distortion of the signal's shape.

After all, there is nothing magical about causal digital filter exhibiting negative group delay at some frequencies. We will see in section 4.7 that for each minimum phase filter that does not have zeroes on the unit circle there exist its inverse which is also minimum phase. As the original filter had regions with positive group delay, the inverse one must have negative group delay there, so that the respective phase shifts would cancel out if the signal is filtered by a cascade which consists of a filter and its own inverse.

101 ◁

4.6.6 Numerical Estimation of the Group Delay

Often, the transfer function is given to us only as a table of values for different frequencies, or as a black-box procedure which returns $H(f)$ for a given f. To estimate the group delay we have to discretize the derivative in (225). Let $h > 0$ be small number. From Taylor's expansion of $\Phi(f \pm h)$, we get

95 ◁

$$\begin{aligned} A :=& \frac{\Phi(f+h)-\Phi(f)}{h} = \Phi'(f) + \frac{\Phi''(f)h}{2} + \frac{\Phi'''(f)h^2}{6} + \mathcal{O}(h^3) \\ B :=& \frac{\Phi(f-h)-\Phi(f)}{-h} = \Phi'(f) - \frac{\Phi''(f)h}{2} + \frac{\Phi'''(f)h^2}{6} - \mathcal{O}(h^3) \\ \frac{A+B}{2} =& \frac{\Phi(f+h)-\Phi(f-h)}{2h} = \Phi'(f) + \mathcal{O}(h^2) \end{aligned} \tag{233}$$

Which leads to

$$\delta(f) \approx \operatorname{Re}\left(\frac{\Phi(f-h)-\Phi(f+h)}{4h\pi i\Phi(f)}\right) \tag{234}$$

Most of the group (or wave) delay plots in this book were generated using this formula.

4.6.7 Note on Decibels

Since decibels are prone to be misunderstood, they deserve explanation. First, decibel is not a unit per-se, it is merely a way of expressing ratios. In this respect, it is similar to percent. To express how $P > 0$ compares to $P_0 > 0$ percentually, we compute $100P/P_0$. In case of decibels the formula is

$$P_{dB} = 10\log_{10}\left(\frac{P}{P_0}\right) \tag{235}$$

There is one important difference, thought. The definition of decibel depends on physical meaning of P. To be used correctly, it requires P to represent power or energy.

Amplitude-like quantities A and A_0, such as voltage or air pressure, can be also expressed in decibels, provided that they have been converted to energy units first. As $P = kA^2$, with k being suitable physical constant (such as conductivity in case of A being a voltage) we obtain the following formula for amplitudes

$$A_{dB} = 10\log_{10}\left(\frac{kA^2}{kA_0^2}\right) = 20\log_{10}\left(\frac{A}{A_0}\right) \tag{236}$$

which is similar to (235), except for the factor 20 instead of 10. This is probably the main cause of confusion with decibels — correct formula depends on what was measured. But once we have the number, we can forget whether it was computed from power or amplitude, because both ways lead to identical answer.

In the basic case, the number P refers to the whole signal, representing its average power (although it would be possible to compute decibels even from instantaneous power P_n at time n, it rarely makes sense). For a finite signal of length T the average power is $P = T^{-1}\sum_{n=0}^{T-1} P_n$. As discussed earlier, the instantaneous power relates to instantaneous amplitude as $P_n = kA_n^2$, dictating value of A in (236) to be

$$A = \sqrt{P/k} = \sqrt{\frac{1}{kT}\sum_{n=0}^{T-1} kA_n^2} = \sqrt{\frac{1}{T}\sum_{n=0}^{T-1} A_n^2} \tag{237}$$

This is called *RMS (Root Mean Square)* value of signal A_n. For a sine wave $A_n := A_P \sin\frac{2\pi kt}{T}$, we have $A = A_{rms} = A_P/\sqrt{2}$.

Most often, decibels are used in spectral and transfer function plots. Informally, we can imagine that we have filtered the signal into many narrow frequency-bands and determined decibel value for each of them, plotting the results. We have already used H_{dB} plot (222) where H_0 was 1, assuring transfer of 0 dB for filter doing nothing.

▷ 94

There are two ways of plotting dB-spectrum of the signal. Either we can plot energy spectrum, that is, for a signal s, we plot

$$E_{dB}(f) = 10\log_{10}\left(\left|\mathcal{F}(s)(f)\right|^2/E_0\right) = 20\log_{10}\left(\left|\mathcal{F}(s)(f)\right|/\sqrt{E_0}\right) \tag{238}$$

Or we can plot power spectrum, averaged over time T:

$$P_{dB}(f) = 10\log_{10}\left(\frac{\left|\mathcal{F}(s)(f)\right|^2}{TP_0}\right) \tag{239}$$

where the signal s has been zeroed outside $[0, T-1]$ interval, in the last formula. The energy spectrum allows us to see H_{dB} from a different angle. Provided that s was such that $\forall f : \mathcal{F}(s)(f) \neq 0$, we can compare the filter's output $h * s$ with its input s as follows

$$10\log_{10}\left(\frac{\left|\mathcal{F}(h*s)(f)\right|^2}{\left|\mathcal{F}(s)(f)\right|^2}\right) = 20\log_{10}\left(\frac{\left|\mathcal{F}(h)(f)\right|\cdot\left|\mathcal{F}(s)(f)\right|}{\left|\mathcal{F}(s)(f)\right|}\right) = 20\log_{10}\left|H(f)\right| \tag{240}$$

So we obtained H_{dB} consistent with (222) as expected. Note that s was the reference signal, therefore $|\mathcal{F}(s)(f)|^2$ played the role of $E_0(f)$ in the above formula. Due to properties of logarithm $H_{dB}(f)$ can also be computed as $E_{dB}(h*s)(f) - E_{dB}(s)(f)$.

Another trouble with decibels is with the 0 dB reference level P_0. As we have just seen, we can use decibels either to compare two signals or to compare single signal with a fixed standard level. The first use is dimensionless, while the second one is dimensional, in which case the unit should be written after dB, as as in case of dBmW, meaning that $P_0 = 1$ mW.

Unfortunately this is often violated if decibels are used to measure 'volume' of the sound, as majority of authors just write dB without proper unit and 0 dB level. Note that there are at least two quantities that could be understood as a volume. Either we can measure *dynamic pressure* (that is the fluctuations of absolute pressure around its mean value), or we can measure power density of the wave, also known as the *intensity*, which is the energy of the wave escaping thru a unit area perpendicular to the wave's travel direction.

In case of dynamic pressure, we use dBSPL, which compares sound's pressure with standard pressure level p_0 of 20 μPa (rms). For power density, we use dBSIL, comparing sound's intensity to $I_0 = 1$ pWm^{-2}. Note that (236) and (237) should be used for dBSPL and (235) for dBSIL. These two quantities are connected by so called *specific acoustic impedance* ζ in the following way.

$$p^2 = I\zeta \tag{241}$$

For the air at 20 °C, we have $\zeta = 413$ Pa·s/m, which gives $p_0 = \sqrt{\zeta I_0} = \sqrt{413 \cdot 10^{-12}} = 20.32\,\mu$Pa (rms). So both definitions lead to similar (but not exactly same, as ζ depends on temperature) numbers. Clearly, p_0 and I_0 were selected to match each other (under usual conditions) as well as the threshold of human hearing (for a 1 kHz wave).

In a free space, far enough from the source, the sound propagates as a spherical wave (possibly spanning only a sector of sphere if the source was directional). Energy conservation causes the intensity to fall like r^{-2} and so the pressure falls as r^{-1}, where r is the distance from the source. This means that the dB-volume at the distance r is $\kappa + 10\log_{10}(I) - 20\log_{10}(r)$, where κ is a constant, dependent on I_0 and units in which we measure the distance r. In reality, the volume goes-off faster because the temperature gradient[23] causes the sound to be refracted upwards.

4.7 Minimum Phase Filters

4.7.1 Definition *Linear Phase LTI System*

LTI system with $\varphi(f) = 2\pi\beta f$, for $\beta \in \mathbb{R}$ will be called *linear phase*.

4.7.2 Definition *Minimum Phase Filter*

A zp-canceled filter with all its zeroes and poles inside the unit circle will be called a *minimum phase filter*. This is essentially a misnomer. Correct name should be *least group delay filter* because the group delay is minimized, as will be apparent from the theorems below. Note that the definition disallows anything to lie at the unit circle.

4.7.3 Note Accordingly, *maximum phase filter* is defined to have all its poles and zeroes outside the unit circle. Sometimes, the phrase *mixed phase filter* is used to refer to the filter which is neither minimum nor maximum phase, having its zeroes and poles anywhere. Note that minimum as well as maximum phase filters are invertible.

In this section we will ask how many different phase responses a zp-canceled filter can have for a fixed magnitude response. We will find that there is only finitely many

[23] Normally, the temperature falls with the height and so does the speed of sound.

of them. It will also be shown that if the filter was *minimum phase*, its group delay will be least possible for a given magnitude response. As a byproduct, one-to-one correspondence between magnitude response shape and group delay will be established. In the following, let us consider a zp-canceled filter with N zeroes $z_k = Z_k e^{i\zeta_k}$ and M poles $p_k = P_k e^{i\lambda_k}$, where $Z_k, P_k \in \mathbb{R}^+$. Note that its $H(f)$ is a smooth function because M and N are finite and there is no $P_k = 1$.

4.7.4 Observation *For the above zp-canceled filter, a quantity proportional to its magnitude can be written in a following way*

$$\frac{\log(10)}{10} H_{dB}(f) = \log|H(f)|^2 = 2\log|c_0| + \\ + \sum_{k=1}^{N} \log(1 + Z_k^2 - 2Z_k\cos(2\pi f - \zeta_k)) - \sum_{k=1}^{M} \log(1 + P_k^2 - 2P_k\cos(2\pi f - \lambda_k)) \tag{242}$$

where c_0 is the amplification factor appearing in (210). ▷ 90

Proof It follows from (207), if we consider individual factors in this way ▷ 89

$$|e^{2\pi if} - z_k|^2 = (e^{2\pi if} - Z_k e^{i\zeta_k})\overline{(e^{2\pi if} - Z_k e^{i\zeta_k})} = (1 + Z_k^2 - 2Z_k\cos(2\pi f - \zeta_k)) \tag{243}$$

Q.E.D.

It is possible to get rid of the amplification factor by differentiating $\log|H(f)|^2$. The differentiated function will then represent the shape of magnitude response in the sense that all filters with the same shape of $|H(f)|$ but possibly different amplification will be assigned an identical representing function.

4.7.5 Lemma *The following function represents the shape of $|H(f)|$ invariantly to amplification:*

$$\frac{\partial}{\partial f}\log|H(f)|^2 = \\ \sum_{k=1}^{N} \frac{2\pi\sin(2\pi f - \zeta_k)}{\frac{1}{2}\left(Z_k^{-1} + Z_k\right) - \cos(2\pi f - \zeta_k)} - \sum_{k=1}^{M} \frac{2\pi\sin(2\pi f - \lambda_k)}{\frac{1}{2}\left(P_k^{-1} + P_k\right) - \cos(2\pi f - \lambda_k)} \tag{244}$$

Proof Differentiating $\log(1 + Z_k^2 - 2Z_k\cos(2\pi f - \zeta_k))$ we get

$$\frac{1}{1 + Z_k^2 - 2Z_k\cos(2\pi f - \zeta_k)} 2Z_k\sin(2\pi f - \zeta_k)2\pi \tag{245}$$

which directly leads to terms of (244), since $Z_k \neq 0$. Q.E.D.

4.7.6 Note For an invertible filter, $\frac{1}{2}\left(Z_k^{-1} + Z_k\right) > 1$ because $Z_k \neq 1$ as the zeroes cannot appear on the unit circle. Hence, the denominators of (244) are always positive.

It follows from (244), that the magnitude's shape does not change when a pole or a zero is swapped around the unit circle. In other words (244) is invariant upon changing $Z_k \mapsto 1/Z_k$ (which translates into $z_k \mapsto 1/\overline{z_k}$ — compare it with time reversal (219)). Using this switching, it is possible to generate up to 2^{N+M} different zero/pole arrangements, depending on multiplicity of poles and zeroes[24]. It will be shown in the sequel, that there are no other sets of zeroes and poles (even if more zeroes and poles would be used) leading to identical shape of $|H(f)|$. ▷ 93

[24] Multiplicity is to be counted when everything was switched inside the circle.

4.7.7 Lemma *The group delay of invertible zp-canceled filter with N zeroes and M poles can be expressed the following way*

$$\delta(f) = -\frac{\partial}{\partial f}\frac{\operatorname{Arg}(H(f))}{2\pi} = \sum_{k=1}^{N}\frac{\frac{1}{2}(\cos(2\pi f-\zeta_k)-Z_k^{-1})}{\frac{1}{2}\left(Z_k^{-1}+Z_k\right)-\cos(2\pi f-\zeta_k)} - \sum_{k=1}^{M}\frac{\frac{1}{2}(\cos(2\pi f-\lambda_k)-P_k^{-1})}{\frac{1}{2}\left(P_k^{-1}+P_k\right)-\cos(2\pi f-\lambda_k)} \tag{246}$$

89 ◁ **Proof** Rewriting factors of (207) as $e^{2\pi if}-z_k = z_k\left(e^{2\pi if}z_k^{-1}-1\right)$ we get

$$-2\pi\delta(f) = (\operatorname{Arg}(H(f)))' = \frac{\partial}{\partial f}\left(\operatorname{Arg}\left(c_0\frac{\prod z_k}{\prod p_k}\right)+\sum_{k=1}^{N}\operatorname{Arg}\left(e^{2\pi if}z_k^{-1}-1\right)-\sum_{k=1}^{M}\operatorname{Arg}\left(e^{2\pi if}p_k^{-1}-1\right)\right) \tag{247}$$

Where each occurrence of Arg is meant to be a continuous function of f. Differentiating $\operatorname{Arg}\left(e^{2\pi if}z_0^{-1}-1\right)$ we obtain:

$$\begin{aligned}&\frac{\partial}{\partial f}\operatorname{Arg}\left(e^{2\pi if-i\zeta_0}Z_0^{-1}-1\right) = \left(\operatorname{atg}\frac{\sin(2\pi f-\zeta_0)}{\cos(2\pi f-\zeta_0)-Z_0}\right)' = \\ &\frac{1}{1+\left(\frac{\sin(2\pi f-\zeta_0)}{\cos(2\pi f-\zeta_0)-Z_0}\right)^2}\cdot\frac{\cos^2(2\pi f-\zeta_0)-Z_0\cos(2\pi f-\zeta_0)+\sin^2(2\pi f-\zeta_0)}{\left(\cos(2\pi f-\zeta_0)-Z_0\right)^2}\cdot 2\pi \\ &= \frac{2\pi(1-Z_0\cos(2\pi f-\zeta_0))}{\sin^2(2\pi f-\zeta_0)+\left(\cos(2\pi f-\zeta_0)-Z_0\right)^2} = \frac{2\pi(1-Z_0\cos(2\pi f-\zeta_0))}{1+Z_0^2-2Z_0\cos(2\pi f-\zeta_0)} \\ &= \frac{-\pi\left(\cos(2\pi f-\zeta_0)-Z_0^{-1}\right)}{\frac{1}{2}\left(Z_0^{-1}+Z_0\right)-\cos(2\pi f-\zeta_0)}\end{aligned} \tag{248}$$

As $H(f)$ was smooth we could have ignored undefined points of atg's argument, still
95 ◁ getting the correct answer, as discussed in 4.6.2, Note that the limit used at 4.6.2 disappeared because the last formula is defined and smooth everywhere. Q.E.D.

4.7.8 Definition For $Q := \frac{1}{2}\left(Z^{-1}+Z\right) > 1$, let's define the following 'signals'.

$$ib := \mathcal{F}^{-1}\left(\frac{\sin(2\pi f)}{Q-\cos(2\pi f)}\right) \qquad a := \mathcal{F}^{-1}\left(\frac{\cos(2\pi f)}{Q-\cos(2\pi f)}\right) \qquad c := \mathcal{F}^{-1}\left(\frac{1}{Q-\cos(2\pi f)}\right) \tag{249}$$

As $\mathcal{F}^{-1}$ of an even or odd function will only contain the cosine or sine terms, respectively, we can write p-th element of the above signals as

$$b_p = \int_{-1/2}^{1/2}\frac{\sin(2\pi f)\sin(2\pi pf)}{Q-\cos(2\pi f)}df \qquad a_p = \int_{-1/2}^{1/2}\frac{\cos(2\pi f)\cos(2\pi pf)}{Q-\cos(2\pi f)}df \qquad c_p = \int_{-1/2}^{1/2}\frac{\cos(2\pi pf)}{Q-\cos(2\pi f)}df \tag{250}$$

Due to trigonometrical identity $\cos(\alpha\pm\beta) = \cos(\alpha)\cos(\beta)\mp\sin(\alpha)\sin(\beta)$ we have

$$\begin{aligned}a_p+b_p &= \int_{-1/2}^{1/2}\frac{\cos(2\pi pf-2\pi f)}{Q-\cos(2\pi f)}df = c_{p-1} \\ a_p-b_p &= \int_{-1/2}^{1/2}\frac{\cos(2\pi pf+2\pi f)}{Q-\cos(2\pi f)}df = c_{p+1}\end{aligned} \tag{251}$$

Hence

$$a_p = \frac{c_{p-1}+c_{p+1}}{2} \qquad b_p = \frac{c_{p-1}-c_{p+1}}{2} \tag{252}$$

Recovering the complex exponential in the integral for c_p, we can see that

$$c_p = \int_{-1/2}^{1/2}\frac{e^{2\pi ifp}}{Q-\cos(2\pi f)}df = \int_{-1/2}^{1/2}\frac{e^{2\pi ifp}}{Q-\frac{e^{2\pi if}}{2}-\frac{e^{-2\pi if}}{2}}df \tag{253}$$

Using z as an abbreviation for $e^{2\pi if}$ we can write

$$c_p = 2\int_{-1/2}^{1/2}\frac{z^p}{2Q-z-z^{-1}}df = 2\int_{-1/2}^{1/2}\frac{z^{p+1}}{2Qz-z^2-1}df \tag{254}$$

which could subsequently be expressed using complex path integral[25] in a following way:

$$c_p = -\frac{1}{\pi i}\int_{\substack{t\in[0,1)\\ z=e^{2\pi it}}}\frac{z^p}{z^2-2Qz+1}dz \tag{256}$$

The denominator can be written as $(z-z_A)(z-z_B)$, where $z_A = Q-\sqrt{Q^2-1}$ and $z_B = Q+\sqrt{Q^2-1} > 1$. It is an easy exercise to show that $z_A < 1$ whenever $Q > 1$. So z_B is outside the encircled area, while z_A is inside. Employing Cauchy's integration formula for $p \geq 0$, we get

$$c_p = -2\frac{1}{2\pi i}\int_{\substack{t\in[0,1)\\ z=e^{2\pi it}}}\frac{z^p/(z-z_B)}{z-z_A}\,dz = -2\frac{z_A^p}{z_A-z_B} = \frac{\left(Q-\sqrt{Q^2-1}\right)^p}{\sqrt{Q^2-1}} \tag{257}$$

[25] Recall that for $\varphi:[a,b]\to\mathbb{C}$ being a representation of oriented curve C, the *complex path integral* is defined the following way.

$$\int_C f(z)\,dz := \int_a^b f(\varphi(t))\cdot\varphi'(t)\,dt \tag{255}$$

Note that its value does not depend on particular choice of φ, as long as it represents the same curve C. In our case, C is a unit circle around the origin, that is $\varphi(t) = e^{2\pi it} = z$, hence $\varphi'(t) = 2\pi ie^{2\pi it} = 2\pi iz$.

For $p<0$, we have $c_p=c_{-p}$ because c_ps are Fourier coefficients of even function. Using abbreviation $\mu_Q:=z_A=Q-\sqrt{Q^2-1}=\min(Z,Z^{-1})<1$, we get for $p>0$

$$\begin{aligned}a_p&=\frac{c_{p-1}+c_{p+1}}{2}=\frac{\mu_Q^{p-1}}{2\sqrt{Q^2-1}}\left(1+Q^2+Q^2-1-2Q\sqrt{Q^2-1}\right)\\ b_p&=\frac{c_{p-1}-c_{p+1}}{2}=\frac{\mu_Q^{p-1}}{2\sqrt{Q^2-1}}\left(1-Q^2-Q^2+1+2Q\sqrt{Q^2-1}\right)\end{aligned}\tag{258}$$

Hence for $p>0$, the following holds:

$$\begin{aligned}a_p&=\frac{\mu_Q^{p-1}}{\sqrt{Q^2-1}}Q\left(Q-\sqrt{Q^2-1}\right)=\frac{Q\,\mu_Q^p}{\sqrt{Q^2-1}}\qquad c_p=\frac{\mu_Q^p}{\sqrt{Q^2-1}}\\ b_p&=\frac{\mu_Q^{p-1}}{\sqrt{Q^2-1}}\left(1-Q^2+Q\sqrt{Q^2-1}\right)=\left(-\sqrt{Q^2-1}+Q\right)\mu_Q^{p-1}=\mu_Q^p\end{aligned}\tag{259}$$

Due to symmetry/antisymmetry we have $a_p=a_{-p}$, $a_0=c_1=Q/\sqrt{Q^2-1}-1$ and
102◁ $b_p=-b_{-p}$. Now, we can determine $\mathcal{F}^{-1}$ of the terms appearing in formula (244). Let

$$U\left(f-\frac{\zeta_k}{2\pi}\right):=\frac{2\pi\sin(2\pi f-\zeta_k)}{\frac{1}{2}\left(Z_k^{-1}+Z_k\right)-\cos(2\pi f-\zeta_k)}\tag{260}$$

and let us temporarily assign $\frac{1}{2}\left(Z_k^{-1}+Z_k\right)$ to Q appearing in c_p, referring to it as to Q_k in the following formula, keeping in mind that $\mu_{Q_k}=\min(Z_k,Z_k^{-1})$.

$$\begin{aligned}y(k)_p:=\mathcal{F}^{-1}\left(\frac{2\pi\sin(2\pi f-\zeta_k)}{\frac{1}{2}\left(Z_k^{-1}+Z_k\right)-\cos(2\pi f-\zeta_k)}\right)_p&=\int_{-1/2}^{1/2}U\left(f-\frac{\zeta_k}{2\pi}\right)e^{2\pi ifp}df\\ =e^{ip\zeta_k}\int_{-1/2}^{1/2}U(f)e^{2\pi ifp}df=2\pi e^{ip\zeta_k}ib_p&=2\pi i\,\mathrm{sgn}(p)\left(e^{i\zeta_k\,\mathrm{sgn}(p)}\mu_{Q_k}\right)^{|p|}\end{aligned}\tag{261}$$

In a similar way, we can compute $\mathcal{F}^{-1}$ of terms of (246) for $p\neq 0$, getting:

$$\begin{aligned}x(k)_p:=\mathcal{F}^{-1}\left(\frac{\frac{1}{2}\left(\cos(2\pi f-\zeta_k)-Z_k^{-1}\right)}{\frac{1}{2}\left(Z_k^{-1}+Z_k\right)-\cos(2\pi f-\zeta_k)}\right)_p&=\frac{1}{2}e^{ip\zeta_k}\left(a_p-Z_k^{-1}c_p\right)\\ =\frac{1}{2}e^{ip\zeta_k}\left(\frac{Q_k\mu_{Q_k}^{|p|}}{\sqrt{Q_k^2-1}}-\frac{Z_k^{-1}\mu_{Q_k}^{|p|}}{\sqrt{Q_k^2-1}}\right)&=\frac{1}{4}\frac{Z_k-Z_k^{-1}}{\sqrt{Q_k^2-1}}e^{ip\zeta_k}\mu_{Q_k}^{|p|}\\ =\frac{1}{2}\frac{Z_k-Z_k^{-1}}{\sqrt{Z_k^2+Z_k^{-2}+2-4}}e^{ip\zeta_k}\mu_{Q_k}^{|p|}=\frac{1}{2}\frac{Z_k-Z_k^{-1}}{\sqrt{\left(Z_k-Z_k^{-1}\right)^2}}e^{ip\zeta_k}\mu_{Q_k}^{|p|}&\\ =\frac{1}{2}\frac{Z_k-Z_k^{-1}}{|Z_k-Z_k^{-1}|}e^{ip\zeta_k}\mu_{Q_k}^{|p|}=\frac{\mathrm{sgn}(Z_k-1)}{2}\left(e^{i\zeta_k\,\mathrm{sgn}(p)}\mu_{Q_k}\right)^{|p|}&\end{aligned}\tag{262}$$

where for $p=0$ we obtain $x(k)_0=\frac{1}{2}\left(a_0-Z_k^{-1}c_0\right)=\frac{1}{2}\left(\frac{Q_k}{\sqrt{Q_k^2-1}}-1-\frac{Z_k^{-1}}{\sqrt{Q_k^2-1}}\right)=\frac{1}{2}\big(\mathrm{sgn}(Z_k-1)-1\big)$. Now we have a technology to prove main results of this section.

4.7.9 Theorem *(Uniqueness of Shape Parameters) For a fixed magnitude response shape (244) of a zp-canceled filter there exists at most one pair of parameter multisets (Q_k,ζ_k) and (W_k,λ_k) such that these parameters generate the given shape. Here, $Q_k:=\frac{1}{2}\left(Z_k^{-1}+Z_k\right)$ and $W_k:=\frac{1}{2}\left(P_k^{-1}+P_k\right)$ for a filter with poles $p_k=P_k\exp(i\lambda_k)$ and zeroes $z_k=Z_k\exp(i\zeta_k)$, s.t. $P_k,Z_k>0$ and $P_k\neq 1$.*

Proof First, let us consider zeroes on the unit circle. Clearly, neither poles nor zeroes off the circle can make $|H(f)|$ zero for some f. Moreover, $\{f\in\mathbb{R}\mid|H(f)|=0\}$ is unique to the shape, too. Finally, the multiplicity of these roots can be decoded from $|H(f)|$ in their closest neighborhood. Therefore the zeroes on the unit circle are unique to the shape and we can disregard them, considering only invertible filters herefrom.

Let us suppose, that there exist a shape (of an invertible filter) which could be generated by two different multiset pairs $\langle(Q_k,\zeta_k),(W_l,\lambda_l)\rangle$ and $\langle(\widehat{Q}_k,\widehat{\zeta}_k),(\widehat{W}_l,\widehat{\lambda}_l)\rangle$. From all such counter-examples, select one with the smallest size of $(\widehat{Q}_k,\widehat{\zeta}_k)$ plus size of $(\widehat{W}_l,\widehat{\lambda}_l)$. Hence for any shape with smaller description, the theorem would already hold. Note that the selected shape's description is non-empty, since empty multiset describes frequency response[4] $H=\lambda f.1$, for which it is unique. ▷42

I will show that the pairs constituting the smallest counter-example must always contain common element, which would be a contradiction because this element could be removed from both descriptions, leading to even smaller counter-example. Hence the theorem would hold, after demonstrating the existence of the common element. To do this, let us apply $\mathcal{F}^{-1}$ on (244). From (261) we have the following for $p>0$:

$$\mathcal{F}^{-1}\left(\left(\log|H(f)|^2\right)'\right)_p=2\pi i\left(\sum_{k=1}^{N}\left(e^{i\zeta_k}\mu_{Q_k}\right)^p-\sum_{k=1}^{M}\left(e^{i\lambda_k}\mu_{W_k}\right)^p\right)\tag{263}$$

For two possible shape's descriptions $((Q_k,\zeta_k),(W_l,\lambda_l))$ and $((\widehat{Q}_k,\widehat{\zeta}_k),(\widehat{W}_l,\widehat{\lambda}_l))$ the following must hold

$$\underbrace{\sum_{k=1}^{\widehat{N}}\left(e^{i\widehat{\zeta}_k}\mu_{\widehat{Q}_k}\right)^p-\sum_{k=1}^{\widehat{M}}\left(e^{i\widehat{\lambda}_k}\mu_{\widehat{W}_k}\right)^p}_{\widehat{r}_p}=\underbrace{\sum_{k=1}^{N}\left(e^{i\zeta_k}\mu_{Q_k}\right)^p-\sum_{k=1}^{M}\left(e^{i\lambda_k}\mu_{W_k}\right)^p}_{r_p}\tag{264}$$

for any $p>0$ because both sides of the equation describe identical shape. Without the loss of generality, let us assume that

$$\mu_{Q_1}\geq\max(\max_k(\mu_{Q_k},\mu_{\widehat{Q}_k}),\max_k(\mu_{W_k},\mu_{\widehat{W}_k}))\tag{265}$$

Note that $\mu_{Q_1}<1$ because $Q_1>1$. Then we can explore the following expression:

$$\begin{aligned}Y(r):=\lim_{n\to\infty}\frac{1}{n}\sum_{p=1}^{n}\left(e^{i\zeta_1}\mu_{Q_1}\right)^{-p}r_p=&\\ \sum_{k=1}^{N}\lim_{n\to\infty}\frac{1}{n}\sum_{p=1}^{n}\left(e^{i\zeta_k-i\zeta_1}\frac{\mu_{Q_k}}{\mu_{Q_1}}\right)^p-\sum_{k=1}^{M}\lim_{n\to\infty}\frac{1}{n}\sum_{p=1}^{n}\left(e^{i\lambda_k-i\zeta_1}\frac{\mu_{W_k}}{\mu_{Q_1}}\right)^p&\end{aligned}\tag{266}$$

Note that the absolute value of the number being powered to p is ≤ 1 in all cases. When $\mu_x < \mu_{Q_1}$ we have a falling exponential and the limit is 0, due to $1/n$ factor. For $\mu_x = \mu_{Q_1}$, the result depends on phase angles. For $\lambda_k - \zeta_1 = 0 \bmod 2\pi$, we sum all ones, getting n for the sum and 1 for the limit. Otherwise we get rotating complex number, giving a bounded sum, which goes to 0 in the limit, due to the $1/n$ factor.

To summarize the previous paragraph, we can say that $Y(\widehat{r})$ is 1 iff the formula $\widehat{r}_p$ of (264) contained (Q_1, ζ_1) pair, and 0 otherwise. As $\widehat{r}_p = r_p$ for all p and $Y(r) = 1$, $Y(\widehat{r})$ must also be 1, which means that the second multiset of parameters must have contained (Q_1, ζ_1) as some $(\widehat{Q}_i, \widehat{\zeta}_i)$. Q.E.D.

4.7.10 Consequence *All the responses $H(f)$ of the same magnitude shape can be generated using $Z_k \mapsto 1/Z_k$ and $P_k \mapsto 1/P_k$ switching, in case of zp-canceled filter.*

Proof From previous theorem we know that for a fixed magnitude shape $H'_{dB}(f)$, there is uniquely determined multiset of parameters. The theorem follows from the fact that $Q = \frac{1}{2}\left(Z^{-1} + Z\right)$ has only two solutions $Z_{1,2} = Q \pm \sqrt{Q^2 - 1}$, $Z_1 = Z_2^{-1}$. Q.E.D.

4.7.11 Consequence *Invertible minimum-phase zp-canceled filter has the least possible group delay (at all frequencies) among all invertible one-way zp-canceled filters of the same $H_{dB}(f)$ shape.*

Proof Minimum phase filter has all its poles already inside the unit circle. Therefore, we can only switch its zeroes to generate all possible phase responses belonging to the given magnitude response, in accordance with 4.7.10. From (246) we can see that $Z_k^{-1} > 1$ gives lower $\delta(f)$ than $Z_k^{-1} < 1$, independently of other Z_ks. Therefore, switching all of them inside, we get smallest possible $\delta(f)$. Q.E.D.

103 ◁

According to the paragraph after (262), an average group delay $\delta_{AVG} := \int_0^1 \delta(f)\, df$ of a zp-canceled filter is equal to $\mathcal{F}^{-1}(\delta)_0 = \sum_k x_Z(k)_0 - \sum_k x_P(k)_0$, which reads as follows, when expanded.

$$\delta_{AVG} = \sum_{k=1}^{N} \frac{\operatorname{sgn}(Z_k - 1) - 1}{2} - \sum_{k=1}^{M} \frac{\operatorname{sgn}(P_k - 1) - 1}{2} \tag{267}$$

For linear phase filter, we have $\delta(f) = \delta_{AVG}$. From (267) it is then clear that $\delta(f) = \text{const} \in \mathbb{Z} \cap [-N, M]$. Therefore, it is not possible to delay the signal (at all of its frequencies) by a fraction of sampling period, using a digital filter. This strongly contrasts with general LTI systems with impulse response $h \in \sigma$, where any $\delta \in \mathcal{S}$ is attainable (consider $h := \mathcal{F}^{-1}(\lambda f.e^{-2\pi i \delta(f)})$). Unfortunately, these LTI systems cannot be expressed as digital filters with finite N and M as (267) shows. This does not mean that they are completely unrealizable, though. For instance for a finite input x we would only need finite portion of h when evaluating the convolution $x * h$, but the work required to obtain one output sample would depend on the length of x making such approach impractical. For this reason we are usually forced to get by with an approximation by a digital filter, fitting the linear phase in the frequency range which we are interested in, letting it free elsewhere. Technically, this approximation is not a linear phase filter but practically it behaves close to it.

Note that a zp-canceled minimum phase filter has $\delta_{AVG} = M - N$. On the other hand, δ_{AVG} of a linear phase filter is an index around which its impulse response is symmetric, i.e. $h_{\delta_{AVG}-k} = h_{\delta_{AVG}+k}$. Often we encounter filters with peak value of $|h|$ near δ_{AVG} as practical filters tend to be well localized in time. These observations justify the following definition.

4.7.12 Definition *Centered Digital Filter*

A filter obtainable from invertible zp-canceled filter by shifting its impulse response by δ_{AVG} to the left will be called a *centered filter*. Note that the centered filter's δ_{AVG} is zero then. We regard the number of its poles M and the number of its zeroes N to be the same as of the original zp-canceled filter. That means, we do not count the extra poles or zeroes (lying in $0 \in \mathbb{C}$) that only shift the impulse response in time.

It follows from (209) and (267) that for the amplification factor c_0 and the list of poles $p_k = P_k e^{i\lambda_k}$ and zeroes $z_k = Z_k e^{i\zeta_k}$ translated into c and d, such that $c_0 z^N + c_1 z^{N-1} + \cdots + c_N = c_0 \prod_{k=1}^{N}(z - z_k)$ and $z^M + d_1 z^{M-1} + \cdots + d_{M-1} z + d_M = \prod_{k=1}^{M}(z - p_k)$, the following equality holds for the centered filter (and in case of one-way filter it can be directly used as a recurrence which implements it).

▷ 90

$$y_n = \sum_{k=0}^{N} c_k x_{n-k+S} \; - \sum_{k=1}^{M} d_k y_{n-k} \tag{268}$$

where $S = N - M + \sum_{k=1}^{N} \frac{\operatorname{sgn}(Z_k - 1) - 1}{2} - \sum_{k=1}^{M} \frac{\operatorname{sgn}(P_k - 1) - 1}{2} = \sum_{k=1}^{N} \frac{1 + \operatorname{sgn}(Z_k - 1)}{2} - \sum_{k=1}^{M} \frac{1 + \operatorname{sgn}(P_k - 1)}{2}$. Since $Z_k \neq 1$, it equals to $\#\{k \mid Z_k > 1\} - \#\{k \mid P_k > 1\}$. According to (244), (246) and (267), the magnitude and the group delay of this filter satisfies the following relations, where H_M is conveniently scaled version of H_{dB}.

▷ 102

$$\frac{\partial}{\partial f} H_M(f) := \frac{\partial}{\partial f} \frac{\log(10)}{40\pi} H_{dB}(f) = \frac{\partial}{\partial f}\left(\frac{1}{4\pi} \log |H(f)|^2\right) = \sum_{k=1}^{N} \frac{\frac{1}{2}\sin(2\pi f - \zeta_k)}{\frac{1}{2}\left(Z_k^{-1} + Z_k\right) - \cos(2\pi f - \zeta_k)} - \sum_{k=1}^{M} \frac{\frac{1}{2}\sin(2\pi f - \lambda_k)}{\frac{1}{2}\left(P_k^{-1} + P_k\right) - \cos(2\pi f - \lambda_k)} \tag{269}$$

$$\delta(f) = \sum_{k=1}^{N} \frac{\frac{1}{2}\left(\cos(2\pi f - \zeta_k) - \min(Z_k, Z_k^{-1})\right)}{\frac{1}{2}\left(Z_k^{-1} + Z_k\right) - \cos(2\pi f - \zeta_k)} \operatorname{sgn}(Z_k - 1) - \sum_{k=1}^{M} \frac{\frac{1}{2}\left(\cos(2\pi f - \lambda_k) - \min(P_k, P_k^{-1})\right)}{\frac{1}{2}\left(P_k^{-1} + P_k\right) - \cos(2\pi f - \lambda_k)} \operatorname{sgn}(P_k - 1) \tag{270}$$

The advantage of centering the filter can be clearly seen now. Although slightly longer than (244) and (246) the above formulas are more symmetric, namely the terms of (270) are either positive or negative, depending on whether the Z_k lied outside or inside the unit circle. Similarity of (269) and (270) suggests that there might be a connection between the magnitude and group delay. The following theorem shows that this is really the case.

4.7.13 Theorem *(Relationship of Group Delay and Magnitude) For a centered minimum-phase filter, the magnitude's shape $\partial H_M(f)\,/\,\partial f$ is in the following one-to-one correspondence with group delay $\delta(f)$.*

$$H'_M = \frac{\log(10)}{40\pi} H'_{dB} = \mathcal{F}\left(-i\,\mathrm{sgn}\odot\mathcal{F}^{-1}(\delta)\right) \tag{271}$$

or the other way round

$$\delta = \mathcal{F}\left(i\,\mathrm{sgn}\odot\mathcal{F}^{-1}(H'_M)\right) \tag{272}$$

Proof Fourier coefficients of individual terms of $\delta(f)$ of eq. (270) and $\left(\log|H(f)|^2\right)'$ of
105 ◁ eq. (269) were determined as $x(k)_p$ in (262) and $y(k)_p$ in (261), respectively. Summing
105 ◁ them together, we get $\mathcal{F}^{-1}(\delta)$ and $\mathcal{F}^{-1}(H'_M)$ as follows, where $\mu_{Q_k} := \min(Z_k, Z_k^{-1})$ and $\mu_{W_k} := \min(P_k, P_k^{-1})$.

$$\begin{aligned}\mathcal{F}^{-1}(\delta)_p &= \sum_{k=1}^{N} \frac{\mathrm{sgn}(Z_k-1)}{2} e^{i\zeta_k p}\mu_{Q_k}^{|p|} - \sum_{k=1}^{M} \frac{\mathrm{sgn}(P_k-1)}{2} e^{i\lambda_k p}\mu_{W_k}^{|p|} \\ \mathcal{F}^{-1}(H'_M)_p &= \frac{i\,\mathrm{sgn}(p)}{2}\left(\sum_{k=1}^{N} e^{i\zeta_k p}\mu_{Q_k}^{|p|} - \sum_{k=1}^{M} e^{i\lambda_k p}\mu_{W_k}^{|p|}\right)\end{aligned} \tag{273}$$

Note that $\mathcal{F}^{-1}(\delta)_0 = 0$ because of centering, therefore the first equation of (273) applies only to $p \neq 0$, whereas the second one holds for any $p \in \mathbb{Z}$. As the filter is minimum-phase we have $\mathrm{sgn}(Z_k - 1) = -1$, which implies that $\mathcal{F}^{-1}(\delta)_p = i\mathcal{F}^{-1}(H'_M)_p\,\mathrm{sgn}(p)$ for any $p \in \mathbb{Z}$. Q.E.D.

4.7.14 Note The theorem could be formulated for the maximum phase as well. It would come out as $\delta = \mathcal{F}\left(-i\,\mathrm{sgn}\odot\mathcal{F}^{-1}(H'_M)\right)$.

4.7.15 Lemma *For $C \in \mathcal{S}$ and $c := \mathcal{F}^{-1}(C)$ such that c_k is absolutely convergent (i.e. $\sum_{k=-\infty}^{\infty}|c_k|$ converges) and $c_0 = 0$, the integral $D(x) := \int_0^x C(f)\,df$ equals to $\mathcal{F}(d)(x)$, where*

$$d_k := \begin{cases} \sum_{n=1}^{\infty} \frac{1}{2\pi i n}(c_n - c_{-n}) & \text{for } k=0 \\ \frac{-1}{2\pi i k} c_k & \text{else} \end{cases} \tag{274}$$

78 ◁ **Proof** Expanding $D(x)$ using 4.2.7, we get

$$\int_0^x C(f)\,df = \int_0^x \lim_{N\to\infty} \sum_{k=-N}^{N} c_k e^{-2\pi i k f}\,df \tag{275}$$

As c_k is absolutely convergent, the partial sums converge uniformly towards the limit, thus we can swap the integral with the limit, getting

$$\sum_{k=-\infty}^{\infty} c_k \int_0^x e^{-2\pi i k f}\,df = \sum_{\substack{k=-\infty \\ k\neq 0}}^{\infty} c_k \frac{e^{-2\pi i k x} - e^0}{-2\pi i k} = \sum_{\substack{k=-\infty \\ k\neq 0}}^{\infty} \frac{-c_k}{2\pi i k} e^{-2\pi i k x} + \sum_{k=1}^{\infty} \frac{c_k - c_{-k}}{2\pi i k} \tag{276}$$

80 ◁ where we also used that $c_0 = 0$. The claim follows from 4.2.8. Q.E.D.

4.7.16 Corollary *Centered minimum-phase filter satisfies the following relation.*

$$\mathrm{sgn}^2 \odot \mathcal{F}^{-1}(\widetilde{\varphi}) = -i\,\mathrm{sgn}\odot\mathcal{F}^{-1}(H_L) \qquad \text{where} \qquad \widetilde{\varphi}(f) := -2\pi\int_0^f \delta(g)\,dg \tag{277}$$

and $H_L(f) := 2\pi H_M(f) = \log|H(f)|$.

Proof From (273) we can see that both $\mathcal{F}^{-1}(\delta)_k$ and $\mathcal{F}^{-1}(H'_M)_k$ are composed of $N+M$ falling exponentials (since $\mu_Q < 1$), therefore being absolutely convergent. Since it is additionally true that $\int_0^1 \delta(f)\,df = 0$ (because the filter is centered) and $\int_0^1 H'_M(f)\,df = H_M(1) - H_M(0) = 0$ (because H_M is 1-periodic), we can apply 4.7.15 to justify that the following holds for $k \neq 0$.

$$\mathcal{F}^{-1}(\delta)_k = ik\mathcal{F}^{-1}(\widetilde{\varphi})_k \quad \text{and} \quad \mathcal{F}^{-1}(H'_M)_k = -ik\mathcal{F}^{-1}\left(\lambda f.\int_0^f H'_L(g)\,dg\right)_k \tag{278}$$

Up to a constant the lambda term is equal to H_L. The constant would appear as $\mathcal{F}(H_L)_0$ but we don't need to care about it because $\mathrm{sgn}(0) = 0$. Using (278) in 4.7.13 we get what had to be proved. Q.E.D.

4.7.17 Note It is not necessarily true that $H_L = \mathcal{F}\left(i\,\mathrm{sgn}\odot\mathcal{F}^{-1}(\varphi)\right)$ or even $\varphi = \mathcal{F}\left(-i\,\mathrm{sgn}\odot\mathcal{F}^{-1}(H_L)\right)$ holds because the average level of H_L as well as of φ might not be zero. Both can be controlled by the 'gain' parameter c_0 of (210), where $|c_0|$ controls ▷ 90
the overall gain, i.e. the average level of H_L, whereas $\mathrm{Arg}(c_0)$ controls the phase delay, i.e. the average level of φ, where $\varphi(f) := \mathrm{Arg}_C(H)(f)$ is smooth in f. The average level of $\widetilde{\varphi}$, on the other hand, remains fixed (though generally non-zero in case of filters with complex h_k) because its definition in 4.7.16 requires $\widetilde{\varphi}(0) = 0$.

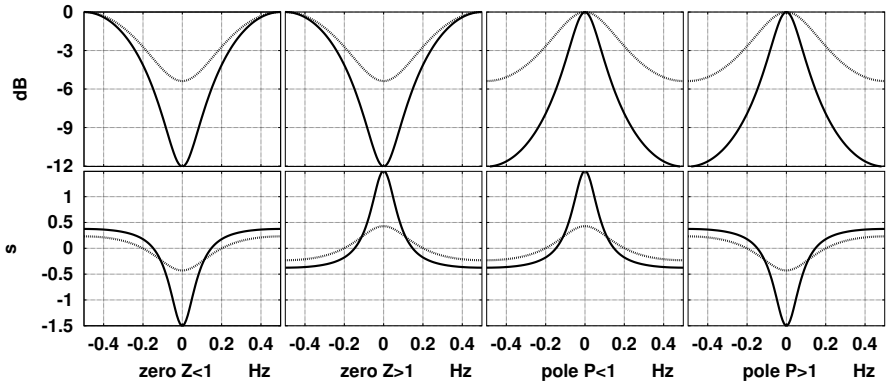


Fig. 15. *Magnitude (upper panels) and group delay (lower panels) of single zero or pole of a centered filter, plotted on normalized frequency scale, using formulas (279) and (270). From the left to the right* ▷ 111
there is a response of a zero lying inside the unit circle, followed by a response of the zero outside the circle, followed by the pole inside the circle, finally followed by the pole outside the circle. Each panel contains two plots, one for Z or P being 0.6 (black line), the other for 0.3 (grey line, actually a dashed one, for the hawk-eyed) or the reciprocal values thereof, for cases outside the circle. The resonant frequency ζ or λ is zero (nonzero values would only shift the plot sideways in a 1-periodic way). Notice that the group delay plot is slightly "narrower" than the magnitude plot that corresponds to it.

102 ◁ Returning back to (242), we can see that

$$\begin{aligned} H_L = &\frac{1}{2}\sum_{k=1}^{N}\log\left(\tfrac{1}{2}(Z_k+Z_k^{-1})-\cos(2\pi f-\zeta_k)\right)\\ &-\frac{1}{2}\sum_{k=1}^{M}\log\left(\tfrac{1}{2}(P_k+P_k^{-1})-\cos(2\pi f-\lambda_k)\right)\\ &+\log|c_0|+\frac{1}{2}\sum_{k=1}^{N}\log Z_k-\frac{1}{2}\sum_{k=1}^{M}\log P_k \end{aligned} \tag{279}$$

108 ◁ This formula, together with (270) allows us to think about responses of centered filters as of sums of the basic shapes plotted in fig. 15 (each shape corresponding to the zero or pole).

4.7.18 Note Centered linear phase filters have zero group delay, which implies that $\mathcal{F}(\delta)_p = 0$ for all p. This means that the terms of (273) must cancel each other. From 106 ◁ the proof of uniqueness 4.7.9, we know that only exactly matching parameters Q_k and ζ_k can cancel each other for all p. Moreover Q_k will not have a chance to cancel with W_k, as the filter was already zp-canceled before it has been centered. Hence, what remains is a zero $Ze^{i\zeta}$ phase-canceling with another zero $e^{i\zeta}/Z$ or a pole $Pe^{i\lambda}$ phase-canceling with another pole $e^{i\lambda}/P$. Therefore, any linear phase filter with non-constant $H(f)$ must be a mixed-phase filter.

4.8 Inverse of Minimum Phase Filter

Here, I will briefly mention practical way of computing inverse of digital filter without 86 ◁ using $\mathcal{F}$ (which the definition (192) suggests). Let the response h be minimum phase and causal and let $h_0 \neq 0$. Let the filter with such a response be determined by c and d such that:

$$y_n = \sum_{k=0}^{N} c_k x_{n-k} - \sum_{k=1}^{M} d_k y_{n-k} = \sum_{k=0}^{\infty} h_k x_{n-k} \tag{280}$$

Note that $c_0 \neq 0$ because $h_0 \neq 0$ and that we treat d_0 as 1. Rearranging it, we obtain

$$d_0 y_n + \sum_{k=1}^{M} d_k y_{n-k} = c_0\left(x_n + \sum_{k=1}^{N}\frac{c_k}{c_0}x_{n-k}\right) \tag{281}$$

which gives (y is input and x output here)

$$x_n = \sum_{k=0}^{M}\frac{d_k}{c_0}y_{n-k} - \sum_{k=1}^{N}\frac{c_k}{c_0}x_{n-k} \tag{282}$$

So, for causal, minimum phase $h = c * d^{-1}$ s.t. $h_0 \neq 0$ the inverse is $w = (d/c_0) * (c/c_0)^{-1} = d * c^{-1}$, which is not surprising. Note that w is also causal, minimum phase and $w_0 \neq 0$.

Noting that $h[A] * w[B] = (h * w)[A+B]$ we immediately se that $h[-R]$ (for $R \in \mathbb{N}$) is inverse to $w[R]$, which solves the case of causal minimum phase $\widehat{h}$ with zero $\widehat{h}_0$. Note that $\widehat{w} = w[R]$ is no longer causal then.

Now what if we only knew h. Could we still find a simple way of computing w? It turns out that is it possible if we also knew N and M. h can be obtained by feeding $x = \vec{1}$ into (280). Noting that the FIR becomes zero for $n > N$ we get the following set of equations.

$$-h_n = \sum_{k=1}^{M} d_k h_{n-k} \qquad \text{for } n > N \tag{283}$$

Taking first M equations and solving them, we obtain d, whereas $c = h * d$, provided that we defined $d_0 := 1$. Note that c_ks for $k > N$ should come out as zeroes. Now we can get $w = c^{-1} * d$ as before.

Unfortunately, I often found this method unstable for values of M above 50. Presumably, this is caused by inherent instability of direct implementation of the IIR part of the response. In 4.13.2, I will show IIR implementation that does not suffer from ▷ 140 this. Nevertheless, it cannot be easily applied to this case.

4.9 Response Decomposition Theorem

In the last section we learned how to decompose one-way filter into its zero and pole parts. Unfortunately, that would not work in general, as the IIR part might be two way (4.5.1). ▷ 92

This section addresses this problem theoretically, by providing a formula that, given an impulse response of invertible digital filter h returns impulse responses h_I and h_O corresponding to the poles and zeroes inside and outside the unit circle, respectively. So, h_I is one-way, ρh_O too and we could use the previous section the separate poles from zeroes.

The decomposition will be formulated using filter's frequency response $H := \mathcal{F}(h)$. Let us start by defining a pair of transformations that switch poles and zeroes inside the unit circle in two different ways. Henceforth, $\mathcal{S}_D \subseteq \mathcal{S}$ will denote the set of responses achievable by a digital filter, that is $\{(\lambda f.R(e^{2\pi i f})) \in \mathcal{S} \mid R \text{ is } \mathbb{C}\to\mathbb{C} \text{ rational function}\}$. Accordingly, we define $\sigma_D := \mathcal{F}^{-1}[\mathcal{S}_D]$.

4.9.1 Definition $\mathcal{I} : \mathcal{S}_D \to \mathcal{S}_D$ — *Direct Phase-Minimizing Conversion*
Let $H \in \mathcal{S}_D$. Due to the definition of $\mathcal{S}_D$, it can be written as

$$H(f) = c_0\frac{\prod_{k=1}^{N}\left(e^{2\pi i k f} - z_k\right)}{\prod_{k=1}^{M}\left(e^{2\pi i k f} - p_k\right)} \tag{284}$$

The *direct phase-minimizing* mapping $\mathcal{I}$ converts this H into minimum phase response by switching the poles and zeroes inside. Formally

$$\mathcal{I}(H) := \lambda f.\, c_0\frac{\prod_{k=1}^{N}\left(e^{2\pi i k f} - \min(Z_k, Z_k^{-1})e^{i\zeta_k}\right)}{\prod_{k=1}^{M}\left(e^{2\pi i k f} - \min(P_k, P_k^{-1})e^{i\lambda_k}\right)} \tag{285}$$

where $z_k = Z_k e^{i\zeta_k}$, $p_k = P_k e^{i\lambda_k}$ s.t. $Z_k \in [0,\infty)$ and $P_k \in [0,\infty)\backslash\{1\}$. The mapping can be naturally extended to $\sigma_D \to \sigma_D$ by defining $\mathcal{I}(h) := \mathcal{F}^{-1}(\mathcal{I}(\mathcal{F}(h)))$.

4.9.2 Definition $\mathcal{X} : \mathcal{S}_D \to \mathcal{S}_D$ — *Crossed Phase-Minimizing Conversion*

Using notation of the previous definition the *crossed phase-minimizing* mapping $\mathcal{X}$ converts H by swapping the outside poles with zeroes, switching them inside afterwards. 77◁ Formally (note that $P_k \neq 1$; you might also like to refresh the definition 4.2.2):

$$\mathcal{X}(H) := \lambda f.\, c_0 \frac{\prod_{k=1}^{N} \left(e^{2\pi i k f} - \min(Z_k, Z_k^{-1}) e^{i\zeta_k}\right)^{\frac{1}{2}\operatorname{sgn}^+(1-Z_k)}}{\prod_{k=1}^{M} \left(e^{2\pi i k f} - \min(P_k, P_k^{-1}) e^{i\lambda_k}\right)^{\frac{1}{2}\operatorname{sgn}(1-P_k)}} \tag{286}$$

Again, we can extend the mapping to σ_D.

The idea of computing $\mathcal{I}$ and $\mathcal{X}$ without identifying individual zeroes and poles 110◁ comes from the theorem 4.7.16 and fig. 15. We can notice in the figure that if we used only the magnitude information, reconstructing the phase as if all the poles and zeroes were inside, we would get a minimum phase filter with identical $|H(f)|$. This action is equivalent to switching the outer poles and zeroes inside the circle.

Analogically, if we retained only the the group delay and reconstructed the magnitude, we would obtain the behavior of $\mathcal{X}$ since forgetting the phase makes zero inside the circle indistinguishable from the pole outside of it (and vice versa). It therefore effectively cross-switches poles and zeroes inside.

As 4.7.16 requires a centered filter we have to adjust H before actually using it, taking the adjustments back after the 4.7.16 has been applied. Also, to make things easier, let us only consider invertible filter h now. The formula for computing $\mathcal{I}(H)$ for $H \in \mathcal{S}_D \cap \mathcal{S}_I$ is the following[26].

$$\begin{aligned}
\mathcal{I}(H) &:= \lambda f.\, |H(f)| \exp\left(i(\widehat{\varphi}(f) + \varphi_0 + \psi_0 f)\right) && \text{where} \\
\widehat{\varphi} &:= \mathcal{F}\left(-i \operatorname{sgn} \odot \mathcal{F}^{-1}(\lambda f.\, \log|H(f)|)\right) && \text{and} \\
\varphi(f) &:= \operatorname{Arg}_C(H(f)) \qquad \varphi_0 := \int_{-1/2}^{1/2} \varphi(f)\, df \qquad \psi_0 := \int_{-1/2}^{1/2} \varphi'(f)\, df = \varphi(\tfrac{1}{2}) - \varphi(-\tfrac{1}{2})
\end{aligned} \tag{287}$$

The adjustment first removed mean value of φ, calling it φ_0. Then it removed group delay (calling the quantity proportional to it ψ_0). Note that φ was smooth since we assumed no zeroes on the unit circle. Then, 4.7.16 was applied, giving out $\widehat{\varphi}$. In fact it could have been applied on an unadjusted $H(f)$ as well, since it only uses its magnitude which remains unaffected by phase adjustments. Note that the resulting $\widehat{\varphi}$ has zero mean and it represents centered filter (therefore $\varphi(1/2) = \varphi(-1/2)$).

So far we have divided the original filter into multiplication by $e^{i\varphi_0}$ (involves no poles nor zeroes) followed by a delay of $-\frac{1}{2\pi}\psi_0$ (involves poles or zeros in $0 \in \mathbb{C}$) followed by a centered filter (whose outer poles and zeroes are to be switched inside by 4.7.16). The adjustments effectively multiplied the original $H(f)$ by $e^{-(i\varphi_0 + i\psi_0 f)}$. Hence the result has to be multiplied by $e^{i\varphi_0 + i\psi_0 f}$ to undo these. As $e^{i\varphi_0 + i\psi_0 f}$ did not contain any poles or zeroes from outside of the circle we can see that (287) did what it was supposed to do — switched the outer poles and zeroes inside, leaving the inner poles and zeroes intact.

[26] The function $\operatorname{Arg}_C(H)$ turns H into φ (formally $\operatorname{Arg}_C : (\mathbb{R} \to \mathbb{C}) \to (\mathbb{R} \to \mathbb{R})$), making φ as continuous as possible (so in case of $H \in \mathcal{S}_D \cap \mathcal{S}_I$, φ will be continuous everywhere).

Similarly, $\mathcal{X}(H)$ for $H \in \mathcal{S}_D \cap \mathcal{S}_I$ can be computed as follows.

$$\begin{aligned}
\mathcal{X}(H) &:= \lambda f.\, \exp\left(\widehat{H}_L(f) + H_0\right) \frac{H(f)}{|H(f)|} && \text{where} \\
\widehat{H}_L &:= \mathcal{F}\left(i \operatorname{sgn} \odot \mathcal{F}^{-1}(\lambda f.\, (\varphi(f) - \psi_0 f))\right) && \text{and} \\
H_0 &:= \int_{-1/2}^{1/2} \log|H(f)|\, df \qquad \varphi(f) := \operatorname{Arg}_C(H(f)) \qquad \psi_0 := \varphi(\tfrac{1}{2}) - \varphi(-\tfrac{1}{2})
\end{aligned} \tag{288}$$

First, H is adjusted by removing mean phase φ_0 and mean group delay $-\frac{1}{2\pi}\psi_0$. Then, H_0 is computed, the average level of $|H(f)|$, which carries information about $|c_0|$ of (210). After that, 4.7.16 is applied on the adjusted (centered) phase. As $\operatorname{sgn}(0) = 0$, ▷90 the subtraction of φ_0 from $\varphi(f)$ may be omitted, when centering the phase. Since the computed $\widehat{H}_L$ is of zero mean we have to recover $|c_0|$. This can be done by adding H_0 to it because when $H_0 = \int \log|H(f)|\, df = \int H_L(f)\, df$ is expanded using (279) the only terms that change by cross-switching are those $\int \log P_k$ and $\int \log Z_k$ that correspond to the outer P_ks and Z_ks. The way they are switched inside changes P_k into $1/P_k$ but at the same time the new value is treated like a zero which changes the sign before the logarithm, which is why its contribution does not change at all. Analogically for Z_k. Therefore, H_0 is invariant under cross-switching and has to be added to zero-mean $\widehat{H}_L$ to recover proper cross-switched response.

4.9.3 Theorem *(Response Decomposition Theorem) Up to the amplification factor, $H \in \mathcal{S}_D \cap \mathcal{S}_I$ can be uniquely decomposed into $I \in \mathcal{S}_D \cap \mathcal{S}_I$ and $O \in \mathcal{S}_D \cap \mathcal{S}_I$ such that $I \cdot O = H$ and I has poles and zeroes only inside the unit circle and O only outside. The decomposition can be carried out by the following formulae.*

$$\begin{aligned}
I &:= \sqrt{\mathcal{I}(H) \cdot \mathcal{X}(H)} \\
O &:= H \cdot I^{-1}
\end{aligned} \tag{289}$$

where the complex square root is meant to be continuous, that is $\sqrt{z} := \sqrt{|z|} e^{\frac{i}{2}\operatorname{Arg}_C(z)}$.

Proof Considering that we are given $H = c_0 I_0 O_0$, where c_0 is the amplification factor, I_0 represents inner poles and zeroes (with unit amplification) and O_0 represents the outer ones, we have

$$\mathcal{I}(H) = c_0 I_0 \mathcal{I}(O_0) \qquad \mathcal{X}(H) = c_0 I_0 (\mathcal{I}(O_0))^{-1} \tag{290}$$

Hence, $\mathcal{I}(H)\mathcal{X}(H) = c_0^2 I_0^2$. Note that c_0 gets embedded into I, whereas the O's amplification factor will equal to 1. Q.E.D.

4.10 Sampling Theorem

So far, we have considered signals of very high (10^{50} Hz) sampling frequency as a way of representing analog (continuous-time) signals. At the same time we developed theory independent of actual sampling frequency. In any practical application the sampling frequency must be many orders of magnitude smaller. For instance by taking only every 10^{45}th value and forming a new signal from these values we would obtain signal with 100 kHz sampling rate, which is easily attainable with today's hardware. This process is called a *decimation*. Of course, the above procedure is meant only theoretically — we do not actually have 10^{50} samples per second which we could decimate. But we can imagine that all the physical processes in analog circuits before the A/D converter could be well described by LTI systems running at this high sampling frequency. This not only allows us to talk about sampling and reconstruction of analog signal but, more importantly, it makes possible to judge distortion caused by combined action of analog circuits and converters, once we work out all the details. The central question answered in this section is under what conditions we can reconstruct (at least theoretically) the original signal from its decimated version.

4.10.1 Aliasing

Decimation does not in general allow for reconstruction of the original signal because some information is lost. The mechanism of this loss is quite interesting when observed in the spectral domain. Suppose we have a sequence $a_n := \exp(2\pi i f_A n)$ and we are taking its every K-th sample to obtain $b_n := a_{Kn} = \exp(2\pi i f_A K n) = \exp(2\pi i f_B n)$, where $f_B := K f_A$. Due to periodicity of complex exponential the relationship between normalized frequencies of a and b can as well be written as $f_B = K f_A - \lfloor K f_A + 0.5 \rfloor$, as shown in fig. 16.

This effect can be seen in movies with scene of accelerating spoked-wheel carriage. For a wheel with s spokes and small speeds, such that the wheel revolves by less than π/s radians per a film frame we can observe its true rotation. As the speed increases it eventually exceeds π/s radians per frame, and it seems to start rotating backwards (because π/s corresponds to $1/2$ normalized frequency and we can see from fig. 16 that the frequency f_B became negative after reaching $1/2$). If the wheel accelerates even more it seems to slow down until its true speed reaches $2\pi/s$ per frame at which speed the wheel seems to be standing still. For even higher speeds it seems to accelerate again, and the whole process repeats in accordance with fig. 16.

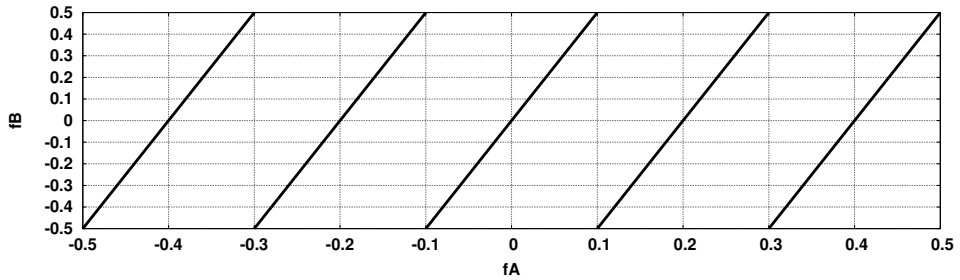


Fig. 16. *Frequency folding of $\mathcal{D}_5$ decimation.*

4.10.2 Definition *Decimation* Mapping $\mathcal{D}_n : \sigma \to \sigma$ defined the following way

$$\mathcal{D}_n(x)_k := x_{nk} \tag{291}$$

will be called a *decimation*. Often we drop index n if it follows from context and we also extend $\mathcal{D}_n$ to $\mathcal{S} \to \mathcal{S}$ in the following way.

$$\mathcal{D}_n(X) := \mathcal{F}\left(\mathcal{D}_n\left(\mathcal{F}^{-1}(X)\right)\right) \tag{292}$$

Note that decimation is linear operation (but not time invariant).

4.10.3 Observation *(on Folding) For $A \in \mathcal{S}$ and $B = \mathcal{D}_K(A)$ we have $B \in \mathcal{S}$ and*

$$B(f) = \frac{1}{K} \sum_{k=0}^{K-1} A\left(\frac{k+f}{K}\right) \tag{293}$$

Proof Although (293) is apparent from fig. 16, linearity of $\mathcal{D}$ and conservation of amplitude during decimation, I will provide formal proof for sake of completeness. From (292) we have

$$B(g) = \sum_{n=-\infty}^{\infty} \int_{-1/2}^{1/2} A(f) e^{2\pi i f n K}\, df e^{-2\pi i g n} = \lim_{N\to\infty} \int_0^1 A(f) \sum_{n=-N}^{N} e^{2\pi i n (Kf-g)}\, df \tag{294}$$

Using the change of variables, we obtain

$$B(g) = \lim_{N\to\infty} \frac{1}{K} \int_0^K A\left(\frac{u}{K}\right) D_N(u-g)\, du \tag{295}$$

where D_N is defined by (159). Defining 1-periodic functions $\widehat{A}_k(u) := A\left(\frac{k+u}{K}\right)$ for $u \in (0,1)$ and $\widehat{A}_k(0) := \frac{1}{2}\left(A^+(\frac{k}{K}) + A^-(\frac{k+1}{K})\right)$ we can write ▷ 78

$$B(g) = \frac{1}{K} \sum_{k=0}^{K-1} \lim_N \int_0^1 \widehat{A}_k(u) D_N(g-u)\, du = \sum_{k=0}^{K-1} \frac{A^+\left(\frac{k+g}{K}\right) + A^-\left(\frac{k+g}{K}\right)}{2K} \tag{296}$$

where the last equality comes from 4.2.10. For $g \in (0,1)$ it gives (293), since $A \in \mathcal{S}$. For $g = 0$ we also obtain (293) because $\frac{1}{K}\sum_{k=0}^{K-1} \widehat{A}_k(0) = \frac{1}{K}\sum_{k=0}^{K-1} A(\frac{k}{K})$. The obtained B is from $\mathcal{S}$ since $\mathcal{S}$ is closed on shifting and addition. Q.E.D. ▷ 80

Obviously, signal a can be recovered from the decimated signal $b := \mathcal{D}_K(a)$ iff its spectrum can be reconstructed. This in turn happens if and only if the frequencies f_A at which the source signal had non-zero power do not collide[27] after being mapped to f_B

[27] In fact there can be collisions if the set of points they form has a zero measure — such collisions cannot change integral inside $\mathcal{F}^{-1}$. Note that signals from σ are meant here, not everlasting sinusoids.

in accordance with fig. 16. The then sum (293) contains at most one non-zero number at each frequency, and as a result, no information is loss there (if we manage to assign correct alias band (of the K possibilities) for every frequency during reconstruction, which is easy in the usual case of contiguous frequency band of the original signal).

Most often, only the (normalized) frequencies in the interval $\left[-\frac{1}{2K}, \frac{1}{2K}\right)$ are allowed to be of non-zero power (but in radar, sonar or oscilloscope applications we may want to process the high frequency band instead). A signal with such a spectrum is said to be *band-limited*. Typically, the signal we measure is not band-limited and we have to
88 ◁ constrain it by filtering, ideally using a brick-wall filter (200). This filter or its practical approximation is often called an *antialiasing filter*. The reconstruction then concerns only the band in question — anything outside $\left[-\frac{1}{2K}, \frac{1}{2K}\right)$ that could have been present in the original signal will be lost. This kind of filtering followed by decimation is called a *downsampling*.

Reconstruction from a K-times downsampled signal b goes as follows. First we assemble the signal c from b by interleaving its samples with blocks of $K-1$ zeroes:

$$c_n := \begin{cases} b_{n/K} & \text{for } n\%K = 0 \\ 0 & \text{else} \end{cases} \tag{297}$$

Its spectrum $C(f) = \mathcal{F}(c)(f) = \sum_n b_n e^{-2\pi i n K f} = B(fK)$ is K-times compressed and periodically repeated copy of b's spectrum. Equation (293) expresses the relationship between B and A — the spectrum of original signal. As we assume that A was antialiased before the decimation took place, the eq. (293) actually reduces to $\frac{1}{K}A(\frac{f}{K}) = B(f)$ from which we have $C(f) = \frac{1}{K}A(f)$ for $f \in \left[-\frac{1}{2K}, \frac{1}{2K}\right]$. So, to recover the original signal a from signal b, we have to multiply b's interleaved version c by constant K and filter the result with a brick-wall filter[28] which removes superfluous copies of A from C. This process is known as *upsampling*. These findings are formalized in the following theorem.

4.10.4 Definition *Sinc Function*

Function $\mathrm{sinc} : \mathbb{R} \to \mathbb{R}$ is defined as follows. Note that $\mathrm{sinc}(0) = 1$.

$$\mathrm{sinc}(x) := \lim_{y \to x} \frac{\sin y}{y} \tag{298}$$

4.10.5 Theorem

(Shannon's Sampling Theorem) The signal a, whose spectrum A is band-limited to the interval $\left[-\frac{1}{2K}, \frac{1}{2K}\right]$ in a normalized frequency scale, can be exactly reconstructed from $b := \mathcal{D}_K(a)$, i.e. from its every K-th sample, in the following way.

$$a = c * \lambda n.\, \mathrm{sinc}\, \frac{\pi n}{K} \qquad \text{where } \; c_n := \begin{cases} b_{n/K} & \text{for } \; n\%K = 0 \\ 0 & \text{else} \end{cases} \tag{299}$$

Proof The theorem has been essentially proven in the foregoing discussion. All that remains is to determine the impulse response of the interpolating brick-wall filter but
88 ◁ this, again, was done in 4.4.4. Note that the multiplication factor K has been incorporated into the filter's response, causing it to be equal to 1 at $n=0$. Q.E.D.

[28] Identical to antialiasing filter but now called an *interpolating filter* from obvious reasons.

4.10.6 Band-Limited Signals

The sampling theorem requires the signal a to be band-limited. Unfortunately, it turns out that any band-limited signal cannot be finite, as will be shown in the following theorems. This makes the above method of downsampling and subsequent reconstruction impractical as it would have to cope with infinite-duration signal even if the original signal was finite.

4.10.7 Observation

Any non-zero band-limited signal x cannot be from σ_F.

Proof Let us assume we have $x \in \sigma_F$. Then $X(f) = \sum_{k=-M}^{M} x_k e^{-2\pi i k f}$ for certain M. Defining $w := 1/z = e^{-2\pi i f}$ we can write $X(f) = z^M \sum_{k=0}^{2M} x_{k-M} w^k$. But according to the *fundamental theorem of algebra*, this (non-zero) polynomial may have at most $2M$ zeroes. Hence the signal could not be band-limited as this would require $X(f) = 0$ for a whole interval of frequencies. Q.E.D.

As there is no fundamental difference between a signal and an LTI system having that signal as its impulse response, the observation also implies that a brick-wall filter cannot have finite impulse response. Actually, more than that — it does not matter what shape the filter has in the pass band (it may be even smooth). What counts is that $H(f)$ is zero in certain interval. For the same reason we cannot have non-constant filter with $H(f) = 1$ on an interval — just consider $\widehat{H}(f) := 1 - H(f)$.

So far we know that perfectly attenuating filter has infinite impulse response or, equivalently, that a non-zero band-limited signal cannot be finite. So, if we wanted to filter the signal before the decimation, we would need to know all its samples beforehand. This is highly impractical and we might ask if there is another LTI system with perfect attenuation which would use only the past samples at the cost of having non-linear phase response. The following theorem states that this is impossible, too.

Therefore the best we can hope for is an approximation of the brick-wall filter with sufficient attenuation in the stop band and small enough distortion in the pass-band.

4.10.8 Theorem

Any signal $a \in \sigma$ such that $a_n = 0$ for $n < 0$ and $a_0 \neq 0$ cannot be band limited.

Proof Let us assume for contradiction that we have such signal a whose spectrum $A(f)$ is zero at frequencies $|f| > f_0$ and $a_n = 0$ for $n < 0$. We can assume that $f_0 < 1/8$. If it was higher we could have taken $\widehat{A}(f) := A(f)A(f-\alpha)A(f+\alpha)$ instead. For suitable value of $\alpha > 0$ this would extend zero interval of the transfer function (we might need to perform this trick several times before getting spectrum with f_0 below $1/8$). It is easy to check that $\widehat{A}(f)$ corresponds to $\widehat{a} = a * (a \odot m) * (a \odot \overline{m})$ where $m := \lambda n.\exp(2\pi i \alpha n)$ and that $\widehat{a}_0 = a_0^3 \neq 0$ and $\widehat{a}_n = 0$ for $n < 0$.

Now, assuming that A is already zero on $\left[-\frac{1}{2}, \frac{1}{2}\right] \setminus \left[-\frac{1}{8}, \frac{1}{8}\right]$, let us look at spectral convolution $B := \overline{A} * A$. Since $f_0 < 1/8$ there is $f_1 < 1/4$ for which $B(f_1) = 0$. On the other hand $\mathcal{F}^{-1}(B) = \overline{\rho a} \odot a = ||a_0||^2 \cdot \vec{1}$, according to 4.3.3 and (189). But $\mathcal{F}(\vec{1})(f) = 1$ for any f, which is in contradiction with $B(f_1) = 0$. Q.E.D.
▷ 84 ▷ 85

4.10.9 Note

This theorem can be informally justified by yet another way: If there would be a brick-wall low-pass LTI system only using the past samples, we could run it on signal composed of impulses $\vec{1}[n_k]$ spaced such that their original position would be still apparent after filtering. Then by filtering the result with long enough cascade

97 ◁ of (232), using its negative group delay at low frequencies, we should be able to decode future position of those impulses. But this is impossible because we only used samples from the past which have no information about positions of future impulses, in general. Nevertheless, to make this idea a proof, we would need to quantify shape distortion introduced by (232), which would be likely more complicated than direct proof of 4.10.8.

4.10.10 Universal Resampling

So far we can downsample or upsample the signal by an integer factor. As we do not want to assume anything about the input signal we need to filter it before decimating it. This leads to the following downsampling formula, which downsamples the signal D-times.

$$y_n := \sum_k z_{(nD-k)} \frac{\operatorname{sinc}(\pi k/D)}{D} \tag{300}$$

On the other hand, the upsampler, making the sampling frequency U-times higher, is given by the following formula, according to (299).

$$z_n := \sum_k x_k \operatorname{sinc} \pi \left(\frac{n}{U} - k \right) \tag{301}$$

By stacking the downsampler after the upsampler we can also resample the signal by any rational number $r := U/D$. Special case of $D = U$ seems to be useless but only before we consider the possibility of delaying the intermediate signal y by P sampling periods. It turns out that this corresponds to constant group delay of $s := P/U$ between y and x, an effect that cannot be achieved by any digital filter, as we already know
107 ◁ from (267).

By plugging (301) delayed by P samples into (300), we receive the following.

$$\begin{aligned} y_n &= \frac{1}{D} \sum_p \sum_k x_k \operatorname{sinc} \pi \frac{nD - Uk - P - p}{U} \operatorname{sinc} \pi \frac{p}{D} \\ &= U \sum_k x_k \sum_p \frac{1}{U} \operatorname{sinc} \left(\pi \frac{(nD - Uk - P) - p}{U} \right) \frac{1}{D} \operatorname{sinc} \pi \frac{p}{D} \end{aligned} \tag{302}$$

The last sum may be understood as a convolution evaluated at sample $nD - Uk - P$. Ac-
83 ◁ cording to 4.3.1, its spectrum is then a product of spectra of the two $\lambda n.2 f_0 \operatorname{sinc}(2\pi f_0 n)$
88 ◁ signals. As these are in fact brick-wall filters 4.4.4, their product is also a brick-wall filter with cut-off frequency f_0 being a minimum of these two. Hence we obtain universal resampler, which changes the sampling rate by arbitrary rational factor $r = U/D$, applying rational delay $s = P/U$, which is measured in the time units of the original signal.

$$\begin{aligned} y_n &= \frac{U}{\max(U, D)} \sum_k x_k \operatorname{sinc} \pi \frac{nD - P - Uk}{\max(U, D)} \\ &= \min(r, 1) \sum_k x_k \operatorname{sinc} \pi \left(n \min(r^{-1}, 1) - (k + s) \min(r, 1) \right) \end{aligned} \tag{303}$$

Note that the last formula embeds (300) and (301) in itself and it also justifies notion of
96 ◁ non-integer group delay, as promised in the paragraph after equation (227). Moreover, it can be used to define resampling by a real factor $r \in \mathbb{R}$ and time shift $s \in \mathbb{R}$.

4.10.11 Practical Resampling

The above resampling formula, although being conceptually simple, is quite troublesome when it comes to its implementation. First, it cannot be implemented by a digital filter (not even IIR) because its frequency response is exactly zero on an interval and digital filters only allow for finite set of zeores. Second, the entire signal must be known in advance before the processing could begin. Third, the impulse response falls off very slowly (only as $1/|n|$) and it is not absolutely convergent. Consequently, peaks of arbitrary height can appear in the resampled signal even if the input signal was amplitude limited. Some of those peaks can be very far from the portion of the signal that caused them (cf. 4.6.5). ▷ 97

While the first two objections are implementation related, the third one disputes the very principle of resampling. Intuitively, we feel that information about the signal's shape should not be spread too much in time by resampling. But the brick-wall filter makes future information available anywhere in the past, at least theoretically (whereas practically this is limited by precision of the arithmetics).

For these reasons, instead of brick-wall filter, practical resamplers use only an approximation thereof. These approximative filters have finite steepness of the transition region between the pass-band and stop-band, and the attenuation in the stop band is not perfect albeit still high enough to keep aliasing artifacts well below quantization (or noise) level of the original signal.

As already hinted, the design objective of these resampling filters is not solely a computationally efficient approximation of the brick-wall filter but rather a trade-off between good shape of the frequency response[29] and suppression of the time domain artifacts of which the most profound is so called *ringing*, also known as the *Gibbs' phenomenon*. It exhibits itself as wavelets around discontinuous or sharp changes in the signal. It is visible in the middle panel of fig. 14, where the original impulse got ▷ 98 transformed into the ringing wave. Gibbs' phenomenon is especially annoying in image processing[30] because it introduces false edges into the image. In audio processing it is usually not a big concern, firstly because the amplitude of high frequency signals is usually small[31] and, secondly, our brain is not very good at detecting that extra signal if it lasts less than, say, 1 ms (which is attainable with transition band of about 2 kHz). That is why there would be no problem unless the resampling filter is too steep and input's sampling frequency too small (such as in case of upsampling from 8kHz to 48 kHz). In audio systems intended for human listeners, the ringing artifacts can be further mitigated by making the resampling filter minimum phase. While it has the same amplitude response as the original linear phase design, its output peaks much sooner after the first wavelets associated with the same input have appeared. Even though the minimum phase filter has longer ringing tail than its linear phase counterpart, the resulting artifact is less audible, owing to ear's temporal masking 5.8.7. Minimum phase resampling filter is also welcome for its low latency, which might ▷ 171 be advantageous in telephony applications.

[29] High attenuation in the stop band, flat passband and narrow transition band.

[30] The signal theory can be extended to higher dimensions. Images are two-dimensional signals then.

[31] In case of $f_s \geq 48\,\text{kHz}$ the ringing carrier will be beyond the frequency range of most people anyway.

A price paid for this trick is non-constant group delay that distorts time-domain shape of the signal. This is of no concern in audio playback, as the human ear is rather insensitive to phase but in precision measurement applications (like the recording of an ECG signal), temporally undistorting linear phase filter is preferable.

4.10.12 Note on D/A Conversion

Physical conversion of binary numbers representing the samples into a continuous electric signal takes place in a device called *Digital to Analog Converter*, or shortly the *D/A converter* or *DAC*. The converter is followed by analog low pass filter (made of resistors, capacitors and inductors) which acts as the interpolating filter. But unlike the theoretical upsampler (301), DAC does not create the signal from spiky impulses. Instead, it makes it from steps, holding the output voltage constant until the next sample arrives. As a consequence the mirrored higher spectra that have to be removed by the interpolating filter are weaker than the main spectrum, which further simplifies filtering circuits.

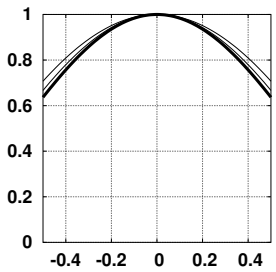


Fig. 17. *Distortion $U_K(f/K)$ drawn for $K=2$ (upper curve), $K=3$ (the middle one) and for $K\to\infty$ (lower curve).*

However, the stepped upsampling has its problems, too. Feeding the interpolator 117◁ (299) with $\widehat{c}=\lambda n.\,K^{-1}b_{\lfloor n/K\rfloor}$ to simulate the steps, we get the following formula for its spectrum.

$$\begin{aligned}\widehat{C}(f)=\mathcal{F}(\widehat{c})(f)&=\sum_{k=0}^{K-1}\sum_{n=-\infty}^{\infty}\frac{b_n}{K}e^{-2\pi i(nK+k)f}=\sum_{n=-\infty}^{\infty}\frac{b_n}{K}e^{-2\pi inKf}\sum_{k=0}^{K-1}e^{-2\pi ikf}\\&=B(Kf)\underbrace{\frac{\sin(\pi Kf)}{K\sin(\pi f)}}_{U_K(f)}\underbrace{e^{-\pi i(K-1)f}}_{\star}\end{aligned}\tag{304}$$

A perfect upsampler would then multiply $\widehat{C}$ with a brick-wall response, retaining only the frequencies from $\left[-\frac{1}{2K},\frac{1}{2K}\right]$. However, instead of getting the correct $B(Kf)$, there are two extra factors in (304). $U_K(f)$ causes amplitude distortion of the spectrum, while $\star$ is only a consequence of asymmetric creation of the steps — it would disappear had we defined the steps centered around the original spikes. Distortion factor U_K is shown in fig. 17 for various values of K in common frequency scale so that the plots could be compared. Hence, to obtain correct reconstruction the digitized signal b should be filtered with the following correction response, before being sent to the converter.

$$V(f)=\frac{1}{U_K(f/K)}=\frac{K\sin(\pi f/K)}{\sin(\pi f)}\quad\xrightarrow[K\to\infty]{}\quad\frac{\pi f}{\sin\pi f}\tag{305}$$

Though $\mathcal{F}^{-1}(V)$ cannot be realized by a digital filter (as it has discontinuous derivative in $f=1/2$) it can be closely approximated. Truly Hi-Fi system would additionally include other imperfections of the analog path into the correcting filter.

However, such corrections are rarely implemented in commodity computer hardware. Moreover, this analysis assumed traditional DAC (such as the *R-2R ladder* circuit) that holds constant voltage between consecutive samples. Nowadays, common D/A converters work on different principle which is similar to the dithering of halftone images on monochromatic displays. Such a converter may be of only 1-bit resolution but much higher sampling rate — say, 512-times higher than f_s, the sampling rate of the feeding digital signal. It turns its output on and off very rapidly so that the mean value over the original sampling period would correspond to the intended analog output, whilst shaping the spectrum in a way that makes majority of quantization noise to occur above $f_s/2$. This high-frequency noise is then attenuated by cheap RC-filter connected to the output of the DAC. Hence, there are no apparent boundaries between individual samples, therefore no need to do the compensation (the distortion is still present but since 1 of the normalized frequency corresponds to $512f_s$, the useful spectrum lies only in $(-2^{-10},2^{-10})$ interval of fig. 17 where the distortion is negligible).

The distortion $U_K(f)$ also affects displays, as these have non-zero pixel size. Normally, it is not very visible (perhaps it is compensated for in the camera). But if one tries to shrink the image by true downsampling (303) the resulting image would not look as sharp as it ought to. This can be corrected by filtering rows and columns with $V(f)$, which 'sharpens' the image. However, note that the downsampling was correct and this compensation should be carried out only for viewing purposes — should the shrunken image undergo another processing in the future, its uncompensated version would have to be used.

Also note that resizing an image by true downsampling is a bad idea. It is slow and, as noted earlier, it leads to ringing on the edges. Better resampler would use antialiasing filter with localized impulse response with only negligible ringing. Nevertheless, the problem of $U_K(f)$ distortion would apply to it, too. Moreover, there is another problem typical for images, described in the next subsection. As it does not apply to sound (with $f_s>44\,\mathrm{kHz}$) you might want skip to 4.10.14 if interested in speech processing only. ▷123

4.10.13 Kell Factor

Sometimes the signal reconstruction cannot easily incorporate the interpolating filter. Notable examples are LCD and plasma displays[32]. Here, the higher mirrored spectra exist in the reconstructed signal[33] (for pure colors (red, greed and blue) these higher spectra are more profound then we would expect from stepped reconstruction formula (304) because the reconstruction of purely red image is closer to the spiked reconstruction of (301) since neighboring red and green subpixels are turned off, that is zero).

The existence of these higher spectra can be directly observed by naked eye in an image whose all scan lines are of brightness $1+\cos(2\pi fx)$, where $f<1/2$. Note that the image is antialiased as it contains only single frequency[34] less than 1/2. Neglecting all higher spectral copies except the first one, our eyes would see scan line of brightness $1+\cos(2\pi fx)+\alpha\cos(2\pi(1-f)x)=1+(1-\alpha)\cos(2\pi fx)+2\alpha\cos(\pi n)\cos(\pi(1-2f)n)$. So, the true image is attenuated by $1-\alpha$ while the error term superimposes a moire on

[32] For (monochromatic) CRT displays this applies only to individual columns. Rows are filtered electronically. Even the columns may be sort-of filtered by letting the scan lines partially overlap.

[33] To stay within a realm of one-dimensional signals, imagine a single line of the display as a signal.

[34] Assuming very long display so that the spectral line width and off-band energy can be neglected.

it. Moreover, for f close to $1/2$ the number $1-2f$ is small, causing the last factor to vary slowly with x, which produces visible *beating* in intensity of the artifact. It might look like an aliasing but it is of different origin.

Fig. 18. *The Kell artifact. The upper stripe represents discrete display showing a pattern $1+\cos(\pi kx)$ for $k=0.98$, where $x\in\mathbb{N}$ is the number of a pixel in a row. To emphasize the effect by reducing attenuation of the $U_K(f)$ distortion, the pixels are interleaved with a white space of twice their width (this corresponds to real LCD, displaying red color with blue and green subpixels turned off). Three 'beats' of the artifact pattern can be seen there. The period of the error term's factor $\cos(\pi(1-k)x)$ is $\frac{2}{1-k}=2/0.02=100$. There are two visible beats per period due to the factor $\cos(\pi x)=(-1)^x$. As the display is 146 pixels wide, it gives 3 beats. The stripe below contains thrice as many pixels placed next to each other without gaps, showing the same signal. The Kell artifact is hardly visible there. The upper stripe would look similarly had it been filtered by proper interpolating filter. Due to nonlinear dependence of blackness (which is hard to be completely compensated by a gamma correction, owing to paper and ink aging) there is also an aliasing artifact present. The aliased third harmonic has a period of 2.13 pixels (of the upper display), while for the second one it is 50 pixels, the same as the beating period. However, the aliasing leads to purely sine waves without the $(-1)^x$ modulation factor that is clearly visible in the figure, which shows that we really see the Kell effect. Even higher harmonics are of negligible amplitude in comparison to $\alpha\approx 1$ of the error term.*

How to suppress this artifact to visually acceptable levels was proposed by Raymond D. Kell, television researcher working for RCA Corporation, in 1934. The method effectively reduces sharpness by filtering the image data so as to remove all (normalized) frequencies above $0.7/2$. This makes the number $1-2f$ always larger than 0.3, thereby disabling visually most annoying long waves in the beating pattern. The number 0.7 is called the *Kell factor*. It is usually taken in range of 0.64 to 0.9, depending on the display and digital filter used to remove higher frequencies. Note that the filter must not be too steep either, or the ringing on edges would appear, as discussed earlier.

There are other complications with images, though. I will only mention one for illustration, before returning to sound: Constantly bright area displayed on LCD in fact contains high frequency components caused by a thin black frame around each pixel. The amplitude of these components depends on the area's brightness — for black area it is zero, whereas for a shiniest white it is maximal. As a result, properly antialiased image (imagine a white page of black text) looks inappropriately when displayed as a negative even if it has been correctly compensated for gamma correction. It may be understood as a consequence of black pixels being slightly larger than the white ones.

4.10.14 Effect of Resampling on Digital Filter Coefficients

Suppose we consider all parasitic effects of the analog path[35] to be modeled by a digital filter running at very high ($\approx 10^{50}$ Hz) sampling rate. Surely this is an approximation as it disregards nonlinear effects which are also present. Nevertheless, let us neglect them now, for sake of simplicity. Then we have the input sound x transformed into y, the electric signal entering the A/D converter, in the following way.

$$y = d^{-1} * c * x \qquad \text{where } c,d\in\sigma_F \tag{306}$$

Now we could still invert the distortion by convolving y with $c^{-1}*d$, provided that c was invertible. But in the converter the y gets heavily downsampled to reasonably slow-rate signal $\widetilde{y} := \mathcal{D}_K(y*h_K)$, where h_K is a brick wall filter removing any normalized

[35] Reverb of the room and uneven sensitivity of the microphone and amplifier at different frequencies.

frequency $|f|>(2K)^{-1}$. The question arises whether it is possible to reconstruct $\widetilde{x}$, the downsampled version of x, from $\widetilde{y}$ with a digital filter. To answer it, let us investigate properties of the downsampling operation $\lambda s.\widetilde{s}$.

4.10.15 Observation *For $x,y\in\sigma$ and $z\in\sigma_I$ we have: $\widetilde{x*y}=\widetilde{x}*\widetilde{y}$ and $\widetilde{z}^{-1}=\widetilde{z^{-1}}$.*

Proof The left side of the first equation is $\mathcal{D}_K(x*y*h_K)$ or, in the frequency domain, $\mathcal{D}_K(XYH_K)$. Since H_K is brick-wall filter we have $H_K=H_K\cdot H_K$ and $H_K(f)=0$ for $|f|>(2K)^{-1}$. From (293) it then follows that $\mathcal{D}_K(XYH_K)=\mathcal{D}_K((XH_K)(YH_K))=$ ▷116
$\mathcal{D}_K(XH_K)\mathcal{D}_K(YH_K)$ which equals $\widetilde{x}*\widetilde{y}$ when translated back to the time domain. The second equation follows from the first one, since $\widetilde{\vec{1}}=\widetilde{z*z^{-1}}=\widetilde{z}*\widetilde{z^{-1}}$. Multiplying both sides by $\widetilde{z}^{-1}$ and noting that $\widetilde{\vec{1}}=\vec{1}$, we get $\widetilde{z}^{-1}=\widetilde{z^{-1}}$. That $\widetilde{z}$ was invertible is clear from its spectrum. Q.E.D.

The observation implies that $\widetilde{y}=\widetilde{d}^{-1}*\widetilde{c}*\widetilde{x}$. Unfortunately, as $\widetilde{c}$ and $\widetilde{d}$ got band-limited by downsampling, they cannot live in σ_F any longer, due to 4.10.7. Even ▷118
though these fall towards zero, it is a slow process, owing to a sinc shape of h_k. With already discussed effect of ringing, this is another reason against exact downsampling (apart from its prohibitive computational complexity). Notice that although $\widetilde{\vec{1}}$ is finite, $\widetilde{\vec{1}}[1]$ is not. This is a problem especially for video where it is responsible for motion artifacts. A thin vertical line moving from left to right would change its apparent width in the downsampled movie. But it is a problem for sound, too because the information that leaks to the surrounding samples during downsampling depends on exact timing, which, intuitively, should not happen.

4.11 Hilbert Transform and Analytic Signal

Negative frequencies in the spectrum of a real valued signal are superfluous because the spectrum satisfies $X(-f)=\overline{X(f)}$. This observation leads to the following definition.

4.11.1 Definition *Analytic Signal*

Signal a is *analytic to signal* x iff $x_n\in\mathbb{R}$ for all $n\in\mathbb{Z}$ and the following holds for its spectrum $A:=\mathcal{F}(a)$.

$$A(f)=\begin{cases}2X(f) & \text{for } f>0\\ X^+(0) & \text{for } f=0\\ 0 & \text{for } f<0\\ X^-(f) & \text{for } f=-1/2\end{cases} \tag{307}$$

where $X:=\mathcal{F}(x)$ and $f\in[-1/2,1/2)$. We call the signal *analytic* iff it is analytic to some signal, which it is iff $A(f)=0$ for $f\in(-1/2,0)$. The operator that makes A from X using eq. (307) will be denoted by $\mathcal{Q}$ and called the *quadrature filter*. It can be extended to time domain by defining $\mathcal{Q}(x):=\mathcal{F}^{-1}(\mathcal{Q}(\mathcal{F}(x)))$. So if x was real-valued, then $\mathcal{Q}(x)$ would represent the signal which is analytic to it.

As only counterclockwise rotating exponentials remain in a spectrum of analytic signal, the signal seems to rotate in that direction, especially if it is narrow-band so that the exponentials present are all about the same frequency. This effect can be seen in figure 19. ▷125

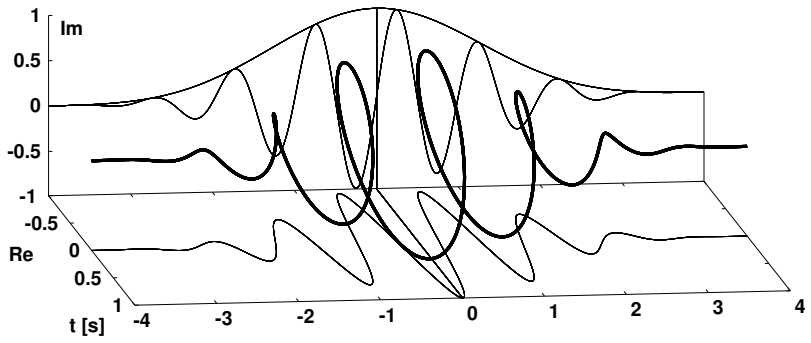


Fig. 19. Analytic signal $z_n := \left(\frac{\sin(2\pi\beta t)}{2\pi\beta t}\right)^4 e^{2\pi i t}$, *where* $\beta := 0.11$ *and* $t := n/1000$ *(the sampling frequency is 1000 Hz here). Also drawn are its real and imaginary projections which are related thru the Hilbert transform in the following way:* $\mathrm{Im}(z) = \mathcal{H}(\mathrm{Re}(z))$. *Finally, there is an amplitude envelope* $||z||$ *drawn above the* $\mathrm{Im}(z)$ *projection.*

4.11.2 Definition *Hilbert Transform*

The *Hilbert transform* is $\sigma \to \sigma$ function defined as follows[36]

$$\mathcal{H}(x) := \mathcal{F}^{-1}(-i\,\mathrm{sgn}_{\mathcal{S}} \cdot \mathcal{F}(x)) \tag{308}$$

We extend it to $\mathcal{S} \to \mathcal{S}$ as $\mathcal{H}(X) := \mathcal{F}(\mathcal{H}(\mathcal{F}^{-1}(X))) = -i\,\mathrm{sgn}_{\mathcal{S}} \cdot X$.

4.11.3 Note This means that the spectrum $X(f)$ is multiplied by $-i$ for $f > 0$ and
96 ◁ by i for $f < 0$, which introduces $\pi/2$ phase delay to x, according to (228). Even this effect is visible in fig. 19 when looking carefully — the ripples in the imaginary projection lag 1/4 second behind the ripples in the real projection. As the ripple wavelength is 1 second, it amounts to a $\pi/2$ phase delay.

4.11.4 Observation $\mathcal{Q}(x) = x + i\mathcal{H}(x)$

Proof Clearly, $\mathcal{Q}(X) = X + i\mathcal{H}(X)$ for $f \in \left[-\frac{1}{2}, \frac{1}{2}\right) \setminus \{0, -\frac{1}{2}\}$. But those two points at which the spectrum may differ cannot change value of the integral in $\mathcal{F}^{-1}$, hence $\mathcal{Q}(x) = \mathcal{F}^{-1}(\mathcal{Q}(\mathcal{F}(x))) = \mathcal{F}^{-1}(\mathcal{F}(x) + i\mathcal{H}(\mathcal{F}(x))) = x + i\mathcal{H}(x)$. Q.E.D.

4.11.5 Lemma $\mathcal{H}(x) = h * x$, *where*

$$h_k = \begin{cases} 0 & \text{for } k = 2n,\ n \in \mathbb{Z} \\ \frac{2}{\pi k} & \text{else} \end{cases} \tag{309}$$

Consequently $\mathcal{Q}(x) = (\vec{1} + ih) * x$.

87 ◁ **Proof** $h = \mathcal{H}(\vec{1}) = \mathcal{F}^{-1}(\mathcal{H}(1))$ according to 4.4.2. Hence

$$h_k = \int_{-1/2}^{1/2} -i\,\mathrm{sgn}_{\mathcal{S}}(f) e^{2\pi i f k}\,df = -i\int_{0}^{1/2} e^{2\pi i f k}\,df + i\int_{1/2}^{1} e^{2\pi i f k}\,df \tag{310}$$

[36] The $\mathrm{sgn}_{\mathcal{S}}(f)$ is 1-periodic function being equal to $\mathrm{sgn}(f)$ on $f \in (-\frac{1}{2}, \frac{1}{2})$ such that $\mathrm{sgn}_{\mathcal{S}} \in \mathcal{S}$.

Integrating it, we get

$$h_k = -i\frac{e^{\pi i k} - 1}{2\pi i k} + i\frac{1 - e^{\pi i k}}{2\pi i k} = \frac{1 - (-1)^k}{\pi k} \tag{311}$$

Q.E.D.

Under certain conditions, the quadrature filtering can be used for decoding amplitude and phase modulated signals. Suppose the modulated signal is $s := a \odot c$, where $a_k \geq 0$ is the amplitude envelope and $c_k := \mathrm{Re}(d_k) = \cos(2\pi f_c k + \varphi_k)$ is phase modulated carrier wave, where $d_k := \exp(2\pi i f_c k + i\varphi_k)$ is its complex-valued version.

For φ_k such that its corresponding d_k is band-limited[37] to $[f_c - f_\Delta, f_c + f_\Delta]$ and a_k being band-limited to $[-f_a, f_a]$ the product $a \odot d$ is band-limited to $[f_c - f_\Delta - f_a, f_c + f_\Delta + f_a]$, since product in the time domain corresponds to a convolution in spectral domain. When additionally $0 \leq f_c - f_\Delta - f_a$ and $f_c + f_\Delta + f_a \leq 1/2$ hold, the signal $z := a \odot d$ is analytic to $s = \mathrm{Re}(z)$ and can be exactly reconstructed from s as $z = \mathcal{Q}(s)$. It is then possible to recover its envelope $a_n = ||z_n||$ as well as its phase $\varphi_n = \mathrm{Arg}(z_n) - 2\pi f_c n$. This process is called the *demodulation*.

Note that exact demodulation relied on z's zero power at negative frequencies which allowed the quadrature filter to be used for recovering z from the measured signal s. This would not quite work if a or φ were spoiling z's analyticity in some way. For that reason the amplitude envelope in fig. 19 was carefully selected to be band-limited onto the interval $[-4\beta, 4\beta]$, where $\beta = 0.11$ Hz. The monochromatic carrier wave of 1 Hz shifts this up to $[0.66, 1.44]$, making the signal analytic.

Unfortunately, this happens rarely in the real world. As it was demonstrated in 4.10.7, any time-limited signal cannot be band-limited. But any practical (measured) ▷ 118 signal is finite-duration. For long enough signals, this could be neglected as the power at negative frequencies would be relatively small in comparison with the power at the positive ones. What is worse, however, is that the band-limitedness of φ does not imply band-limitedness of d. Even the simple case of $\varphi_n := \cos(2\pi f n)$ produces wide-band d.

Practical solution, routinely used in radio communication, is to 'remove' (attenuate) negative frequencies from z before it is converted to $s := \mathrm{Re}(z)$ and transmitted. This, however, changes z such that it will correspond to a slightly different signals a and φ. Often, it is more convenient to remove off-band frequencies directly from $2s = z + \overline{z}$, which frees us from knowing z. This, however, costs even greater distortion — in case of filtered z the distortion came solely from the missing sidebands of z, whereas the distortion of filtered s has additional component caused by the aliasing of sideband of $\overline{z}$ into the main-band of z. Nevertheless, this is of little concern in practical applications because the power of the sidebands is typically quite small.

4.11.6 Bandwidth of the Demodulated Components

As we usually demodulate narrow band-signals a question arises how many sample points do we need to uniquely represent the resulting envelope and phase if z's spectrum

[37] Technically, d is not a signal since $d \notin \sigma$ and we cannot speak of its spectrum and use spectral convolution 4.3.3 for it. But a is from σ and, as such, it is going towards zero at its tails. So we can ▷ 84 take long enough finite segment of d such that the energy of the discarded portion of $a \odot d$ would be negligible. By growing this segment indefinitely, we would obtain the result in the limit. This is how d-related formulas in this section should be read. In fact this is the place where theory of distributions that I decided to side-step would allow us to write it directly.

115 ◁ was non-zero only on interval $[f_A, f_B]$. According to 4.10 we need sampling frequency to be at least $f_B - f_A$ to represent z. Accordingly, twice as many sampling points would be needed to represent s, as its negative frequencies must also be encoded. Nonetheless it leads to the same amount of data in both cases because s is real, whereas z was complex. As outlined above, phase φ_n and envelope a_n can be directly recovered from z_n, so these can be sampled with frequency $f_B - f_A$, too. Notice that the resampling must be performed already on the analytic signal z. Only after that we can convert z to phase and envelope. If we extracted a_n and φ_n before resampling, these would have
85 ◁ unnecessarily wide spectra due to nonlinear nature of this extraction and (191).

4.11.7 Note Hilbert transform usually finds its application as a demodulator of narrow band signals. However, it sometimes unexpectedly emerges from deep theory. For instance, we have met its dual, that is a Hilbert transform acting on spectra as if
109 ◁ they were continuous signals, in formula (271).

4.12 Quantization and the Information Carried by the Signal

A single signal as it has been considered so far could be used to encode infinite information, even in its single non-zero sample. This is because our definition allowed the signal to take real numbers as its values. Realistic signals, on the other hand, do not allow anything close to this. Their values are either limited into finite number of discrete levels or (in case of analogue signals) there is a ubiquitous noise that impedes any attempts to store information below certain amplitude level. This second case can be approximated by the first one — with the quantization hidden deeply below the noise level, the distinction would be negligible. Thus, in this section we shall work with integer-valued signals. This way, each sample would be able to carry only finite information. Integer-valued signals are equivalent to signals quantized to equidistant levels. We just leave the decision about actual size of the step to implementation (in this respect it is similar to the normalized frequency scale).

Let us first investigate how much information can be stored in a single sample of a signal (i.e. in a single number) in presence of unknown noise. We will need the following definitions for that.

4.12.1 Definition *Variance and Standard Deviation*

For random variable $X : \Omega \to \mathbb{R} \cup \{\infty, -\infty\}$ we call

$$\mathrm{Var}_\Omega(X) := \mathcal{E}_\Omega\left((X - \mathcal{E}_\Omega(X))^2\right) = \mathcal{E}_\Omega(X^2) - (\mathcal{E}_\Omega(X))^2 \tag{312}$$

its *variance* and $\sigma(X) := \sqrt{\mathrm{Var}_\Omega(X)}$ its *standard deviation*. We can write σ for $\sigma(X)$ if there is no danger of confusion with $\sigma(Y)$ or with the set of all signals σ.

4.12.2 Note The equality in (312) holds because $\mathcal{E}\left((X-\mathcal{E}(X))^2\right) = \mathcal{E}(X^2)+(\mathcal{E}(X))^2 - 2\mathcal{E}(X\mathcal{E}(X)) = \mathcal{E}(X^2) - 2\,\mathcal{E}(X)\mathcal{E}(X) + (\mathcal{E}(X))^2 = \mathcal{E}(X^2) - (\mathcal{E}(X))^2$. Also note that the standard deviation scales linearly, i.e. $\sigma(\alpha X) = \alpha\sigma(X)$ for $\alpha \in \mathbb{R}$.

4.12.3 Note Additionally, $\mathrm{Var}(X+Y) = \mathrm{Var}(X)+\mathrm{Var}(Y)$ for uncorrelated X and Y because $\mathrm{Var}(X+Y) = \mathcal{E}\left((X+Y-\mathcal{E}(X+Y))^2\right) = \mathcal{E}\left((X-\mathcal{E}(X))^2\right) + \mathcal{E}\left((Y-\mathcal{E}(Y))^2\right) - 2\mathcal{E}((X-\mathcal{E}(X))(Y-\mathcal{E}(Y)))$, where the last term is zero, owing to uncorrelatedness 3.1.11 ▷ 42 of X and Y. This observation can be generalized for mutually uncorrelated random variables X_k as $\mathrm{Var}\left(\sum_k \alpha_k X_k\right) = \sum_k \alpha_k^2 \mathrm{Var}(X_k)$. This simple result has important practical consequence for N uncorrelated identically distributed random variables X_k and $\alpha_k := 1/N$, giving $\sigma(\frac{1}{N}\sum_k X_k) = \sigma(X_1)/\sqrt{N}$. This means that averaging N realizations of a signal, each of which being corrupted by random noise of known $\sigma(X_1)$ makes standard deviation of the result only $\sqrt{N}$-times smaller than $\sigma(X_1)$. Thus, relative effectiveness of noise suppression diminishes with the number of measurements we average. Moreover, in practical application we can never be completely sure about uncorrelatedness of the noise (for instance, when averaging sound from several microphones, the electrical noises of amplifiers are likely to be uncorrelated but that cannot be said about the environmental noise being picked by the microphones). Averaging would introduces systematic error in such a case.

4.12.4 Note Standard deviation can be thought of as a measure of interval within which the value of the random variable dwells most of the time. For instance, the value $X(\omega)$ of X having a Gaussian distribution[38] falls into $(\mathcal{E}(X)-3\sigma(X),\ \mathcal{E}(X)+3\sigma(X))$ with probability greater than than 99.7%. It is remarkable that similar statement holds even if nothing is assumed about the distribution of X, at the price of somewhat larger interval. This is the subject of the following lemma.

4.12.5 Lemma *(Chebyshev Inequality) For any random variable $X : \Omega \to \mathbb{R}$ with finite $\mathcal{E}(X)$ and $\mathrm{Var}(X)$ and any $\alpha > 0$, the following holds.*

$$\Pr_\Omega\left(\text{“}|X - \mathcal{E}_\Omega(X)| \geq \alpha\text{”}\right) \leq \frac{\mathrm{Var}_\Omega(X)}{\alpha^2} \tag{313}$$

Proof For sake of simplicity, let us shift the distribution so that $\mathcal{E}(X)=0$ and consider total probability mass of the tails, that is $\Pr(\text{“}|X| \geq \alpha\text{”}) = \sum_{x\,\in\,M} p(x)$ where $M := \{x \in \mathrm{Rng}(X) \,|\, |x| \geq \alpha\}$ and $p(x) := \Pr_\Omega(X^{-1}[x])$. Now

$$\alpha^2 \Pr_\Omega(\text{“}|X| \geq \alpha\text{”}) = \sum_{x\,\in\,M} \alpha^2 p(x) \leq \sum_{x\,\in\,M} x^2 p(x) \leq \mathrm{Var}_\Omega(X) \tag{314}$$

Q.E.D.

4.12.6 Relationship Between Variance and Entropy

Let us find probability distribution that has the maximal entropy of all the distributions that share a fixed variance σ^2. Let us do it on finite domain $D := \mathbb{Z} \cap [-B, B]$, where $B \in \mathbb{N} \setminus \{0\}$, to avoid advanced math of Banach spaces. After all, we can always take B so high that even a single encounter with a value of such magnitude would destroy our instrumentation apparatus anyway, as well as most of the nearby buildings. That is why finite B would not lessen practical relevance of the result.

[38] Approximated on dense enough discrete domain $\Omega \subset \mathbb{R}$ to avoid the need of Kolmogorov axioms.

Let us first consider related problem of finding the most entropic distribution on D when both its variance σ^2 and its expectation μ are given, provided that μ allows a distribution to exist[39]. Finding the solution involves maximizing $H_e := -\sum_k p_k \log p_k$ over the following constraint set.

$$C := \{\, p \in \mathbb{R}^{\#D} \mid p_\bullet = 1,\ p_k \geq 0,\ \textstyle\sum_{k\in D} kp_k = \mu,\ \sum_{k\in D}(k-\mu)^2 p_k = \sigma^2 \,\} \tag{315}$$

As an intersection of hyperplanes and half-spaces it is obviously *convex*[40]. Additionally, it is compact, being an intersection of compact sets. Since the entropy is continuous function of $p \in C$ and a continuous function attains both the minimum and maximum on a compact set we know that the distribution of maximal entropy with given σ and μ really exist, ruling out the possibility of sequence of distributions of ever increasing entropy with no limit in C.

The maximum can be located by *Lagrange multipliers*[41] applied on a domain $X = (0,1)^{\#D}$ with constraints $g_1(p) = p_\bullet - 1$, $g_2(p) = \sum_k kp_k - \mu$ and $g_3(p) = \sum_k k^2 p_k - \mu^2 - \sigma^2$. Noting that H_e and g_k have continuous partial derivatives and $\nabla g_1 = \langle \ldots, 1, 1, 1, \ldots \rangle$, $\nabla g_2 = \langle \ldots, -1, 0, 1, \ldots \rangle$ and $\nabla g_3 = \langle \ldots, 1, 0, 1, \ldots \rangle$ are linearly independent we obtain the following system.

$$\begin{gathered} \frac{\partial H_e}{\partial p_k} + \lambda_1 + \lambda_2 k + \lambda_3 k^2 = -\log p_k - 1 + \lambda_1 + \lambda_2 k + \lambda_3 k^2 = 0 \quad \text{for } k \in D \\ \textstyle\sum_{k\in D}\ p_k = 1 \\ \textstyle\sum_{k\in D}\ kp_k = \mu \\ \textstyle\sum_{k\in D} k^2 p_k - \mu^2 = \sigma^2 \end{gathered} \tag{318}$$

[39] Probability distribution on $\mathbb{Z}$ with $\sigma < 1/2$ cannot have arbitrary μ. Consider $\mu \in [-1/2, 1/2]$. Chebyshev inequality used with $\alpha = |\mu|$ gives $1 = \Pr(|X - \mu| \geq |\mu|) \leq \sigma^2/\mu^2$, that is $\sigma \geq |\mu|$. No such restriction exist for $\sigma \geq 1/2$ on $\mathbb{Z}$. On $\mathbb{Z} \cap [-B, B]$, $|\mu| \leq B$, of course.

[40] The set $X \subseteq \mathbb{R}^k$ is called *convex* iff for every $x, y \in X$, the line segment connecting x with y is also in X, formally $\forall \alpha \in [0,1] : \alpha x + (1-\alpha) y \in X$.

[41] Let me shortly remind the method of *Lagrange multipliers*, summarizing it in the following theorem.

4.12.7 Theorem *Let the functions $f, g_1, \ldots, g_k : X \to \mathbb{R}$ have continuous partial derivatives on an open set $X \subseteq \mathbb{R}^N$ and let the vectors $\nabla g_1(x), \ldots, \nabla g_k(x)$ be linearly independent for all $x \in A$, where $A := \{\, x \in X \mid g_1(x) = 0, \ldots, g_k(x) = 0 \}$ is a set of admissible points. Note that for $k = 1$, this means that $\nabla g_1[A] \not\ni \vec{0}$. Then, if f has a maximum on A, it will be among the points x that solve the following system for some $\lambda_1, \ldots, \lambda_k$.*

$$\nabla_x \Big(f(x) + \textstyle\sum_{i=1}^k \lambda_i g_i(x) \Big) = \vec{0} \quad \textit{and} \quad x \in A \tag{316}$$

Or, in another words, stationary points (i.e. those points with zero gradient) of (316) on A are in one-to-one correspondence with stationary points of so called *Lagrangian*, a function defined as follows.

$$L(x_1, \ldots, x_N, \lambda_1, \ldots, \lambda_k) := f(\langle x_1 \ldots x_N \rangle) + \textstyle\sum_{i=1}^k \lambda_i g_i(\langle x_1 \ldots x_N \rangle) \tag{317}$$

That the gradient of (317) gives (316) is immediate — the gradient's derivatives along x_j give the first part, while partial derivatives along λ_i give the condition of x being in A. Also note that the maximum of f on A may actually correspond to the saddle point of L or even to the saddle point of $f(x) + \sum_{i=1}^k \lambda_i g_i(x)$, understood in space X.

Note that it only concerns those points p in which ∇H_e is continuous, i.e. the points where $p_k > 0$ for all $k \in D$ (this is because $-\partial(x \log x)/\partial x$ goes to infinity for $x \to 0$). Another reason for that was the requirement of X being an open set. As a result, Lagrange method can detect maximum only on $A := C \cap X = \{\, p \in C \mid \forall k : p_k > 0 \}$.

Thus, we have to rule out maximum on $C \setminus A$ by different means. Fortunately, this is easy. Take $a \in C \setminus A$ and $b \in A$. Due to convexity of C, the whole line lies in C. Let us calculate entropy at $x(\varepsilon) = a + \varepsilon(b-a)$, for some small $\varepsilon > 0$. Writing $h(\varepsilon)$ for $H_e(x(\varepsilon))$ while using the *mean value theorem*, we obtain the following increase of entropy when going from distribution a to distribution $x(\varepsilon)$.

$$h(\varepsilon) - h(0) = h'(\gamma)\varepsilon = \varepsilon \sum_{k\in D} \left(\frac{\partial H_e}{\partial x_k} \frac{\partial x_k}{\partial \varepsilon} \right)(\gamma) = -\varepsilon \sum_{k\in D} (\log x_k(\gamma) + 1)(b_k - a_k) \tag{319}$$

This holds for certain $\gamma \in (0, \varepsilon)$ whose exact value depends on function h. For those k s.t. $a_k = 0$, we can choose ε so small that $-\log x_k(\gamma) = -\log(\gamma b_k) > 0$ becomes so large that it overpowers the contribution of other (possibly negative) terms. Therefore $h(\varepsilon) > h(0)$ and we proved that maximum cannot lie in $C \setminus A$, provided that $A \neq \emptyset$.

Taking $a, b \in A$, let us compute $h''(\varepsilon)$.

$$h''(\varepsilon) = \frac{\partial}{\partial \varepsilon} \left(\frac{\partial H_e}{\partial x_k} \frac{\partial x_k}{\partial \varepsilon} \right) = \sum_{k\in D} (\log x_k(\varepsilon) + 1)' \cdot (a_k - b_k) = \sum_{k\in D} \frac{-(b_k - a_k)^2}{a_k + \varepsilon(b_k - a_k)} \tag{320}$$

Obviously, $h''(\varepsilon) < 0$ for any $\varepsilon \in [0,1]$. By 3.1.17, h'' is a strictly concave function. This holds for any $a, b \in A$, implying that there cannot be minima nor saddle points there (a saddle point is concave along certain directions and convex along the others). ▷ 43

It also means that the maximum is unique. Imagine that it was not. Then there would be at least two local maxima a and b. Taking the entropy along the line segment $\{a + \alpha(b-a) \mid \alpha \in [0,1]\}$ as a function of α, we would get strictly concave function of single variable with two maxima, which would be in contradiction with the following.

4.12.8 Observation *Strictly concave function $f : J \to \mathbb{R}$ on interval $J \subseteq \mathbb{R}$ has at most one maximum.*

Proof Maximum is a point $x \in J$ such that for small $\varepsilon > 0$ and for any $\gamma \in (0, \varepsilon)$: $f(x + \gamma) < f(x)$. Having two maxima x and y, assume that $f(x) \leq f(y)$. As f is concave it lies above the chord going from x to y. But already that chord is $\geq f(x)$, which contradicts the condition for maximum in x. Q.E.D.

So we have proved that the entropy achieves unique maximum[42] (if it exists[39]), having no saddle points nor minima on A. Therefore, (318) has a single solution, representing the maximum. The first line of (318) may be rewritten as follows.

$$p_k = \exp\left(\lambda_1 - 1 + \lambda_2 k + \lambda_3 k^2\right) = e^{\lambda_1 - 1 - \beta\gamma^2 + 2\beta\gamma k - \beta k^2} = \alpha e^{-\beta(k-\gamma)^2} \tag{321}$$

[42] In fact we have proved pretty general theorem that can be appreciated once we extend the notion of convexity/concavity to functions of multiple variables as it is done in the following definition. It was demonstrated in (320) that the entropy is strictly concave in the sense of this definition.

4.12.9 Definition The function $f : X \to \mathbb{R}$ is called *strictly concave* iff $X \subseteq \mathbb{R}^N$ is a convex set and the functions $\lambda\alpha \in [0,1]. f(x + \alpha(y - x))$ are strictly concave for all choices of $x, y \in X$.

4.12.10 Theorem *Strictly concave $f : X \to \mathbb{R}$ achieves at most one maximum on convex set X.*

4.12.11 Theorem *Entropy $H : P \to \mathbb{R}$, where $H(p) = -\sum_{k=1}^D p_k \log_2 p_k$ and $P = \{p \in [0,1]^D \mid p_\bullet = 1\}$ is strictly concave function.*

where $\beta := -\lambda_3$, $\gamma := \lambda_2/(2\beta)$ and $\alpha := \exp(\lambda_1 - 1 - \beta\gamma^2)$. For positive β, sampled version of Gaussian distribution can be recognized in this result. Note, however, that β can be zero or even negative if σ^2 was too big for a given B and the only way to achieve such a variance was uniform or even U-shaped distribution, respectively. Plugging (321) back to (318) we obtain the following system of equations parametrized by σ and μ, the solution of which yields to α, β and γ.

$$\sum_{k\in D} \alpha e^{-\beta(k-\gamma)^2} = 1 \qquad \sum_{k\in D} \alpha k e^{-\beta(k-\gamma)^2} = \mu \qquad \sum_{k\in D} \alpha k^2 e^{-\beta(k-\gamma)^2} = \sigma^2+\mu^2 \quad (322)$$

Unfortunately, I was unable to solve it analytically as I failed to evaluate the sums involved. Nevertheless it is possible to find an upper bound on the entropy, at least. Let us do it for $\mu = 0$. This is easier as it implies[43] $\gamma = 0$. It also solves the original question of the most entropic distribution when only σ is specified — this follows from the following lemma[44].

4.12.12 Lemma *A probability distribution p defined by (321) on a set $D := \mathbb{Z} \cap [-B, B]$ having a variance $\sigma^2(p)$ and mean $\mu(p) \neq 0$ can be modified into a distribution q such that $H(q) > H(p)$, $|\mu(q)| < |\mu(p)|$ and $\sigma(q) = \sigma(p)$.*

Proof We can assume $\mu(p) > 0$ since the case of negative μ would analogical. Then there are two cases there. Either the β of (321) is negative or it is positive (it cannot be zero as it would imply $\mu(p) = 0$).

For $\beta < 0$, we define $q_k := p_k$, except q_B and q_{-B} that will be set to $(p_{-B} + p_B)/2$. Then $\mu(q) = \mu(p) - B(p_B - p_{-B})$, which makes the mean smaller because $p_B > p_{-B}$, which in turn is a consequence of having $\mu(p) > 0$. Moreover, $|\mu(q)| < |\mu(p)|$ because $\mu(p) = \Sigma + B(p_B - p_{-B})$, where $\Sigma \geq 0$. The entropy has increased by

$$-p_{-B}\log_2 p_{-B} - p_B \log_2 p_B + (p_{-B} + p_B)\log_2 \frac{p_{-B} + p_B}{2} > 0 \quad (323)$$

42 ◁ where the inequality comes from $\lambda x. -x\log_2 x$ being a concave function (cf. (66)). Finally, the variance has increased too because $\sigma^2(q) = \sum_k k^2 q_k - \mu^2(q) = \sum_k k^2 p_k - \mu^2(q) = \sigma^2(p) + \mu^2(p) - \mu^2(q)$ and $\mu(q) < \mu(p)$. But this increase can be corrected by smoothing q with a uniform distribution that would make the new variance equal to $\sigma^2(p)$, as desired. The smoothing would increase the entropy even more (because the uniform distribution has maximal entropy on $\mathbb{Z} \cap [-B, B]$) while pushing the mean closer to zero, so these two properties would not be spoiled by the operation.

Proof H is concave on every line from a to b for $a, b \in P \cap (0,1)^D$ as was shown in (320). For $a, b \in P$, we can first discard dimensions for which $a_k = b_k = 0$ as these do not contribute the entropy. In the reduced dimension, a and b still may have some of their coordinates zero. But a minute movement along the line that is joining them brings us into the interior of the simplex where we already know the theorem holds. Due to continuity of H, it holds even on its border. Q.E.D.

[43] It is a simple exercise to check that for $\mu = \gamma = 0$ there exist α and β solving (322) for any $\sigma > 0$. Due to already proven uniqueness of the maximum and absence of other stationary points on A, there cannot be any other triple $\langle \alpha, \beta, \gamma \rangle$ that would solve (322), hence γ must be 0 once μ is.

[44] To simplify notation, let us write $\sigma(p)$ for $\sigma(X)$ where p is a probability distribution on interval of $\mathbb{Z}$ and X is a random variable such that $\Pr(X = k) = p_k$. We already use this notation for entropy.

As for $\beta > 0$, let us first prove it for $\mu \in (0, 1]$. Let the displacement vector v be defined by setting $v_0 := 3$, $v_1 := -2$, $v_{-1} := -1$ and zero otherwise. For small $t > 0$, the vector $p + vt$ is still a probability distribution because $v_\bullet = 0$ and thanks to (321) which assures that $p_k > 0$. Let us show that $H_e(p + vt) > H_e(p)$, $\mu(p + vt) < \mu(p)$ and $\sigma^2(p + vt) < \sigma^2(p)$ for small enough $t > 0$. The proof would be finished by taking $q_k := (1-\alpha)(p_k + v_k t) + \alpha$, where $\alpha \in (0, 1)$ is a smoothing factor making $\sigma^2(q) = \sigma^2(p)$.

From the *mean value theorem* we have $\zeta \in (0, 1)$ such that

$$H_e(p + vt) - H_e(p) = (\nabla H_e(p + \zeta tv))^{\mathrm{T}} vt = (\nabla H_e(p))^{\mathrm{T}} vt + \varepsilon t \quad (324)$$

Where the last equality follows from continuousness of ∇H_e for $p_k > 0$, where ε is roughly proportional to t. For small t it is therefore negligible and it is enough to prove that $(\nabla H_e(p))^{\mathrm{T}} v > 0$ to show the entropy increase. So we get

$$\begin{aligned}(\nabla H_e(p))^{\mathrm{T}} v &= 3\log p_0 - 2\log p_1 - \log p_{-1} = -2\log\frac{p_1}{p_0} - \log\frac{p_{-1}}{p_0} \\ &= 2\beta((1-\gamma)^2 - \gamma^2) + \beta((1+\gamma)^2 - \gamma^2) = 3\beta - 2\gamma\beta \geq \beta > 0\end{aligned} \quad (325)$$

where β and γ came from (321) and the inequality followed from $\gamma \leq 1$ and $\beta > 0$. Applying this kind of argument to μ and σ^2 as well, we get

$$\begin{aligned}(\nabla\mu(p))^{\mathrm{T}} v &= v_1 - v_{-1} = -2 - (-1) = -1 < 0 \\ (\nabla\sigma^2(p))^{\mathrm{T}} v &= v_1 + v_{-1} - 2\mu(v_1 - v_{-1}) = -3 + 2\mu \leq -1 < 0\end{aligned} \quad (326)$$

where the last inequality used the fact that $\mu \leq 1$.

For $\mu > 1$, define $r_k := p_k + 1$ except for r_B, for which $r_B := p_{-B}$ has to be used. Obviously, the entropy is unaffected by this, as it is invariant to reordering. It is straightforward to check that $\mu(r) = \mu(p) + (2B+1)p_{-B} - 1$. Since $\beta > 1$ and $\mu > 1$ the p_{-B} conforming to (321) is strictly lower than other p_k's. Therefore, $(2B+1)p_{-B} < 1$. If it was not so, $p_\bullet$ would exceed 1. So we managed to decrease the mean. Let us look at variance now. Expanding the sum $\sum_k k^2 r_k$ we get $\sigma^2(r) + \mu^2(r) = \sigma^2(p) + \mu^2(p) - 2\mu(p) - (2B+1)p_{-B} + 1$. This can be further rewritten into $\sigma^2(r) = \sigma^2(p) - (2B+1)(2\mu(p) - 1)p_{-B} - (2B+1)^2 p_{-B}^2$, hence $\sigma(r) < \sigma(p)$. By taking $q := (1-\alpha)r + \alpha$ for certain small $\alpha > 0$, we get the final distribution. Q.E.D.

Therefore, if $\mu \neq 0$, the lemma assures that it can be made smaller, while increasing the entropy and without changing the variance. That would mean that the distribution in question did not have maximal entropy in the set of all distributions of the given variance. That is why the maximizing distribution must be of $\mu = 0$.

Knowing this, let us express the entropy now (note that $H_e = H\log(2)$).

$$H_e(B, \sigma) = -\sum_{k\in D} p_k \log\left(\alpha e^{-\beta k^2}\right) = -\log\alpha + \sum_{k=-B}^{B} p_k k^2 \beta = \sigma^2\beta - \log\alpha \quad (327)$$

Note that it depends on two numbers, B and σ — these determine α and β via the following equations that were derived from (322).

$$\alpha = \left(\sum_{k=-B}^{B} e^{-\beta k^2}\right)^{-1} \qquad \sum_{k=-B}^{B} (k^2 - \sigma^2) e^{-\beta k^2} = 0 \quad (328)$$

130 ◁ **4.12.13 Observation** *The entropy of solutions of equations (321) and (328) for B and $B+1$ satisfies $H(B+1,\sigma) > H(B,\sigma)$.*

Proof Suppose $H(B+1,\sigma) < H(B,\sigma)$. Taking the distribution p of the right side with additional $p_{B+1} := p_{-B-1} := 0$ gives a distribution on $[-B-1, B+1] \cap \mathbb{Z}$ with higher entropy than the supposed maximum $H(B+1,\sigma)$. That is why this case cannot happen.

If $H(B+1,\sigma) = H(B,\sigma)$ was the case, then we would have two different distributions achieving maximal entropy $H(B+1,\sigma)$ on $[-B-1, B+1] \cap \mathbb{Z}$. The first one prescribing $p_{B+1} > 0$ according to (321) and the second one being a zero padded version of the distribution corresponding to the right side. But this is impossible as we proved earlier that the solution is unique for fixed B and σ and $\mu := 0$. That is why the third possibility of $H(B+1,\sigma) > H(B,\sigma)$ holds. Q.E.D.

The uniqueness of the most entropic distribution on $[-B, B] \cap \mathbb{Z}$ assures uniqueness of the solution $\langle \alpha_B, \beta_B \rangle$ of (328) for every single choice of σ and B. Specially, it means that the second equation (which only contains β_B) has a unique solution for every $B \in \mathbb{N} \setminus \{0\}$. I will show in the following that the numbers β_B have a limit β for $B \to \infty$ and that this limit also solves (328), thereby leading to a distribution p on $\mathbb{Z}$ with a variance of σ^2. The entropy of this distribution will then be used to define a meaning of $H(\infty,\sigma)$. Then, after showing that $\lim_{B\to\infty} H(B,\sigma) = H(\infty,\sigma)$ it will be clear from 4.12.13 that p is the distribution on $\mathbb{Z}$ whose entropy forms an upper bound on entropy of any finite distribution with a given variance[45].

Let us show that the limit $\beta := \lim_{B\to\infty} \beta_B$ exists and that it solves the second equation of (328) for $B = \infty$. First, note that $f_n(\beta) := \sum_{k=-n}^{n} (k^2 - \sigma^2) e^{-\beta k^2}$ is a continuous function for any $n \in \mathbb{N}$. Let f_n be regarded as $f_n : [a,b] \to \mathbb{R}$, where $a, b > 0$ will be determined later. The sequence of functions f_n converges absolutely and uniformly[46] because for any $x \in [a,b] : |f_n(x) - f_{n-1}(x)| < c_n := 2(\sigma^2 + n^2) e^{-n^2 a}$ and c_n is convergent. Let the limit, which we now know is itself continuous, be denoted by f. As everything happens on compact space $[a,b]$, we also have f and f_n uniformly continuous. Moreover, as the derivatives f_n' are continuous and $|f_n'(x) - f_{n-1}'(x)|$ can be upper-bounded by $2n^2(\sigma^2 + n^2) e^{-an^2}$, which is convergent, we obtain that the sequence f_n' has a limit g and that this limit equals to f' (when considering that $f = \lim_n f_n$). It also implies that that f is differentiable and that f' is continuous (uniformly on $[a,b]$).

The numbers a and b are to be selected as follows. It is easy to see that there is a positive solution of $f_{n_0}(a) = 0$ for sufficiently large n_0. This defines the value of a. Note that the root of f_n for $n > n_0$ is no less than a. On the other hand, b has to be taken so high that $f_n(b) < 0$ for all n. This is possible because high b makes the gaussian packed closely around 0 so that only the zeroth term of the sum remains relevant. This way, we have the root of each f_n bracketed between a and b for any $n > n_0$ because $f_n(a+\varepsilon) > 0 > f_n(b-\varepsilon)$ for small $\varepsilon > 0$. It gives $f(a+\varepsilon) \geq 0 \geq f(b-\varepsilon)$ in the limit, hence f has at least one root on (a,b).

The set of f's roots is discrete because f is *holomorphic* on $\mathbb{C} \setminus \{0\}$ when regarded as a complex function (if there was an interval on which it would be zero it would have to be zero everywhere). So there must be at least one $\beta \in (a,b)$ s.t. $f(\beta) = 0$ and $\delta_0 > 0$ s.t. $\forall \delta \in (0, \delta_0) : f(\beta - \delta) > 0 > f(\beta + \delta)$. Using uniform convergence we can find n_0 s.t. $|f_n(x) - f(x)| < \varepsilon := \frac{1}{2} \min(|f(\beta-\delta)|, |f(\beta+\delta)|)$ for $x \in [\beta - \delta, \beta + \delta]$ and $n > n_0$. In another words, all f_n's for $n > n_0$ stay inside ε-belt around f in the δ-neighborhood of β. As this belt crosses the zero line, every root β_n of f_n lies inside $(\beta - \delta, \beta + \delta)$. As δ was arbitrarily small, we see that the limit $\lim_n \beta_n$ exists and that it converges to one of the roots of f. There is only single root of f which crosses the zero line because each f_n has a unique root. However, we did not excluded the possibility of having roots of f that would only touch the zero-line from above or from below. Nevertheless, these hypothetic cases are not important here as we only needed the limit.

All that remains to be shown is that $\lim_n H_e(n,\sigma) = H_e(\infty,\sigma)$. The left side rewrites into $\sigma^2 \beta_n + \log \sum_{k=-n}^{n} e^{-\beta_n k^2}$, where the fisrt term has a limit of $\sigma^2 \beta$. The second term can be written as $s_n := 1 + 2 \sum_{k=1}^{n} e^{-\beta_n k^2}$. Now

$$d_n := \left| s_n - 1 - 2 \sum_{k=1}^{\infty} e^{-\beta k^2} \right| \leq 2 \left| \sum_{k=n+1}^{\infty} e^{-\beta k^2} \right| + 2 \sum_{k=1}^{n} \left| e^{-\beta_n k^2} - e^{-\beta k^2} \right| \quad (330)$$

As the exponential falls towards zero, we can take n so high that the first absolute value would be less than $\varepsilon/4$. The second one can be bounded from above by $n \left| e^{-\beta_n} - e^{-\beta} \right|$ which, again, can be made smaller than $\varepsilon/4$ because of continuousness of the exponential and $\lim_n \beta_n = \beta$. Hence $d_n < \varepsilon$ for large enough n and the limit is therefore equal to $\sigma^2 \beta + \log \sum_{k=-\infty}^{\infty} e^{-\beta k^2} = H_e(\infty,\sigma)$.

So we have proved that on $\mathbb{Z}$ there really exists a distribution with variance σ^2 and entropy $H(\infty,\sigma)$ s.t. $H(\infty,\sigma) > H(B,\sigma)$ for any finite B. As such, it constitutes an upper bound on any finite distribution. Let us estimate it for σ being much larger than 1. That would make β much smaller than 1, allowing the following integral approximation. Using $\sum_{k=-\infty}^{\infty} e^{-\beta k^2} \approx \int_{-\infty}^{\infty} e^{-\beta x^2} dx = \sqrt{\pi/\beta}$ in (328) gives $\beta \approx (2\sigma^2)^{-1}$, which leads to the following entropy estimate (G stands for *Gaussian*).

$$H(G) = \frac{\sigma^2 \beta}{\log(2)} + \log_2 \sum_{k=-\infty}^{\infty} e^{-\beta k^2} \approx \log_2(e) + \log_2 \sqrt{\frac{\pi}{\beta}} = \log_2 \sqrt{2\pi e \sigma^2} \approx \log_2(4.133\,\sigma) \quad (331)$$

Analogically we can find maximal entropy distribution among those that have $p_k = 0$ for $k < 0$ and fixed mean $\mu := \sum k p_k$. Let us do it only informally, without proper treatment of infinite spaces involved, as this would be analogical to the just demonstrated case of Gaussian distribution.

Lagrange multiplier method (informally used on $\mathbb{N}$) gives

$$\begin{aligned} -\log p_k - 1 + \lambda_1 + \lambda_2 k = 0 \qquad & \text{for } k \in \mathbb{N} \\ \sum\nolimits_{k\in\mathbb{N}} p_k = 1 & \\ \sum\nolimits_{k\in\mathbb{N}} k p_k = \mu & \end{aligned} \quad (332)$$

[45] Note, however, that this argument does not yet prove that p is the most entropic distribution of given variance on $\mathbb{Z}$ — all that it states is that p is an upper-bound on any finite distribution.

[46] The sequence of functions $f_n : X \to \mathbb{R}$ is said to *converge uniformly* towards a function f iff

$$\forall \varepsilon > 0 : \exists n_0 : \forall n > n_0 : \forall x \in X : |f_n(x) - f(x)| < \varepsilon \quad (329)$$

It is a stronger condition than ordinary pointwise convergence ($\forall x \in X : f(x) = \lim_{n\to\infty} f_n(x)$), making f continuous if all f_ns were continuous (contrary to the pointwise case). It can be easily proved that f_n converge uniformly if there are numbers b_n s.t. $\sum b_n$ converges and $\forall n \forall x \in X : |f_{n+1}(x) - f_n(x)| < b_n$.

This leads to exponential distribution $p_k = \alpha e^{-\beta k}$, where α and β come from the last two equations in the following way.

$$\begin{aligned} p_\bullet &= \sum\nolimits_{k\in\mathbb{N}} \alpha \left(e^{-\beta}\right)^k = \alpha(1-e^{-\beta})^{-1} = 1 \\ \mu &= \sum\nolimits_{k\in\mathbb{N}} k\alpha e^{-\beta k} = \frac{1-e^{-\beta}}{\left(e^{\beta/2}-e^{-\beta/2}\right)^2} = \frac{1}{e^\beta - 1} \end{aligned} \tag{333}$$

So we have $\beta = \log(1+1/\mu)$ and $\alpha = (1+\mu)^{-1}$. This implies that the exponential distribution (designated by E) and its variance are given as follows[47].

$$p_k = \frac{1}{1+\mu} e^{-k\log(1+1/\mu)} \qquad \mathrm{Var}(E) = \frac{e^\beta}{(e^\beta-1)^2} = \mu^2 + \mu \tag{335}$$

This leads to the following entropy.

$$H(E) = -\log_2 \frac{1}{1+\mu} + \sum_{k\in\mathbb{N}} p_k \frac{k}{\log 2} \log\left(1+\frac{1}{\mu}\right) = \log_2\left((1+\mu)(1+1/\mu)^\mu\right) \tag{336}$$

Noting that $\mu = -1/2 + \sqrt{1/4 + \mathrm{Var}(E)}$, we can express the entropy in terms of $\sigma = \sqrt{\mathrm{Var}(E)}$ so that it could be compared with (331). Let us do it for large μ.

$$H(E) \approx \log_2(e(1+\mu)) = \log_2\left(\frac{e}{2} + e\sqrt{1/4+\sigma^2}\right) \approx \log_2(2.7183\,\sigma) \tag{337}$$

Since the exponential distribution is the most entropic one on $\mathbb{N}$, it can be used to
49 ◁ improve the upper bound of lemma 3.2.11, as follows.

$$H \le H(E) = \log_2\left((1+\mu)(1+1/\mu)^\mu\right) < \log_2\left(e\,(1+\mu)\right) = \log_2\left(e\,\mathcal{E}(N)\right) \tag{338}$$

Note that $\mathcal{E}(N) = \mu + 1$ because the variable N has beed defined to take values starting from 1 in 3.2.11 instead of 0.

Finally, let us investigate the entropy of the uniform distribution of given variance. Uniform distribution on $[-N, N] \cap \mathbb{Z}$ has an entropy of $\log_2(2N+1)$ and the following variance.

$$\mathrm{Var}(U) = \sum_{k=-N}^{N} \frac{k^2}{2N+1} = \frac{N(N+1)}{3} \tag{339}$$

Wherefore we get $N = \sqrt{1/4 + 3\sigma^2} - 1/2$ and consequently $H(U) = \log_2\sqrt{1+12\sigma^2}$ which is about $\log_2(3.464\,\sigma)$ for large $\sigma(U)$. This is less than (331), as expected. Nevertheless, $H(U)$ is maximal among distributions with zero p_k for $k \in \mathbb{Z} \setminus [-N, N]$.

It seems from these results that the entropy of any distribution on $\mathbb{Z}$ comes out close to $\log_2(K\sigma)$ for large σ, where the constant K depends on the distribution. It is indeed true and I will demonstrate it for zero-mean distributions.

[47] The following sums come in handy during calculation of the variance.

$$\sum_{k\in\mathbb{N}} k^2 e^{-\beta k} = \frac{1+e^{-\beta}}{\left(1-e^{-\beta}\right)^2\left(1-e^{\beta}\right)} \qquad \sum_{k\in\mathbb{N}} k e^{-\beta k} = \frac{1}{\left(1-e^{-\beta}\right)\left(e^\beta-1\right)} \tag{334}$$

4.12.14 Theorem *Consider bounded non-negative piecewise continuous function $f : \mathbb{R} \to \mathbb{R}$ such that:*

$$\int_{\mathbb{R}} f(x)\,dx = 1 \quad \int_{\mathbb{R}} x f(x)\,dx = 0 \quad \int_{\mathbb{R}} x^2 f(x)\,dx = 1 \quad \int_{\mathbb{R}} f(x)\log f(x)\,dx = \log\frac{1}{K} \tag{340}$$

This would serve as a template from which the distribution is to be derived as follows, using constant α to control the width.

$$p_k := \frac{f(\alpha k)}{\sum_{n\in\mathbb{Z}} f(\alpha n)} \tag{341}$$

Then the entropy of that distribution is $H \approx \log_2(K\sigma)$, for large variance σ^2.

Proof Obviously, for large width (i.e. small α), we have

$$\sum_{n\in\mathbb{Z}} f(\alpha n) \approx \int_{\mathbb{R}} f(\alpha x)\,dx = \frac{1}{\alpha} \quad \text{and} \quad \sigma = \sqrt{\sum_{n\in\mathbb{Z}} n^2 \frac{f(\alpha n)}{\sum_{k\in\mathbb{Z}} f(\alpha k)}} \approx \frac{1}{\alpha} \tag{342}$$

because $\sum_{n\in\mathbb{Z}} n^2 f(\alpha n) \approx \int_{\mathbb{R}} x^2 f(\alpha x)\,dx = \alpha^{-3}$. Now

$$\begin{aligned} H &= -\sum_k \frac{f(\alpha k)}{\sum_n f(\alpha n)} \log_2 \frac{f(\alpha k)}{\sum_n f(\alpha n)} = \frac{-1}{\sum_n f(\alpha n)}\left(\sum_k f(\alpha k)\log_2 f(\alpha k)\right. \\ &\quad \left. + \sum_k f(\alpha k)\log_2 \frac{1}{\sum_n f(\alpha n)}\right) \approx -\frac{1}{\sigma}\int_{\mathbb{R}} f(\alpha x)\log_2 f(\alpha x)\,dx - \log_2\sigma \\ &= -\frac{1}{\alpha\sigma}\int_{\mathbb{R}} f(x)\log_2 f(x)\,dx + \log_2\sigma = -\log_2\frac{1}{K} + \log_2\sigma = \log_2(K\sigma) \end{aligned} \tag{343}$$

Q.E.D.

Practical importance of this result stems from easier estimation of variance than entropy from the data. Knowing the underlining distribution's constant K, it is enough to estimate its σ from several hundreds of its samples to obtain estimate on the number of bits each sample carries.

Sometimes, we don't even need to known the constant K. For instance, assuming that $f(x)$ represents a distribution of instant amplitudes of an electric signal coming from a microphone, it is possible to compute how many bits of information will be lost by positioning the microphone farther from the sound source — such a withdrawal changes the variance from the nearer σ_1^2 one to the farther σ_2^2 one, leaving the distribution type represented the template function f intact[48]. Hence the constant K does not change and we get the amout of lost information to be

$$\log_2 K\sigma_1 - \log_2 K\sigma_2 = \log_2\frac{\sigma_1}{\sigma_2} \tag{344}$$

[48] Assuming we can neglect echo, possible intersample correlations and thermal noise.

Note that this assymptotic result of theorem 4.12.14 requires both $K\sigma_1$ and $K\sigma_2$ to be much higher than 1. Unfortunately, K may be anywhere near to zero[49]. Nevertheless, distributions whose template function satisfies $\int f^2(x)\,dx \le 1$ behave well, having $K \ge 1$. Because Gaussian distribution achieves maximum entropy for a fixed variance, $K \le 4.333$. To conclude the introductory part of this section, let us summarize its results in the following table.

Condition	Distribution	Entropy
Maximum entropy, given fixed σ	Gaussian	$H(G) \approx \log_2(4.333\,\sigma)$
Maximum entropy on finite contiguous subset of $\mathbb{Z}$	Uniform	$H(U) = \log_2\sqrt{1+12\sigma^2}$ $\approx \log_2(3.464\,\sigma)$
Maximum entropy of one-sided distribution with fixed variance	Exponential	$H(E) = \left(\beta+\frac{1}{2}\right)\log_2\left(\beta+\frac{1}{2}\right) - \left(\beta-\frac{1}{2}\right)\log_2\left(\beta-\frac{1}{2}\right)$ where $\beta = \sqrt{1/4+\sigma^2}$ $\approx \log_2(2.718\,\sigma)$
General distribution shaped after $f(x)$ satisfying (340) and $K\sigma>1$	Sampled $f(x)$	$H \approx \log_2(K\sigma)$

Now we can move to the main part of this section.

4.12.15 Information in Quantized Signal

We cannot talk about information of any definite signal, just as it does not have much sense to ask how much information there is in the letter A, or in certain fixed sentence. The entropy (and mutual infromation) becomes defined only after this sentence or signal can be considered as an alternative to other signals from an ensemble, which would allow definition of probabilities.

4.12.16 Definition *The set $\sigma_Q(B)$*

The set of *B-bounded integer signals* $\sigma_Q(B)$ is defined as $\{s\in\sigma \mid \forall n : s_n\in\mathbb{Z}\cap[-B,B)\}$. We shall drop the actual bound B when it follows from the context, writing just σ_Q.

4.12.17 Note Note that $\sigma_Q(B)\subset\sigma_F$ because for every $x\in\sigma_Q$ the energy $||x||^2$ must be defined and finite, according to 4.2.21. But with $x_k\in\mathbb{Z}$, it would not be possible with infinitely many $x_k\neq 0$. Thus $x\in\sigma_F$. 82 ◁

[49] Consider for example $f(x)$ equal to

$$f(x) = \frac{1}{\sqrt{12-3a^2}-3a} \quad \text{for} \quad x\in\left(\frac{a}{2}-\sqrt{3-\frac{3}{4}a^2},-a\right)\cup\left(a,\sqrt{3-\frac{3}{4}a^2}-\frac{a}{2}\right) \tag{345}$$

and zero otherwise. This makes the last formula of (340) negative for a close to 1. In fact its limit is $-\infty$ for $a\to 1$, which translates as zero limit for K. There even exist unimodal distributions with negative $\log K$.

As a consequence of 4.12.17, the set $\sigma_Q(B)$ is countable[50] so it lends itself to be used as domain Ω for a probability space on which we could define probabilities of the respective signals and then compute entropy of this distribution according to (75). This entropy is the lower bound on the average number of bits that would be needed to encode signals drawn from this distribution or — when viewed from the communication perspective — it is the upper bound on the average number of bits that could be sent in one signal along the lossless channel, provided that the source generates the signals according to the aforementioned distribution. ▷ 46

Unfortunately, this is of little practical benefit as it is not clear where to get so many probabilities. For this reason we shall only consider small blocks of samples, assuming that the probabilistic model of complete signal could be glued from these blocks similarly to the way we built the models of words from models of phonemes in 2.4. In general, the blocks could be of different probability distribution in different places of the signal they make up. We will start with the simplest case of single sample blocks, making them longer later. To make various block lengths comparable, the entropy per single sample will be calculated in all cases. The signals that we consider possible (i.e. with non-zero probability, before probability smoothing) are of distict beginning and end between which the signal will be of rather similar power. Simply speaking the signals we are interested in look more like an audio recording of a song than, say, a quantized version of $\lambda n.10^8\cdot\mathrm{sinc}(n/10)$. This allows us to use single or few probabilistic models of the block throughout the entire signal. ▷ 20

Another problem we have to cope with is the noise. To make things practically manageable, let us assume that the noise is always additive, i.e. that the random variable X representing the signal block gets corrupted by the noise N as $Y := X+N$, which is the variable we measure. For single-sample blocks (that is for simple numbers) we have the following observation.

4.12.18 Observation *For random variables $X:\Omega\to\mathbb{Z}$, $N:\Omega\to\mathbb{Z}$, the following holds for the probability of their sum $Y := X+N$.*

$$\Pr_\Omega(\text{“}Y=y\text{”}) = \sum_{x\in\mathbb{Z}} \Pr_\Omega(\text{“}X=z \;\&\; N=y-x\text{”}) \tag{346}$$

For independent N and X we additionally have

$$\lambda y.\Pr_\Omega(\text{“}Y=y\text{”}) = x * n \tag{347}$$

where the probability distributions are treated like signals so $x := \lambda x.\Pr(\text{“}X=x\text{”})$ and $n := \lambda n.\Pr(\text{“}N=n\text{”})$. In another words, the distribution of sum of independent variables equals to the convolution of distributions of the addends.

Proof The proposition $X{+}N{=}y$ is equivalent to $\exists x\in\mathbb{Z} : X{=}x \;\&\; N{=}y{-}x$, which corresponds to the set $M_y := \bigcup_{x\in\mathbb{Z}} S_{xy}$, where $S_{xy} := \{\omega\in\Omega \mid X(\omega){=}x \;\&\; N(\omega){=}y{-}x\}$. As S_{xy} are pairwise disjoint we can apply additivity axiom (58) to get $\Pr_\Omega(M_y) = \sum_{z\in\mathbb{Z}}\Pr_\Omega(S_k)$, which proves the first part. For independent X and Y we have $\Pr_\Omega(\text{“}X=z \;\&\; N=y-x\text{”}) = \Pr_\Omega(\text{“}X=z\text{”})\Pr_\Omega(\text{“}N=y-x\text{”})$ which transforms (346) into a convolution. Q.E.D. ▷ 39

[50] Even $\bigcup_{b=0}^{\infty}\sigma_Q(b)$ is still countable.

55 ◁ According to 3.2.24, the amount of information about X that can be recovered from $X+N$ is given by

$$I(X+N;X) = H(X) + H(X+N) - \overbrace{H(X+N,X)}^{H(X,N)} = H(X+N) - H(N|X) \quad (348)$$

which reduces to $I(X+N;X) = H(X+N) - H(N)$ in case of independent X and N. For general (dependent or independent) X and N, it leads to the following lower bound.

$$I(X+N;X) = H(X+N) - H(N|X) \geq H(X+N) - H(N) \quad (349)$$

Note that (349) also holds for blocks of samples and even for entire signals as it did not assume anything special about X and Y except that $\langle X+N, X\rangle$ is in a one-to-one correspondence with $\langle N, X\rangle$. From above, $I(X+N;X)$ is bounded by[51]

$$H(X) + H(N) - H(N|X) = H(X) - I(X;N) \quad (350)$$

4.13 Implementing Digital Filters

In this section I will address practically important case of real valued filter, processing real valued signal. Let the impulse response of the filter in question be denoted by h. Short FIR filters can be implemented directly. In common special case of $h = \rho h$ or $h = -\rho h$ we can save half of the multiplications by collecting h_k in $h_k x_{n-k} \pm h_k x_{n+k}$, obtaining $h_k(x_{n-k} \pm x_{n+k})$. For longer FIR filters, however, faster FFT method is more appropriate.

4.13.1 FFT Method for FIR Filters

74 ◁ Using the convolution lemma 4.1.8 we are able to evaluate FIR filter with impulse response h of length $M_h + 1$ in time $\mathcal{O}(\log(M_h))$ operations per sample, whereas the direct implementation would take $\mathcal{O}(M_h)$ operations per one processed sample.

First, we have to chunk the input signal into pieces of M_s+1 samples, zero-padding them to up the length $N = M_h + M_s + 1$, which has to be a power of two. After that, 4.1.8 could be applied to each chunk, leading to correctly computed convolution at indices 0 to M_s, followed by a wrap-around area, that has to be discarded. By doing so and by pasting all the blocks together afterwards, we obtain the desired output. Note that h_n was considered non-zero only for $n \in [0, M_h]$. For best speed, $N := 2^n$ should be selected to minimize average work per output sample, that is $\frac{N \log(N)}{N - M_h}$. Note that actual formula to be minimized could be more complicated if the effect of caches would be accounted for.

Depending on CPU used, this method typically outperforms direct implementation for, say, $M_h > 100$. Apart from great advantage of speed, it however has a disadvantage of block processing, which introduces artificial delay. This may be an issue in low-latency applications.

47 ◁
54 ◁ [51] Using $H(f(A,B)) \leq H(A) + H(B)$, which follows from 3.2.8 and 3.2.20.

4.13.2 Implementing Recursive Filters

We have to be more careful when implementing recursive filters because of the limited precision of computer's arithmetics, which causes that the filter does not behave exactly as the theory would predict. This can lead to imprecise (noisy) results or even to instability, in some cases. This is because the output of the filter is fed back to its input, which can lead to resonant amplification of the error.

There are two sources of precision loss. First, the filter's coefficients d are stored only up to the machine's precision. Usually, we design the filter using zero/pole representation, transforming it into c and d afterwards. Round-off errors of this transformation can move poles past the unit circle, creating an unstable filter.

The second source of errors is due to limited precision of arithmetical operations used at the runtime. Namely, multiplication generates twice as many significant digits each of its operand had. The extra digits are rounded-off, and may be regarded as a noise signal, which, when added to the rounded output would give the theoretical value. This noise creeps into the feedback of the filter where it could be amplified to an observable amplitude.

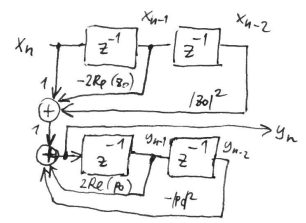


Fig. 20. *Block diagram of (352)*

These two problems make original formula (194) practically unusable even for moderate values of M. Fortunately, decomposing the filter into serial connection of single-pole and single-zero filters as in (208) solves the problem. In this case the transformation from poles to d is trivial and feedbacks are short and local, therefore more predictable. The only disadvantage is that it uses complex numbers, even if we only need to process reals. But this can be corrected. As h was real, we know that the associated z-polynomials of (207) had real coefficients, which implies that the poles appear in complex conjugated pairs except in case of a pole laying on a real axis. The conjugated pair lead to a filter with two real coefficients d_1 and d_2, which is still easily realizable. The big filter can then be built from these small sections, serially connected in agreement with (208).

▷ 86 ▷ 90 ▷ 89

It is often practical to have as many poles as zeroes. Let's constrain ourselves to this kind of filters only, in the rest of this subsection. Let us also suppose that the filter is zp-canceled and that all its zeroes and poles have non-zero imaginary part. This way, we would only need two-coefficient sections as our building blocks. Additionally, these could be grouped such that each group will contain exactly one zero-section and one pole-section. These groups will be our universal (for this class of filters) building blocks, and I will refer to them as to *blocks*. A block belonging to a zero z_0 (or more precisely to a zero pair z_0 and $\overline{z_0}$) and pole p_0 has the following transfer function.

$$H(f) = \frac{(z - z_0)(z - \overline{z_0})}{(z - p_0)(z - \overline{p_0})} = \frac{z^2 - 2\,\mathrm{Re}(z_0)z + |z_0|^2}{z^2 - 2\,\mathrm{Re}(p_0)z + |p_0|^2} \quad (351)$$

Comparing it with (207) we can reconstruct c and d, getting $c_0 = 1$, $c_1 = -2\,\mathrm{Re}(z_0)$, $c_2 = |z_0|^2$ and $d_0 = 1$, $d_1 = -2\,\mathrm{Re}(p_0)$, $d_2 = |p_0|^2$, which leads to the following filter:

$$y_n = x_n - 2\,\mathrm{Re}(z_0)x_{n-1} + |z_0|^2 x_{n-2} + 2\,\mathrm{Re}(p_0)y_{n-1} - |p_0|^2 y_{n-2} \quad (352)$$

Engineers like to draw this kind of equations using wires and boxes so they would feel safe in their world of circuit diagrams. Our equation would be represented by the drawing in fig. 20. It is understood that the circuit works in steps. In each step, data move along the arrows being multiplied by the respective constants and summed in $\oplus$-nodes. z^{-1}-boxes delay the signal by one step, which means that in each step they emit the number from their memory to their output, then wait until the new information has been propagated along all the arrows and sums, memorizing their input after that. Before the filter is started, memory of the boxes has to be set up properly to agree with initial conditions we want to conform to.

We can see that (352) uses four delay boxes. There exist a better way of expressing this filter, which only needs two delay boxes. It is called the *second canonical form* and is shown in fig. 22 and described by the following equation:

$$\begin{aligned} u_n &= x_n + 2\operatorname{Re}(p_0)u_{n-1} - |p_0|^2 u_{n-2} \\ y_n &= u_n - 2\operatorname{Re}(z_0)u_{n-1} + |z_0|^2 u_{n-2} \end{aligned} \tag{353}$$

Note that we only need memory boxes for u_{n-1} and u_{n-2}. Equivalence with (352) is apparent from the fact that the placement of IIR and FIR sections can be swapped because of commutativity of convolution. But this is not enough, we also need to show how to transform initial conditions when going from (353) to (352) and vice versa. When going from (353) to (352) we use the first line of (353) to compute u_{n-3} from numbers we already know.

$$u_{n-3} := \frac{x_{n-1} + 2\operatorname{Re}(p_0)u_{n-2} - u_{n-1}}{|p_0|^2} \tag{354}$$

By shifting (354) one period backwards, we also get u_{n-4}. Then we can use the second line to determine y_{n-1} and y_{n-2}, the initial conditions of (352). Note that x_{n-1} and x_{n-2} will also be needed, but we don't have to compute them as they are part of the input (being usually zero). The direction from (352) to (353) can be done by solving the equation which we used to determine y_{n-1} and y_{n-2} from u_{n-1} and u_{n-2}. I will happily leave this to the interested reader, as a boring exercise.

There is also the *first canonical form*, depicted in fig. 21, working as follows.

$$\begin{aligned} y_n &= x_n + v_{n-1} \\ v_n &= -2\operatorname{Re}(z_0)x_n + 2\operatorname{Re}(p_0)y_n + w_{n-1} \\ w_n &= |z_0|^2 x_n - |p_0|^2 y_n \end{aligned} \tag{355}$$

By substituting last equation into the middle one and the result to the first one, we obtain (352). Initial conditions can be translated this way:

$$\begin{aligned} y_{n-1} &= \frac{w_{n-1} - |z_0|^2 x_{n-1}}{-|p_0|^2} \\ y_{n-2} &= \frac{(v_{n-1} + 2\operatorname{Re}(z_0)x_{n-1} - 2\operatorname{Re}(p_0)y_{n-1}) - |z_0|^2 x_{n-2}}{-|p_0|^2} \end{aligned} \tag{356}$$

For translation in the opposite direction, the above equations can be easily inverted.

4.13.3 Note The fact that we limited ourselves to a special class of filters having as many poles as zeroes appearing in complex-conjugated pairs was not important. It only made the text shorter as we did not need to treat special cases. Generally, any real-valued filter could be described using similar decomposition, if we allowed some of the arrows in the blocks to be missing (that is zero — this would allow us to have FIR only or IIR only block, or a block with single real-valued pole or a zero). Although I only talked about one-way filters here, (355) and (353) can be used in two way case, too, just by reversing the direction of evaluation, as described in 4.5.1. ▷ 92

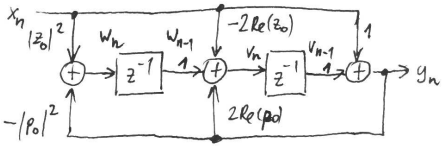


Fig. 21. *First canonical form; formula (355)*

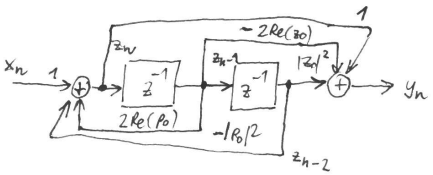


Fig. 22. *Second canonical form; formula (353)*

4.13.4 Computer Representation of Numbers

Numbers used at filter's runtime can be encoded either in fixed point or in floating point representation. Fixed point numbers are in fact integers with thought decimal (well, binary) point somewhere inside the number. Addition is identical with addition on integers, while multiplication must be done into the double precision (for example 32 bit × 32 bit → 64 bit). Note that the two operands does not necessarily need to have same number of decimal bits[52]. For 32 bit operands with m and n decimal bits we get 64 bit result with $m + n$ decimal bits. This has to be normalized down to 32 bits so that it could enter following 32 bit computations. Normalization is done first by adding 2^{m+n-1} followed by shifting the result arithmetically to the right by $m + n$ bits. Note that for $m + n \geq 32$ the result never overflows 32 bit range. The addition of 2^{m+n-1} makes the result rounded to the nearest representable number (just the shifting would round it to the nearest lower number (assuming 2's complement for negative numbers)).

Floating point numbers (or *floats*) automatize the above ideas, freeing programmers from deciding how many bits after the point are needed. Floats are represented as a mantissa and exponent pair, where the former is a fix-point number with the dot placed right after its highest non-zero bit[53], and the later specifies by how many bits the dot has to be shifted so as the mantissa would become the number the float has to represent. Contrary to fix-point numbers which use 2's complement to represent negative values, the sign bit is usually separated from mantissa in floats. Usual implementations also do not store the first bit, as it is always 1 for non-zero numbers. This is called a *hidden bit* and it improves precision by 1 bit at the cost of more complicated representation of zero (usually, certain value of exponent is reserved for it).

Upon addition, the numbers are shifted to match their exponents, mantissas are added and, should it overflow, the result is optionally shifted to the right (with simultaneous counteracting update of exponent). Multiplication is easier — it just multiplies

[52] For instance, in case of invertible minimum phase filter all the coefficients of filter blocks (355) will be less than 2. This means that we can have 30 bits after the point, where for 16 bit signals we take, say, 10 bits after the point.

[53] Therefore, the mantissa represents a number from $[\frac{1}{2}, 1)$.

mantissas (with recovered hidden bit), taking the higher bits as new mantissa. The exponents are simply added then. Note that some information about the result may be lost when the result is forced to fit into the fixed-sized mantissa before the operation is completed. This is where the errors creep into the computation. The following definition helps to quantify how quickly this happens.

4.13.5 Definition *Machine Epsilon*

For selected precision of floating point numbers we call the *machine epsilon* the largest number $\varepsilon > 0$ such that $1 + \varepsilon = 1$, where the addition is meant to be a function on floats, as described above.

Until mid 80's when majority of computer manufactures accepted IEEE-754 standard, the above ideas were implemented differently in different computers. This made programming harder as programmers were given fewer properties to rely on[54]. Specifically, it was not standardized what kind of rounding after each operation should be performed. The one I described above is called *truncation* or *rounding towards zero* as it just discards the extra bits. Unfortunately, it does not behave very well in long computations because the results are biased towards zero. By adding 2^{-b-1} (where b is the number of mantissa bits (including the hidden bit)) before the truncation we obtain *rounding towards the nearest number*. This is what we are taught in elementary school — below 0.5 to zero, above and equal to 0.5 towards one, when rounding to integers. This method is however still slightly biased because at 0.5 we always decide to go down. Although it may seem that single point (zero measure on $\mathbb{R}$ line) will not harm, it is not the case. In computer we cannot represent all the numbers of $\mathbb{R}$, so the probability of encountering 0.5 in a given computation will be non-zero, causing a bias. IEEE-754 tries to fight this by adding so called *banker's tie breaking rule*, also known as *round half to even rule*, which says that the number with two nearest integers will be rounded to the even one. Of course the FPU rounds to multiples of 2^{-b} — talking about integers just makes it easier to express — it can be formally written in C as follows:

```
double round(double x)
{
  if(x-0.5 != floor(x)) return floor(x+0.5);
  else if(((long long int)(x-0.5))&1) return x+0.5;
       else                           return x-0.5;
}
```

Although the execution time overhead of this kind of rounding is small when implemented in hardware, the `if`s make software implementation rather slow. That is why banker's tie breaking is rarely used in fix-point arithmetics on general CPUs.

Banker's tie breaking rule is based on belief that usual calculation contains roughly as many odd as even numbers, which consequently leads to zero rounding bias. If the round-off errors could be treated as random and independent, the standard deviation of the error would be $\varepsilon\sqrt{n}$ after n additions and the mean would be zero, provided that all processed numbers were of comparable magnitude. In this case, we can expect true

[54] Infamous historical example of weird FPU implementation is the Cray-1 computer with its non-commutative multiplication [12].

result to be $\pm 5\varepsilon\sqrt{n}$ around the calculated value most of the time. This is a significant improvement over worst-case error which is εn. Note that in case of truncation rounding we have maximal error of $2\varepsilon n$ since the machine epsilon of arithmetics with truncation rounding is twice as large as ε of arithmetics with rounding to the nearest. Its average error is about εn due to bias. This makes truncation rounding arithmetics less suitable for practical purposes[55]

The advantage of fix-point over the floating point is in its hardware simplicity (more adders and multipliers fit on the same area of silicon) and in the fact that fixed point addition of numbers with same number of fraction bits does not introduce noise into the result (while the floating point addition generally does). When working with 32 bit numbers, fixed points use all 32 bit for mantissa while floats offer only 25 bits for it[56]. Another great advantage is that fixed points are 'cross platform' in the sense that the results will be exactly same on all computers, which may be important in data compression algorithms, for instance. This is not true for floating point numbers, as different compilers can arrange operations differently, and different CPUs can compute slightly differently[57] even if processing equivalent sequence of instructions.

Main disadvantage of fix-points is that the program has to be carefully designed so that all the numbers involved would receive sufficient number of places after the point, while at the same time they would not overflow[58].

4.13.6 Round-off Noise

Not all permutations of basic sections in (208) are equivalent with respect to round-off errors and overflow. This is because the desired shape of the magnitude response is attained by poles modeling the passband, and zeroes modeling the stop band. As we usually want the transition between passband and stop band to be as steep as possible, we end up with poles creating sharp resonance peaks with zeroes correcting them to the target shape. Although the resulting $|H(f)|$ might be well less than 1, the intermediate results after the pole sections can be very high for some frequencies (say, about 1000). This must be accounted for in case of fixpoint implementation by allocating higher dynamic range for the output of the pole section, to prevent overflow[59]. ▷ 90

This is a problem even for floating point representation, since in the feedback we have to mix the amplified signal returning from filter's output with the low level input signal. In case of 1000-fold (60 dB) amplification at the resonant frequency, we would

[55] Despite of this and despite of the fact that IEEE-754 requires compliant hardware to have round-to-nearest/half-to-even rounding mode it is not uncommon to meet CPU that lack it. For instance the version of IBM Cell processor used in PlayStation 3 supports only truncation rounding for single precision operations. Double precision numbers are rounded correctly, but their processing is 7 times slower because the their unit is not pipelined.

[56] In IEEE-754 single precision float, there are 23 physical bits holding the mantissa, one hidden bit and one sign bit, giving 25 bits in total.

[57] For instance on IA-32 architecture, all operations in registers are performed in 80 bit precision being rounded to `float`/`double` precision only when spilled to memory.

[58] Saturation arithmetics found in the MMX instruction set conceals the problem by staying at the largest or the lowest representable value instead of overflowing. This is appreciable behavior for the output of the filter. But should this happen in the feedback, very complicated behavior could occur. Note that in this case the output of the filter can well be inside the representable range making this bug hard to be tracked down.

[59] It is understood that the input signal has a limited amplitude, usually from -32768 to 32767 in case of 16 bit samples.

need 10 more bits to those 16 we needed for the input signal, thus overrunning 24 bit mantissa of single precision numbers. The problem appears even for lower resonant amplifications because, in fact, we won't get by with just 16 bits to represent the signal. We need some bits to represent numbers between 0 and 1, so that the contributions from smaller filter's coefficients would not get lost — imagine that after the pole-section, there
141 ◁ is a zero-section (as in (353)) with $2Re(z_0) = 1/8$. As we want it to process correctly even small signals with amplitude 1, we need at least 3 more bits to represent value of 1/8. Now, imagine that the pole section has transfer of about 1 nearly everywhere except in vicinity of resonant peak, where it reaches 128. For input signal consisting of two sine waves one at the resonant frequency with amplitude 32768 and the other one far from it with amplitude 1 we would need to be able to simultaneously represent values as small as 1/8 and as large as 4194304, for which we would need mantissa with 25 bits.

To alleviate this problem we can either compute everything in double precision (which is far from being elegant, but it is good for debugging and precision testing) or we can try to group zero and pole sections into blocks such that after each section the response will be as flat as possible or — more precisely — free of peaks and trenches which would be undone the following blocks. This could be reasonably achieved by forming blocks from poles and zeroes being close to each other. This way, the numerical problem would concentrate mainly in the blocks and would not be worsened by cascaded amplification of unbalanced resonant peaks.

But we still need to address it in the block itself. From the previous paragraph we see, that the second canonical form has a problem with precision loss and resonant
140 ◁ amplification (which is an issue in integer implementation). The original formula (352) is no better, just reversed, so there would be similar precision loss due to anti-resonant attenuation. On the other hand the first canonical form uses output signal in the feedback, and at the same time the input is distributed to various places in the filter not apparently creating any zero-filtered signal with anti-resonances.

Let us make clear what precision would be needed for the entire filter composed of the blocks. Let us do it for fixed point arithmetics. First let us suppose that the filter will have $|H(f)| \leq 1$, so it can attenuate some frequencies but cannot amplify. This is plausible since we want to have same dynamic ranges on both the input and the output and don't want an overflow to occur. Recall that the amplification factor
89 ◁ ($\alpha = \frac{c_P}{d_M}$ from (207)) is applied after all the blocks have done their job. α is likely to be different from one to fulfill the condition $|H(f)| \leq 1$.

Ideally we would like to have the output of the filter to be a value obtained by rounding the value computed in an arbitrary precision arithmetics to the nearest integer. This would be highly impractical (although not completely impossible), so we have to get by with working in certain finite precision, finally rounding the result to the integer. This way, it is impossible to obtain precise agreement because, due to the feedbacks, the arbitrary precision implementation would steadily grow in the number of the decimal places used. Therefore no finite precision arithmetics could completely capture it — it is enough to be 2ε off the right value of $n + 0.5 - \varepsilon$ and we get wrong rounded value of $n + 1$ instead of correct $n \in \mathbb{Z}$, which would be obtained using an arbitrary precision numbers.

So we could only hope to be ± 1 around the true value after rounding. But still,

we can keep the probability of this error occurring as small as desired, by using long enough numbers.

In a fixed point arithmetics, the imprecisions originate solely in multiplications. Let us suppose that the round-off errors act as a noise sources, which are independent both in time and among one another, each being uniformly distributed on $[-2^{-b-1}, 2^{-b-1}]$, where b is the number of bits after the point. This is certainly not true but it is a good approximation to start with.

Let us have $h = \alpha\, b_1 * \cdots * b_n$, the impulse response of entire filter composed out of the blocks b_k. Let us also suppose that the errors from multiplications used in the first canonical form (355) will combine into the overall error signal ε_k such that the
▷ 141 numerical output of k-th block will be $b_k * x + \varepsilon_k$ instead of theoretical $b_k * x$. This gives the overall output of the filter as follows.

$$\alpha\left((b_1 * x + \varepsilon_1) * b_2 + \cdots + \varepsilon_n\right) = \alpha\left(b_1 * \cdots * b_n * x + \sum_{k=1}^{n} b_{k+1} * \cdots * b_n * \varepsilon_k\right) \quad (357)$$

instead of theoretical $h * x$, so the error is $\sum_{k=1}^{n} \alpha b_{k+1} * \cdots * b_n * \varepsilon_k$ and we need the arithmetics precise enough to keep it well below 1/2.

Anyway, the problem of round-off errors is quite complicated to be solved purely analytically (well, except until recently in [18]). For this reason it is advisable to test the filter on synthetic data before using it.

Finally, I should mention one phenomenon concerning IIR filters, called the *limit cycling*. Due to rounding it may happen that the IIR section does not eventually falls off to zero but keeps oscillating around the machine precision, creating pseudo-random noise pattern. As long as we rounded the result to the nearest integer this is of no concern as we get zero result. But rounding towards the *lower* integer is a problem as it amplifies the clutter the filter makes while oscillating around zero.

The interesting thing is that if the FPU uses truncation rounding (i.e. rounding towards zero) the limit cycle will not develop at all.

4.14 My contribution

Although this chapter contained mostly just the well known facts, it presented them without using advanced math (but still rigorously), potentially making signal processing accessible to a broader audience.

Moreover, the section 4.9 contains an original result which — to my best knowledge
▷ 112 — has not been published elsewhere. However, there is still a lot of work to be done to make this result of any practical use. Namely it has to be resolved how to treat $\mathbb{R} \to \mathbb{C}$ functions that appear there as intermediate results. As a starting point, one might consider taking dense samples of these and use method from chapter *'Computing Fourier Integrals Using the FFT'* of [65] to compute approximation of $\mathcal{F}^{-1}$. Figuring out how densely the functions should be sampled and what precision is required might not be trivial, though.

5 Traditional Front Ends

Before briefly introducing standard front ends, let us quickly mention what traits we are looking for in the waveform. According to phonology, the smallest unit of sound that distinguishes meaning is called a *phoneme*. So called *allophones* are pronunciation variants of a given phoneme which can be acoustically distinguished by human and are unrelated to particular speaker's voice. For instance, the phoneme /p/ can be pronounced aspired, as in the word *pin* or unaspired, as in the word *spin*. Frequently, this happens within so called *complementary distribution*, which means that each allophone has its own context in which it appears[1], the contexts of different allophones do not intersect and union of contexts of all the allophones form the set of all possible contexts. To give an English example [5], aspired allophone /p/ always occurs when it is a syllable onset followed by a stressed vowel. In all other cases, unaspired /p/ occurs. Sometimes, it happens that allophones of one language appear as distinct phonemes in another language, which confirms that they really exist, being more than a theoretical construction.

Similar effect, related to allophones, is called *co-articulation*. In fluent spontaneous speech, we often do not move our vocal tract to its final position, ideal for uttering given phoneme. Instead of it, we start to glide towards configuration of the next phoneme, before reaching the current one. So, only a gesture in the desired direction is made. Also, turning the nasalization and vocalization on and off may be slightly out of sync with the rest of the vocal tract. As a result, the real sound corresponding to a phoneme is influenced by preceding and following phonemes. This influence can extend even across the words, which makes the problem hard — pronunciation dictionary would not be powerful enough to capture it. Some occurrences of distinct allophones could be probably explained by co-articulation. Others are owing to the historical reasons and habits.

Another similar phenomenon is called an *assimilation*. Whereas the coarticulation is considered mainly a mechanical phenomenon beyond speaker's control, assimilation can be understood as lazy pronunciation which substitutes groups of uneasy to vocalize phonemes with similar sounds which come out with less effort, while not changing the meaning (at least in a given semantical context). For example, *'don't be silly'* can be lazily pronounced as *'dombe silly'*. The distinction against coarticulation, as I understand it, is that the assimilation can be voluntarily suppressed whereas coarticulation cannot (except by speaking very slowly). Usually we don't pay any attention to these effects but we are able to hear the difference, when listening for it. For a computer, however, this poses a problem because it 'hears' the difference every time.

Phonemes can be classified into two main categories, *vowels* and *consonants*. Vowels are the sounds pronounced with open mouth so that there is no build-up of air pressure anywhere above the glottis. Consonants lead to partial or complete constriction of the upper vocal tract at some instant of their pronunciation. As a sound, vowels can be pronounced indefinitely in time (before we run out of breath) because they lead to steady configuration of the vocal tract. On the other hand, most consonants are transient sounds, except *fricatives* (/s/, /z/ etc.) where the constriction is only partial, leading to a turbulent air flow, which manifests itself as noise and *trills* (/r/ in Spanish word *'arriba'*, for example) where the articulators return to their initial configuration after generating the transient sound so that the sound can be instantly repeated if the air flow is maintained. Although we may feel that we are able to articulate other consonants, such as /p/ and /l/, for arbitrary long time, it should not be counted because the characteristic sound appears only at the beginning, followed by indefinite prolonging sound which carries little information about the phoneme that was articulated. This contrasts with vowels, fricatives and trills, where the character of the sound does not change over the time (provided that they are pronounced in isolation). Phonemes may be voiced or unvoiced. Base frequency of voicing, so called *pitch* is often denoted as F_0. In some cases, voicing can distinguish a phoneme (/s/ from /z/, for example). Nevertheless, voicing does not convey much informations about what has been said because whispered speech is still quite comprehensible. More important role of voicing will be revealed in chapter 6. ▷ 176

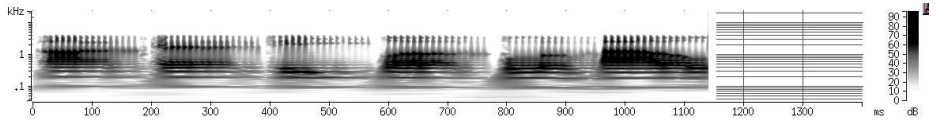


Fig. 23. *Spectrogram of Czech vowels /a/, /e/, /i/, /o/, /u/ and /au/, as generated by the EPOS speech synthesizer.*

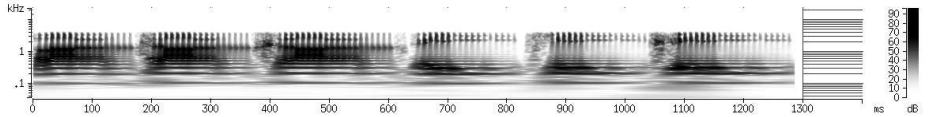


Fig. 24. *Spectrogram of Czech consonant-vowel syllables /pa/, /ta/, /ka/, /pi/, /ti/, /ki/ generated by EPOS.*

In fig. 23 we can see raw spectrogram[2] of Czech vowels. We can notice bands of higher intensity there. The lower ones that appear to come close together as we look to higher frequencies are the higher harmonics of the F_0 frequency, which corresponds to the lowest line in the spectrum. More interesting bands are located higher. These are called *formants* and usually denoted by F_1, F_2, F_3. In fact they are the resonances of the vocal tract and their resonant frequency changes as the speaker moves his articulators. Note that F_1 often lives in the area of visible F_0 harmonics and is therefore horizontally-banded, wheres the higher formants live above and get vertically banded (by the pulses with period $1/F_0$).

Ther relative position of the first three formants characterize the vowel well, as can be seen from fig. 23. Higher formant frequencies F_4, F_5, etc. exist but they do not carry additional information about the phoneme and are often too high to fit into a telephone bandwidth.

Moreover, it turns out that the vowel following or being followed by a consonant is affected by it in a specific way. As the vocal tract reconfigures from the consonant to

17 ◁ [1] Most of the time — in linguistics no 'rule' should be understood absolutely. We have seen in 2.3 that assigning zero probability to an event causes recognition error if that event occurs after all.

[2] It will be defined later, informally it can be thought of as of a visualization of signal's energy distribution over the frequency and time.

the vowel, the articulation already takes place. This causes formants to glide towards ideal vowel configuration with fast transient at the beginning. By extrapolating formant movements backward in time, down to the consonant instant, we obtain F_1, F_2, F_3, so called *locus frequencies*, which are characteristic for that consonant (see fig. 24). For most consonants, locus frequencies have little dependence on the vowel itself. On the other hand, the dependence of the vowel on the consonant might be stronger, because the vowel might not have enough time to stabilize in fast speech. Look at the following spectrogram to appreciate variety of shapes present in real continuous speech.

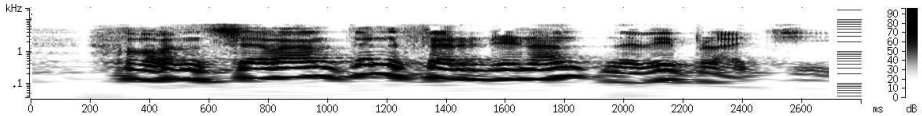


Fig. 25. *Spectrogram of the Czech sentence: "Von se vám ten Ferdinand nevyplatí…" [37].*

So, although the coarticulation causes some troubles, it allows to identify short consonants which are often too silent to be easily distinguished from the ambient background noise. Nevertheless, locus frequencies are not commonly used in today's ASR systems. Partly this is because traditional front ends lack the needed time resolution and partly this is because triphone modeling of phonemes does similar thing implicitly (to the extent permitted by resolution of the front-end).

From the above text, it may seem, that the formant frequencies are perfect candidates for features coming out of the acoustic front end. Unfortunately, they are extremely difficult to extract, because it often happens that closely neighboring formants appear as a single formant, in some voices, strong F_0 peak may be misclassified as formant, and environmental noise and echo both make formant detection even harder. The problem is that while small errors in frequency localization would be acceptable, the errors made by current formant trackers occasionally shoot-off far from correct formant position. So, instead of extracting F_1, F_2 and F_3, most front ends just work with whole vertical slices of the spectrogram in one way or another. Doing so, they avoid the problem of misclassification at the expense of introduction of irrelevant (therefore confusing) information into the recognition chain.

It should be noted that there are also other sources of information in the speech signal, running at larger time scales than phonemes do. Mostly they fall under the *prosody*, which is a collective name for pitch contours, rhythm and stress (which can be understood as sudden change in pitch, loudness and timing) and similar effects. Prosody carries semantic and emotional cues. In English, for instance, the syllable stress is usually expressed on a focused or accented words. For humans, it is unpleasant and tiring to listen to the voice having the prosody removed. So it seems reasonable that this kind of information would be helpful in speech recognizer, too. Nevertheless, in today's systems it is rarely used. This is partly due to the F_0 being removed by most front ends and partly because it is difficult to combine this information with phonemes in a useful way, because of different time scales.

5.1 MFCC

The acronym stands for *Mel-Frequency Cepstral Coefficients*. The method works as follows:

The signal is first preemphasized, that is filtered with $H(f) = 1 - \beta z^{-1}$, where $\beta = 0.95$ in case of 16 kHz sampling frequency. This roughly equalizes the power of formants. Without it, the first formant would dominate, which would be undesirable for the later processing. See fig. 26 for transfer function $H(f)$. Then, the signal is windowed using 25 ms long Hamming window being applied every 10 ms. This means that every 10 ms, a 25 ms chunk of samples is taken out of the signal into the N-dimensional vector x_n which is called a (time)-frame. Note that n indexes frames in time (incrementing by 1) and that the frames x_n and x_{n+1} overlap by more than a half of their samples. The application of the window is point-wise multiplication of x_n with the vector w. So called *Hamming window* is defined as follows.

$$w_n := 0.53836 - 0.46164 \cos\left(\frac{2\pi n}{N-1}\right) \tag{358}$$

After that, the amplitude spectrum is computed as $X_n := |F_N(x_n \odot w)|$. The purpose of windowing was to damp edges of the frame vector x_n in order to minimize discontinuity between the last and the zeroth element of the feature vector x_n. It is necessary because the Fourier transform F_N treats its input as an N-periodic signal and sudden jump at the edge would create lots of phantom frequencies which were not present in the original signal.

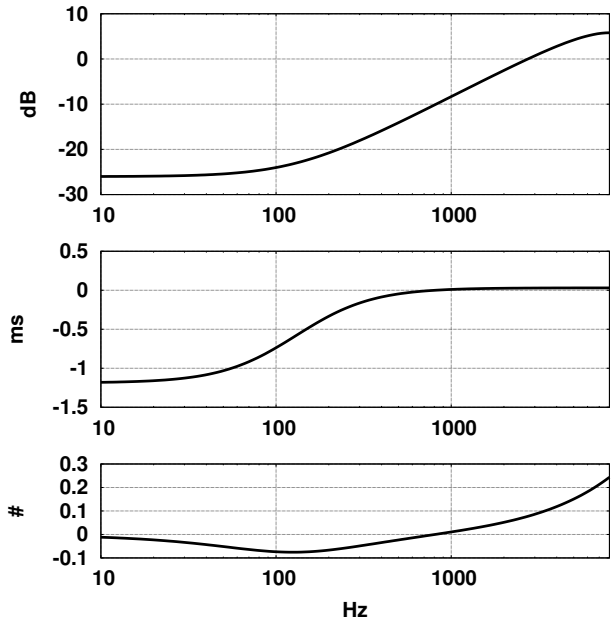


Fig. 26. *Magnitude, group delay and wave delay of typical preemphasis used in MFCC front end.*

84 ◁ It follows from 4.3.3 that windowing leads to convolution of signal's spectrum with window's spectrum in the frequency domain. Spectral components of the signal are smeared due to the window. This is called a *spectral leakage*. To reduce leakage we try to make window's spectrum concentrated around 0 and small elsewhere. There is a tradeoff between width of the main peak and stop-band attenuation. Hamming window leads to attenuation between 40 to 60 dB and 2 times worse resolution, than that of a rectangular window which is equivalent to no windowing at all.

Then, each N-dimensional vector X_n is mapped onto a 24-dimensional vector M such that $(M_n)_k := \log(\varepsilon + T_k^{\mathrm{H}} X_n)$, where vectors T_k are triangular windows, centered approximately around exponentially (in k) rising frequencies[3] (see fig. 27) made such that the adjacent triangles are half-overlapping. $\varepsilon > 0$ is a small constant which makes the formula finite even in case of zero input signal.

Finally, the discrete cosine transform[4] is taken, producing the resulting feature vector

$$m_n := C_{24} M_n \tag{360}$$

Usually, the feature vector going out of the acoustic processor also contains $\Delta_n := m_n - m_{n-1}$ and $\nabla_n := \Delta_n - \Delta_{n-1}$.

It was found experimentally, that components of M_n vectors have approximately Gaussian distribution and that the cosine transform roughly behaves as principal component decomposition, making the components of the resulting MFCC vectors m_n nearly uncorrelated, which is desirable property for the following statistical processing.

[3] This is called a *Mel-frequency scale*. In fact, below 1 kHz it is usually linear, but different authors use different definitions. There is also similar scale called a *Bark scale*. Both scales try to emphasize lower frequencies while compressing the higher ones.

[4] The cosine transform of N-dimensional vector x is defined as $C_N x$, where the matrix C_N is

$$C_N := \frac{1}{2} \begin{pmatrix} I_N & O_N & O_N & O_N \end{pmatrix} F_{4N} \begin{pmatrix} 0 & & & & & & & \\ 1 & & & & & & & \\ & 0 & & & & & & \\ & 1 & & & & & & \\ & & \ddots & & & & & \\ & & & & & 0 & & \\ & & & & & 1 & & \\ & & & & & 0 & & \\ & & & & & 1 & & \\ & & & 0 & & & & \\ & & & 1 & & & & \\ & & \iddots & & & & & \\ 0 & & & & & & & \\ 1 & & & & & & & \end{pmatrix} \tag{359}$$

Note that unlike F_N, C_N is a real valued matrix.

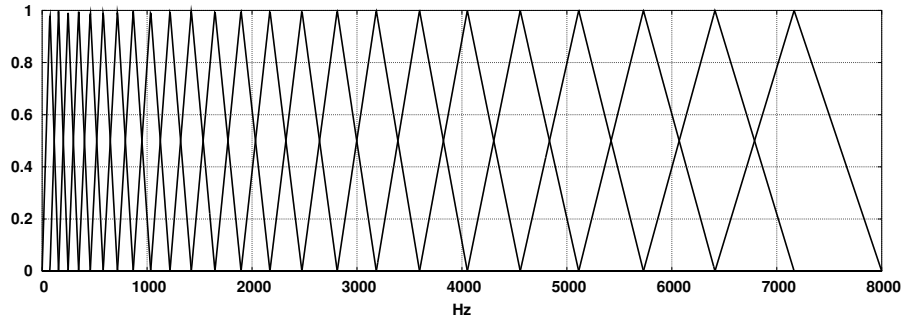


Fig. 27. *Triangular windows T_k for MFCC-HTK-FB-24, according to [57].*

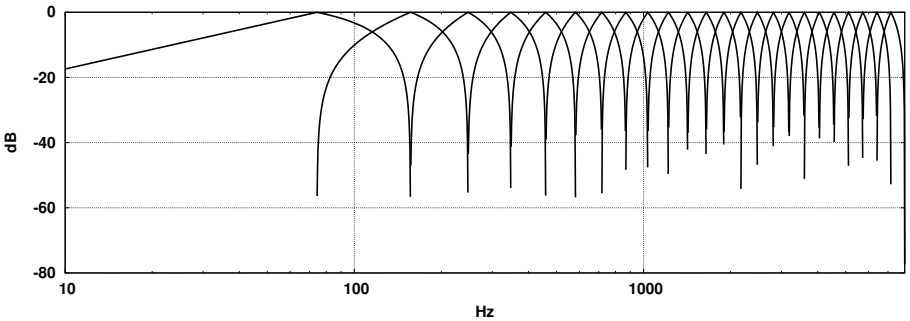


Fig. 28. *The same plot on a log-log paper. Note that in reality, the overall off-band rejection will not be better than -60 dB due to channel leakage caused by FFT window. This effect is not drawn here.*

5.1.1 Cepstral Mean Subtraction (CMS)

Almost always, $m_n - m_A$ is used instead of plain m_n, where m_A is m_n averaged over whole recording. This makes recognition less sensitive to systematical mismatch between testing and training. It can also be understood as a sort of deconvolution of short echo. Note that due to linearity of cosine transform, it is irrelevant whether we average M_n or m_n. If we forget about the triangular windowing we can informally see that

$$M_n \approx \log|F(x_n \odot w)| \approx \log|F x_n| = \log|F(y_n * r)| \tag{361}$$

where y_n represents n-th chunk taken from the ideal (undistorted) signal, while r is the impulse response of the room (and the recording channel). Note that the effect of Hamming windowing has been neglected, too. Furthermore, we assume, that r is sufficiently local, such that $y_n * r$ would be a good approximation to $(y * r)_n$ – the n-th chunk taken from convolution of r with the ideal signal y. Then

$$M_n \approx \log|F(y_n) \odot F(r)| = \log|F(y_n)| + \log|F(r)| \tag{362}$$

We can see from the formula above that the averaging estimates $\log|F(r)|$, provided that the time average of $\log|F(y_n)|$ is close to zero. Also, we can see that Δ_n needs no averaging because $\log|F(r)|$ terms cancel each other. On the other hand, it is only very rude approximation which obviously does not work for realistically long reverberation.

The most we can hope for, is that it will be able to undo the effect of frequency equalizer which might have been present in the electrical part of the channel.

The problems come from the fact that we are working with finite-length Hamming-windowed chunks using cyclic convolution (within each chunk) to model reverberation, whereas the real reverberation mixes information from different chunks. Another problem is that the logarithm is applied after we added informations from different frequency bins. Theoretically it would be more correct to first take the logarithm and do the binning after that. But it would require more evaluations of the logarithm, which is expensive and it is not sure how much it would improve recognition accuracy, after all.

Despite of these facts, the method is widely used. Often, m_A is not computed from the entire sound file, but as a moving average or even more complicatedly in which case the whole thing is called *RASTA filtering*.

5.2 LPC

LPC stands for *Linear Predictive Coefficients* and it is described in [50] in detail. It is based on belief that short (say 15 ms) segments of speech can be adequately modeled using all-pole[5] digital filter being fed by an excitation signal y which is either a white noise or impulses corresponding to glottal closures. So the output of the vocal tract would be:

$$\widehat{x}_n = y_n - \sum_{k=1}^{M} d_k \widehat{x}_{n-k} \tag{363}$$

This formula is relevant for sound generation. In case of analysis we are given the recorded signal x (let's have it preemphasized already), and we are after y and d, which would minimize

$$\sum_{k=M}^{N-1} |x - \widehat{x}_n|^2 \tag{364}$$

where M is the order of LP approximation (usually around 20), and N is the window length. Note that the segments should overlap so that we could reuse last segment's values of $\widehat{x}_0$ to $\widehat{x}_{M-1}$ as initial conditions for the recurrence (363).

Unfortunately, this optimization problem is too complex to be solved in real time. Instead of (364) we will search just for d, which minimizes the energy of $y := d * x$, where $d_0 := 1$. The y obtained this way will not be optimal in sense of (364) but will still be reasonable in many cases. It leads to the following linear system, which has to be solved in the sense of least mean squares:

$$\begin{pmatrix} x_1 & x_2 & \dots & x_M \\ x_2 & x_3 & \dots & x_{M+1} \\ \vdots & & & \vdots \\ x_{N-M} & x_{N-M+1} & \dots & x_{N-1} \end{pmatrix} \cdot \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_M \end{pmatrix} \approx - \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-M-1} \end{pmatrix} \tag{365}$$

[5] That is a filter with $c = \vec{1}$.

In principle, this can be solved by Moore-Penrose pseudo inversion using singular value decomposition of the matrix. But much faster methods exist, see [65] for reference.

Once we have the coefficients we can either use them directly, or convert them into poles (which is impractical as it leads to root-searching of a polynomial) or we can compute the magnitude response of the obtained filter, from which it is possible to derive cepstral coefficients in the very same way we did in case of the MFCC. This is the most common choice and the resulting coefficients are called LPCC, then. Again, direct methods exist (see [50]) so there is no need to actually compute the magnitude response, to get the results.

Moreover, as the residual signal y resembles expected excitation signal it can be used to determine F_0 of voiced sounds.

5.3 PLP

Even clearly recorded speech signal still contains a lot of information which is irrelevant to its phonetic contents. This irrelevant information makes statistical training more difficult because, in principle, we need to observe all combinations of relevant and irrelevant information, to properly estimate the probabilities involved in the acoustic model. A good front end should therefore try to minimize influence of this excessive information.

PLP (*Perceptual Linear Prediction*) tries to do that, based on an assumption that the sounds which cannot be heard by a human listener can hardly be used to decode the phonetic message. Possible 'explanation' could be that our language evolved such that we mainly produce sounds we are able to hear and don't waste our energy with those we cannot. Moreover, even if we produced sounds that we cannot hear, we would not be able to learn how to control them independently of those we can hear, due to the lack of feedback. So these should carry only a duplicate or unrelated information.

Whether this assumption is true or not, it is by all means practical, because common sound compression formats (such as `ogg` or `mp3`) already exploit limits of human hearing, by encoding the waveform with fewer quantization levels, such that the quantization noise stays below current threshold of hearing. Hence, having a front end which can ignore this additional noise is advantageous.

PLP algorithm goes in several steps (see [30]) as follows:

(1) Chop the input signal into overlapping blocks x_n, same way as we did for MFCC.
(2) Compute spectrum $X_n := |F(x_n \odot w)|$, where w is the Hamming window.
(3) Warp the amplitude spectrum, getting Y_n as $(Y_n)_{\lfloor \alpha \log(1+f) \rfloor} :\approx (X_n)_f$, for $f \in \mathbb{N}$, where $\alpha \in \mathbb{R}^+$ is a suitable constant controlling how many frequency bins there will be in Y_n. Note that this is only a conceptual formula. To be fully specified we would have to say how to map possibly multiple values f on a single value of $\lfloor \alpha \log(1+f) \rfloor$. One way to do that could be via triangular windows we have met in MFCC. Also, instead of plain logarithm, Mel or Bark frequency scale could be used.
(4) $Z_n := Y_n * S$. This tries to approximate frequency masking, by smoothing the amplitude spectrum Y_n with a critical band masking curve S (masking will be explained in 5.8).

▷ 161

(5) $Q_n := Z_n \odot E$. This performs loudness equalization. Human ear's sensitivity and loudness perception changes with frequency whereas computer's stays nearly constant (up to about ±3 dB or better). This formula tries to simulate imperfections of our ear by attenuating frequencies we hear badly. Vector E can be obtained experimentally by setting amplitudes A_g of pure sine tones at frequencies corresponding to the bins numbered by g such that human subject will perceive all the tones to be equally loud. This can be accomplished by pairwise comparison followed by modification of the amplitudes. Then, $E_g := 1/A_g$. Obviously this should be repeated with many persons and averaged.

(6) $(H_n)_g := (H_n)_{(2L-g)} := ((Q_n)_g)^{0.6}$, where L is a dimension of vector Q_n and $(H_n)_0 := 0$. This step tries to estimate perceived loudness according to (slightly controversial [6]) Stevens' power law.

(7) Restore periodic signal, which would represent the time frame n, as $h_n := F^{-1}(H_n)$ and perform LPC analysis on it. In [30], Heřmanský recommends LPC order $M = 5$ for 8 kHz sampling frequency. Note that this can properly capture only the first two formants. These LPC or LPCC coefficients form final output of the PLP front end.

5.4 VTLN extension

VTLN stands for *Vocal Tract Length Normalization*. As the name suggests it tries to compensate for frequency effects of different vocal tract lengths. If the vocal tract was physically magnified α-times in all directions, we would obtain α-times lower resonant frequencies because the sound would have to travel α-times longer distances before it would be reflected off the inner surface of the tract. Let us approximate any stiff configuration of the vocal tract with an LTI system having a transfer function $H(f)$. Then, the α-times larger copy of it would be described by a transfer function $R(f) = H(\alpha f)$ for $|f| \leq \frac{1}{2}\min(1, \alpha^{-1})f_s$. For $|f| > \frac{1}{2}\min(1, \alpha^{-1})f_s$ the shape of R is not specified by H because this information was filtered out from H when the sound of the impulse response was sampled into the computer. Likewise, for $\alpha < 1$ we lose higher frequencies of H because these cannot be represented using the given sampling period. Let us suppose that our sampling frequency f_s is so high that all meaningful vocal tracts will have negligible transfer at frequencies which may be lost or added, so we would not need to worry about it.

VTLN technique can be embedded into most front-ends. Its place is just after the computation of the frame's spectrum ($\widehat{X}_n := |F(x_n \odot w)|$ in case of MFCC). It compresses/expands the frequency axis as $(X_n^\alpha)_f := (\widehat{X}_n)_{\lfloor \alpha f \rfloor}$. By properly chosen speaker dependent parameter α the method makes the front-end's output vectors A_n^α invariant to vocal tract magnification, thus enhancing recognition accuracy in speaker independent systems. This is because the acoustic model does not need to capture voice differences. Instead it can use VTL-normalized data to train a more robust model. Note, however, that various speaking styles still persist in these data.

The parameter α has to be selected during recognition, such that it would normalize the input sound. ASR system reestimates α on each utterance, thereby achieving continual adaptation to the user's voice. First, the utterance is recognized using simple non-VTLN acoustic model λ and simple language model l, giving us the first guess $\widehat{W}$

$$\widehat{W} := \arg\max_W \Pr{}_\lambda(A \mid W) \Pr{}_l(W) \tag{366}$$

Then, we can use VTLN acoustic model Λ and VTLN feature vectors A^α to estimate α in the following way:

$$\alpha := \arg\max_\alpha \Pr{}_\Lambda(A^\alpha \mid \widehat{W}) \tag{367}$$

That is, we have chosen α such that the acoustic vectors generated from the approximative transcription would be maximally probable. Finally, the utterance is recognized using VTLN features, previously determined α and full language model L.

$$W := \arg\max_W \Pr{}_\Lambda(A^\alpha \mid W) \Pr{}_L(W) \tag{368}$$

In principle, the last two steps could be iterated but the first iteration is usually good enough already.

Note that if we plotted the spectrograms on a logarithmic frequency axis then $\log \alpha$ would be a horizontal shift of the plot. By its proper selection VTLN moves spectra into their 'normal' position, which simplifies their recognition. In principle, this can be done directly in the time domain by resampling the signal to different sampling frequency because this is exactly what the shift in the log-frequency domain does. Hence there is no need for FFT in the front end to use VTLN. By time domain resampling the method could theoretically be used even with LPC front-end.

VTLN, as described so far, is based on an assumption that the only difference among speakers is in their size, meaning that the scale-normalized shape of their vocal tracts should always be same. This is obviously false. Let us take phoneme /s/, for example. It originates as an aerodynamic noise created by tongue-constricted air-flow. Center frequency of the noise band is determined by dimensions of a resonance cavity delimited by tongue tip, teeth and lips. As we can control its volume by tongue's position to large extent, phoneme /s/ does not change much among speakers[6]. Also, other cavities we use for articulation may scale at different rate than vocal tract length. To compensate for this, VTLN with multiple warping parameters, each corresponding to its preferred phoneme class, can be used. For two parameters α_1 and α_2 this works such that the frame feature vector is a pair $(A_n^{\alpha_1}, A_n^{\alpha_2})$, whose components are computed from single acoustic frame using α_1 and α_2. Emission probabilities of HMM specify only $A_n^{\alpha_1}$-part or only $A_n^{\alpha_2}$-part, depending on phoneme class the respective transition is part of. During decoding, the Viterbi algorithm only uses that half of the feature vector, which is specified in the transition.

Note that this description of VTLN was only introductory and ignores many practical problems, such as the training. See [56] for implementation details and experimental results.

[6] This can be heard by playing a recording of someone's voice using lower sampling frequency than at which it was recorded. Although vowels sound as if they were pronounced by a physically bigger speaker, fricatives sound unnaturally dull, namely /s/ sounds more like /š/.

5.5 Pitch Synchronous Processing

If more pitch periods of voiced speech signal fit into the 25 ms analysis window, they manifest in the spectral domain as closely spaced peaks (so called *harmonic frequencies*) that sample the spectral envelope (see fig. 29) which corresponds to the amplitude spectrum of the phoneme whispered. This happens because the voicing can be understood such that the vocal folds generate impulses $s = \sum_{k=-N}^{N} \vec{1}[kT]$ which are then filtered thru the vocal tract with impulse response $h = \mathcal{F}^{-1}(H)$. The FFT in the front end estimates $|\mathcal{F}(h * s)|$ which is

$$|\mathcal{F}(h*s)|(f) = \left| H \odot \sum_{k=-N}^{N} \mathcal{F}(\vec{1}[kT]) \right| (f) = \left| H(f) \sum_{k=-N}^{N} e^{2\pi i f k T} \right| \tag{369}$$

78 ◁ leading to (see 4.2.7):

$$|\mathcal{F}(h*s)|(f) = |H(f) D_N(fT)| = |H(f)| \cdot \left| \frac{\sin(2\pi f T(N+1))}{\sin(\pi f T)} \right| \tag{370}$$

where $D_N(fT)$ is $1/T$-periodic in f and it has high peak at 0, while it is close to zero elsewhere (except, of course, at $f = k/T$ due to periodicity). It therefore destroys information about $H(f)$ between the peaks of $D_N(fT)$ because the resulting small amplitudes will be hidden below environmental noise.

This is of no concern for men's voices because the triangular windows in MFCC are wider than $F_0 = 1/T$. Hence the peaks will be averaged out. But in case of high pitched women's voices, it can happen that the $1/T$ becomes larger than MFCC triangular window span, causing spurious peaks in triangular-windowed spectrogram
150 ◁ M_n (see 5.1). Moreover, high F_0 means that the peaks of $D_N(fT)$ become so widely spaced that they can miss true maxima of the formants. This is a real problem (even for LPC features) because it creates false positions of formants.

Pitch synchronous processing tries to avoid it by analyzing voiced sounds in variable window, whose length is set to the current pitch period. Hamming window is not needed here, because the wrap-around discontinuity is small (and can be further suppressed by subtracting trend-line from the samples in the block). As a result, the spectrogram has slightly higher frequency resolution.

This method works well on a clean signal. However, in case of real-world noisy sounds, pitch period estimation is far from being reliable. Even small errors can undermine principles upon which the method stands.

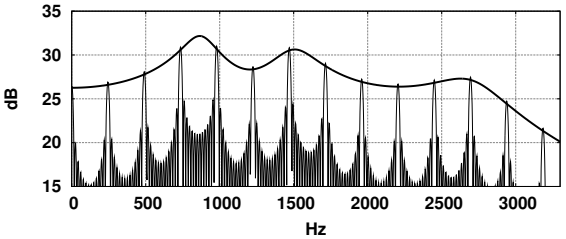


Fig. 29. *Simulated vowel-like amplitude spectrum for $F_0 = 245$ Hz and 25 ms segment. The upper spectral envelope $H(f)$ corresponds to the impulse response of the vocal tract, while the peaks which we actually measure are caused by voicing. In reality they would decline from the envelope since the glottal pulses have more complex shape than $\vec{1}[kT]$ used in this simulation. Ignoring this fact when computing the spectral envelope, we in fact recover $(H \cdot P)(f)$, where P is speaker dependent spectrum of single glottal pulse. However, CMS in the later stage of the front end is likely to minimize its impact.*

5.6 PMVDR – BISN

PMVDR-BISN (*Perceptual Minimum Variance Distortion-less Response – Built-In Speaker Normalization*) tries to address the same problem as pitch synchronous processing, while avoiding its reliance on exact F_0 estimate.

It is rather involved method and, to my best knowledge, it is currently the most powerful one, working well even in noisy conditions. I will only sketch the algorithm here. For explanation and details, see [61] and [68]. Informally, MVDR is a power spectrum estimation method which can be used to estimate the upper spectral envelope of power spectrum affected by harmonic peaks, as the one in fig. 29. MVDR starts by LPC analysis. It then employs magical formula which uses these LPC coefficients and the variance of prediction error to obtain all-pole model for the smoothed spectrum. If there were L harmonic peaks in the spectrum and $M = 2L - 1$ then MVDR using M-th order LPC would be just able to fit all the peaks with no more freedom left. Therefore it would not be able to fit valleys between the peaks, which is why the output will be good estimate of the upper spectral envelope.

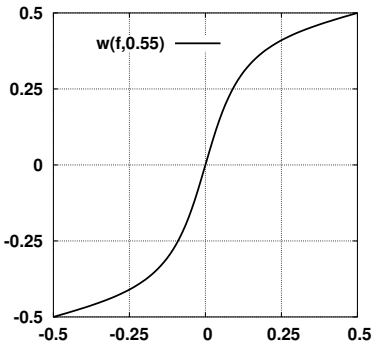


Fig. 30. *Approximated Bark-scale.*

PMVDR front end has two parameters, LPC order M and perceptual warping parameter α (something like α we used in VTLN, but here, the warping will be non-linear). It works the following way:

(1) Preemphasize the input signal and chop it into overlapping segments x_n.
(2) Apply Hamming window and compute power spectrum: $X := |F(x_n \odot w)|^2$.
(3) Warp the spectrum according to 'perceptual deformation'

$$w(f,\alpha) := \frac{1}{2\pi} \operatorname{atg2}\big((1-\alpha^2)\sin(2\pi f),\, (1+\alpha^2)\cos(2\pi f) - 2\alpha\big) \tag{371}$$

obtaining perceptual power spectrum Y in the following way

$$Y_{\lfloor w(f/N,\alpha)\%N \rfloor} :\approx X_f \quad \text{for } f \in [0, N-1] \tag{372}$$

where N is dimension of X and Y vectors. The $:\approx$-mapping is done by simple linear interpolation of neighboring values. For $\alpha = 0$ we get no distortion at all, for $\alpha = 0.42$ get an approximation of Mel scale, while for $\alpha = 0.55$ we get an approximation of the Bark scale (fig. 30). These values of α were for sampling frequency of 16 kHz.

(4) Compute 'perceptual autocorrelation' signal $z := F^{-1}Y$.
(5) Compute M-th order LPC fit to the N-periodic signal z, obtaining coefficients $d_1 \dots d_M$. Note that the prediction error signal is $r = d * z$, if we put $d_0 := 1$.
(6) Use magical MVDR formula to obtain upper spectral envelope $H(f)$ as follows:

$$H(f) := \frac{1}{\sum_{k=-M}^{M} \mu_k e^{-2\pi i f k}} \tag{373}$$

where μ_k can be obtained from d and prediction error variance $V := \mathrm{Var}(r)$ in the following way [61]:

$$\mu_k := \begin{cases} \overline{\mu(-k)} & \text{for } k \in \{-M, \ldots, -1\} \\ \frac{1}{V} \sum_{i=0}^{M-k} (M+1-k-2i) d_i \overline{d_{i+k}} & \text{for } k \in \{0, 1 \ldots M\} \end{cases} \tag{374}$$

(7) $a := \mathcal{F}^{-1}(\log(H(f)))$ is a theoretical cepstrum corresponding to $H(f)$. It can be approximatively computed from μ_k using FFT. By taking first 12 elements of a (excluding the 0-th one) we form the resulting PMVDR cepstral coefficients m_n.
(8) Δ_n and ∇_n temporal differences can be computed from m_n, if required.
(9) Usually, vectors m_n are also subject to cepstral mean subtraction.

BISN extension is in fact VTLN adapted for PMVDR. It continually adjusts α, trying to achieve best possible performance. Unlike VTLN, it does not work in two steps (search for optimal α, followed by recognition phase using this α). Instead, it continually refines α in a feedback, optimizing it against high quality transcription produced by the recognizer using feature vectors generated with the old value of α. Although we get optimal α for already processed utterance, it does not matter much because α changes slowly most of the time. Moreover, the update formula has additional inertia term, so that the α could not shoot-off just because of single misrecognized utterance.

All the optimization is build into the front end, making it independent from the rest of the recognizer. The only thing that is required is time-aligned phoneme (or state) transcription fed back from the recognizer's output into the front end. There must also be separately trained acoustic model of $\Pr(A^\alpha \mid V)$ inside the front end, which is used for α optimization. Using golden section search it is possible to find best α in less than 10 evaluations of $\Pr(A^\alpha \mid V)$. See [68] for details.

5.7 Experimental Front End: TRAPS

All the front ends described so far share one common property. They work locally, looking only at a short piece of sound at a time. In real-world environment, however, reverberation or even simple non-constant group delay longer than 10 ms will cause information leakage among nearby frames returned from such a front end. This is a problem because it introduces extra variability to the acoustic stream that is not related to the spoken words. As a result, HMM must be trained under various environmental conditions, to obtain reasonable robustness. This not only makes HMM bigger but also leads to more demanding decoding because the transition probabilities get equalized. Hence more paths in the trellis seems to be promising and the decoder has more work to explore them all.

On the other hand, psychoacoustic experiments suggest that human auditory system uses context (or window) of at least 200 ms, according to [29][7]. Presumably, this integration time has something to do with echo compensation because room reverberation must be about 300 to 500 ms long before one is even aware of it and must reach several seconds before it affects comprehensibility [14].

[7] One can 'feel' this effect by listening to 0.2 to 0.3 s long excerpts from his favorite tunes, trying to guess which one it was. When I tried this, I was not able to identify the tune correctly, finding the sound strange and sometimes even novel, hardly similar to anything present in the original music.

Another annoying thing with the above front ends is that they treat the spectral vector as a unit. So, if for instance half of it is missing (being attenuated or masked by noise) the feature vector will be wildly different from what the decoder expects. It will then probably completely fail, while human listener finds the sound still comprehensible.

It is suggested in [14], that according to experiments performed by Fletcher in 1920' for AT&T, human auditory system seems to work in about 20 independent frequency bands, running 'local phoneme recognizer' in each of them, fusing their results afterwards. Namely, Fletcher discovered that subjects performing phoneme recognition[8] task on band-pass filtered speech, make errors with probability

$$e_B = \prod_{k \in B} e_{\{k\}} \tag{375}$$

where we suppose that the frequency axis has been divided into non-overlapping bands, indexed by k, and that e_X is a probability of the subject making a phoneme recognition error when listening to the signal where only bands listed in the set X remained after filtering. In this notation $e_{\{k\}} = e_X$, where $X = \{k\}$. If we interpret e_B — the probability that the subject fails on sound filtered by $B = \{b_1, \ldots, b_n\}$ — as

$$\begin{aligned} e_B = \Pr(&\text{subject fails on sound filtered by } \{b_1\} \,\&\\ &\text{he also fails on the same sound filtered by } B \setminus \{b_1\}) \end{aligned} \tag{376}$$

we see from 5.8.9 that the events *'subject fails on sound filtered by* $\{b_p\}$*'* are independent. This suggests that the fusion of partial recognizers' results will be a non-trivial process since it essentially finds the correct recognizer even in when all but one are wrong. Since its discovery, formula 5.8.9 has been verified many times, using different bandwidths, noise conditions and different subjects. ▷ 174

These observations led to the development of TRAPS (*TempoRAl PatternS*) — an experimental system working in time domain in several independent frequency channels, trying to detect phonetic cues directly in each channel.

There are 15 channels, each emitting its signal's average energy every 10 ms. TRAPS then look at these energy tracks using 1 second long analysis window. Each TRAP is in fact a neural network (multi-layer perceptron) being trained to assign one of 29 phonetic classes to its output. This class corresponds to a phoneme being pronounced in the middle of that 1 second window. For technical reasons, the signal in the window is first mean-removed, variance-normalized and then Hamming-windowed and only after that fed into the network. So we have 15 networks each emitting possibly different answer and we have to select the best one. For this task, we train another multi-layer perceptron which takes all these TRAP outputs fusing them into final 29 phonetic classes. These are then used as an acoustic evidence by conventional HMM model.

This front end was meant mainly as a proof-of-the-concept and although it was simple, not containing explicit echo compensation and detection of noisy or unreliable bands, it could measure well with traditional approaches. See [31] for implementation details.

[8] To neutralize influence of human 'language model', Fletcher used CVC (consonant-vowel-consonant) triples drawn independently and uniformly from the set of all such triples, as his testing data.

5.8 Natural Front End: Cochlea and Auditory Pathways

I will skip anatomical description of the outer and middle ear as it can be found in standard physiological textbooks, only noting that the outer ear behaves like a linear filter (which is however direction dependent owing to the pinna) and that the inner ear's lever mechanism is essentially an impedance transformer, transforming air-induced vibrations of the eardrum from high-amplitude/low-pressure mode into a low-amplitude/high-pressure mode so that the sound could enter the liquid environment in the cochlea without being reflected back too much. The lever mechanism also contains muscles (*stapedius* and *tensor tympani*) which control damping — when these are engaged, the transformer attenuates energy. They are controlled reflexively to protect cochlea from loud sounds[9] and they are also used to attenuate speaker's own voice. The muscles are well perfused and accordingly 'overdesigned' so that their tremor would be minimized. Otherwise, it could be heard as a noise. It has been suggested in [26] that the lever mechanism introduces slight nonlinear distortion to the transfered sound.

5.8.1 Inner Ear

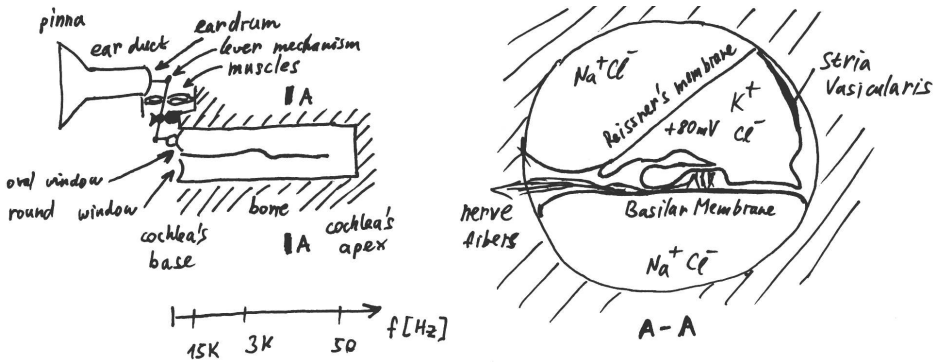


Fig. 31. *Functional schema of our hearing system. On the left, there is an outer ear, an inner ear and unwound cochlea with a wave traveling along its basilar membrane. It takes it 4 ms to travel from cochlea's base to its apex. Below, there is a scale showing where on the membrane the traveling wave of a particular frequency will reach its maximal amplitude. The cross-section A-A on the right shows the structure of the membrane. The Reissner's membrane can be neglected from the wave-mechanical point of view. Its main purpose is to separate the V-shaped duct from the rest of the cochlea so that it could maintain different ionic potential, which is essential for proper function of the hair cells located in the organ of Corti, the actual mechanical/electrical transducer, located roughly along the centerline of basilar membrane. The needed ions are generated by cells of stria vasicularis wall.*

The inner ear – or the cochlea – is mechanical/electrical transducer combined with frequency analyzer, which transforms the input sound into neural firing patterns that

[9] The attenuation is about 20 to 30 dB and is only effective for low frequencies up to 2 kHz. For higher frequencies and for very low frequencies (which are conducted by bone) the effectiveness is lower. The reflex is activated by sounds louder than 70 dB-SPL or when a vocalization starts. It has a delay of 30 to 180 ms, louder sounds leading to shorter delays. Therefore, it does not protect against sudden loud sounds, unless one screams upon anticipating them, thereby activating the reflex in advance by the vocalization mechanism. It seems that the brain compensates for this attenuation because perceived loudness of the sound remains unaffected.

enter the brain. It is a helically shaped duct filled with a water-like liquid separated by a V-shaped membranes along its axis. Basilar membrane supports the actual transducer – the organ of Corti. If we abstract from helical shape, which is irrelevant[10] to the function, we can imagine the cochlea as a rigid[11] tube longitudinally divided by a membrane, as depicted in fig. 31. The sound enters the upper compartment from the left, propagating in the liquid to the right, coupling with elastic basilar membrane, creating a traveling wave on it. Stiffness of the membrane decreases as we move towards the cochlea's apex, which causes characteristic shape of these waves: as the wave travels, its amplitude rises as the stiffness decreases until it reaches maximum. Then it vanishes quickly. Position of the maximum depends on the wave's frequency (see fig. 32), being approximatively logarithmic[12]. Because of approximative linearity of the process, more complex sounds are decomposed into their spectral frequencies this way. Therefore, the successive places along the basilar membrane can be regarded as outputs of a filter bank processing the input sound. This is what Békésy stated in his theory of traveling waves for which he received Nobel price for physiology in 1961. He demonstrated it on magnified mechanical models and, to some extent, on dissected animal cochleas [64].

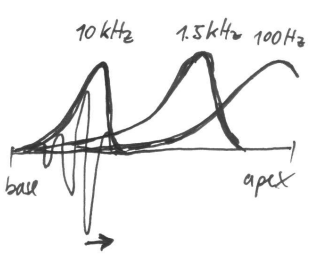


Fig. 32. *Envelopes (thick curves) of Békésy's waves (thin) of various frequencies, with actual traveling wave drawn for $f = 10$ kHz.*

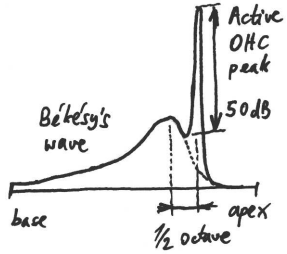


Fig. 33. *Envelope of actively tuned wave rising from Békésy-wave envelope. See [1].*

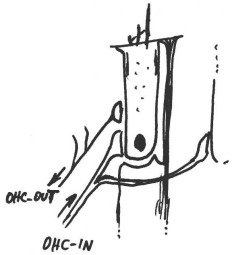


Fig. 34. *Outer hair cell (OHC)*

To fully appreciate his contribution, consider that the displacement of basilar membrane is about 10 to 20 nm even for loud sounds, so it could be hardly observed directly using methods of his time (displacement of the oval window is 200 nm for a 120 dB sound). Instead, he was observing wave-induced transport of micro-particles he deposited onto the basilar membrane[13].

[10] Birds have it straight, for instance.

[11] Because the real cochlea is buried in the bone.

[12] But this is not a physical necessity. For example, mustached bat's cochlea is highly non-uniform in this respect. About 25% of it – so called SI-zone – is dedicated for analysis of frequencies between 61 and 72 kHz (which is less than 0.24 octave) and about half of the cochlea is tuned exactly to 61 kHz (so called CF2-zone). In the CF2-zone, the waves are not attenuated at all. Due to this property and due to reflection from the apex, a standing wave can build up in this area. If the frequency is slightly out of tune, the standing wave will be slowly sliding sideways. Using this mechanism, the bat can detect very subtle Doppler shifts induced by flapping wing of its insect prey [32]. However, it has to continually adjust the frequency of its sonar calls so that the Doppler-shifted echo would hit the CF2-zone as a 61 kHz signal. The SI-zone can be used for this task as a source of feedback information. Note that the filters involved are very sharp, commonly with slopes of 4000 dB/oct and higher, whereas humans have less than 300 dB/oct.

[13] This way, he also demonstrated that waves can only propagate from the base to the apex and not

Before Békésy, theory of Helmholtz was widely accepted. It states that the basilar membrane behaves like a collection of chords each of them being strung between the walls of the cochlear duct (that is perpendicularly to the wave travel). Their stiffness decreases towards the apex so that each audible frequency has its own 'tuned' chord. The pressure wave propagates in the liquid and thru the couplings of adjacent chords (the membrane is contiguous so we have to model it as mechanically coupled chords) and gradually builds up oscillations of those chords whose resonant frequency is present in the sound. Vibrating chords represent the sound, then.

Békésy's and Helmholtz's theories are compatible in a sense that with appropriate selection of stiffness and damping of the membrane, behavior of each of them is possible. Békésy measured membrane's stiffness on real cochleas and according to physics he concluded that there are traveling waves in the human cochlea, which he later confirmed by a particle transport experiment.

Interestingly, actual measurements on neural fibers indicated much steeper filters than those predicted by Békésy's theory. Békésy himself was well aware of rather flat filters in his model and proposed that probably lateral inhibition between neighboring neural fibers makes frequency response steeper. However, this 'second filter' has never been found in the actual neural tissue. It was not until 1983 that it was discovered by measurements on living cochleas that the tuning was as good on the membrane as at the neural output. Moreover, in 1978, it was discovered that the inner ear produces sound after it receives a click sound. The extra sound is quieter and delayed by several milliseconds but still strong enough that it can be measured by a microphone inserted into the ear duct. This phenomenon was named *oto-acoustic emissions* and is nowadays used as a diagnostic method of deafness in newborns. Finally it was found that so called *outer hair cells* are responsible for these emissions and that they in fact vibrate the basilar membrane[14], fine tuning it as shown in fig. 33. We can see that the maximum frequency is about half octave below the maximum of Békésy's wave and that the peak is 50 dB above the Békésy's. So the outer cells also work as amplifiers. When damaged, the threshold of hearing shifts 50 dB upwards and frequency discrimination deteriorates since only flat Békésy's waves remain. Outer hair cells are unique to mammals making their audition superior to that of other species in frequency range and in resolution. For instance birds, which presumably rely on passive waves[15], only hear up to 11 kHz (most of them only up to 5 kHz), whereas bats hear up to 100 kHz and whales up to 200 kHz. Nevertheless, the birds have roughly similar threshold of hearing as mammals. This suggests that the lack of amplification by OHCs can somehow be overcome passively (by different geometry, say) or by a different, yet unknown active mechanism.

Remarkably, already in 1948, there was a theory by T. Gold, who proposed that there should be an active amplification in the inner ear. He essentially took Helmholtz's theory and observed that damping caused by liquid environment would make it difficult, if not impossible, for the resonances to build up. For example, a piano would not play very well immersed in water because of friction between the strings and the water.

in the opposite direction (this is caused by decreasing stiffness of the membrane and does not hold for the CF2-zone of bat's cochlea where waves travel in both directions, creating a standing wave).

[14] The emitted waves then travel towards the base mainly thru the liquid since basilar membrane rapidly attenuates backwardly traveling waves.

[15] At least they lack *prestin* motor protein, so if there is any active mechanism it has to be different.

However, we could make it play again by mounting a sensor, amplifier and an actuator to its every string to counteract that friction. Gold thought that the cochlea is similar to this piano [58]. He even predicted oto-acoustic emissions (as he knew how easy it is to turn amplifier into an oscillator), but failed to detect them using microphones of his time and so the theory was abandoned.

Today's theory of hearing assumes that Békésy wave initiates OHCs activity which then counteract the friction so that a resonant vibrations could build up as in the Helmholtz's theory. It might take some time after the onset of sound before membrane's amplitude gets 50 dB above the Békésy wave. This could explain why we perceive short sounds as quieter — they simply do not have a time to fully resonate.

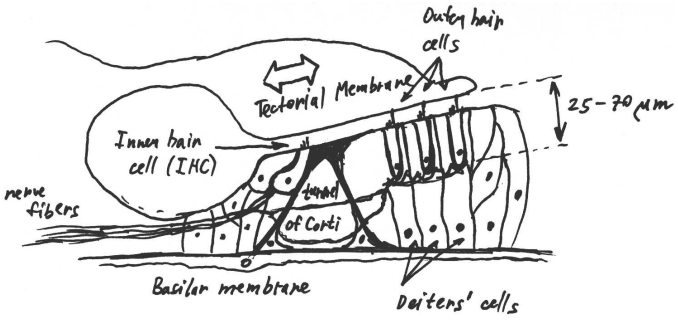


Fig. 35. *Organ of Corti. The longest hairs of the outer hair cells (OHC) are fixed into the tectorial membrane, while the hairs of IHCs are not. Relative shearing motion of the tectorial membrane is bending hairs, causing depolarization of the respective hair cells. On depolarization, IHCs just send spikes over the attached neural fibers, while OHCs also change their length (vibrating synchronously with the sound) by the action of motor protein prestin. These vibrations are passively transfered into the basilar membrane by Deiters' cells, creating highly tuned local spot.*

Fig. 36. *Inner hair cell (IHC) depicted with neural fibers. There are 10 output fibers per cell, on average.*

Let us take a closer look at the *organ of Corti*, shown in fig. 35. We can notice 3 rows of outer hair cells and one row of inner hair cells. Short segments of 4th or even 5th row of OHCs are common in human cochlea. There are about 3500 inner cells and about 13000 to 18000 outer cells. Once destroyed (by a loud noise or due to poisoning), they cannot be replaced as they cannot reproduce. Outer hair cells have part of their membrane coated with motor protein *prestin*, which can rapidly change cell's length keeping up with the sound (that is up to 20 kHz in human). Watch [41] video for clarity. OHC vibrations are then passively transfered into the basilar membrane by Deiters' cells, creating peak of maximum oscillations. This in turn generates more intense relative shearing motion of the *tectorial membrane*, which is then detected by *inner hair cells*, the actual sensor. The length of outer hair cells is 'tuned' to their working frequency, so that the highest frequency cells (which are closest to cochlea's base) are short (25 μm in human) and firmly attached to the tectorial membrane by their longest hairs, while the lowest frequency ones are tall (70 μm in human), and only loosely attached to the membrane. Taller cells also have longer hairs, which causes the gap between the tectorial membrane and hair-cell plane to widen towards lower frequencies. The active mechanism is stronger at higher frequencies and weaker at the lower ones, where Békésy waves become important. At the very low frequencies the

outer hair cells are even used as detectors, while everywhere else, these only amplify and tune the sound, leaving the transduction work solely to inner hair cells.

5.8.2 Auditory Nerve

There are four kinds of fibers in the auditory nerve (fig. 37). There are 30000 *IHC-out* myelinated fibers which form main output from the cochlea. Each inner hair cell connects to 10 IHC-out fibers on average and each IHC-out fiber goes to only one cell. Then there are about 3000 *IHC-in* fibers which synapse with IHC-out fibers changing their threshold (see fig. 36). Then there are 8000 *OHC-in* fibers branching to 40000 fibers in the cochlea. These connect to outer hair cells controlling their threshold. Each fiber of OHC-in channel controls about 20 adjacent OHCs. Finally there are 1500 *OHC-out* non-myelinated fibers which form an output from the outer hair cells. Their function is unknown to me. They might be part of a feedback loop controlling thresholds of OHCs. As low frequency OHCs also receive the sound, maybe some of these fibers encode it, too (but these OHCs could as well be connected to IHC-out channel[16]). Each OHC-out fiber collects data from many outer hair cells. IHC-out and OHC-out fibers have neurons located in so called *spiral ganglion*. But these are bipolar neurons,[17] which only provide the fibers with chemical energy, not altering the information passed thru them. They are therefore unimportant from our point of view.

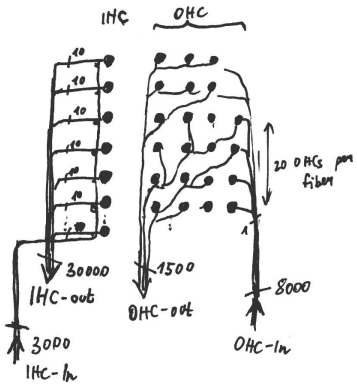


Fig. 37. *Cochlea's neural interface*

Sending analog signals over nerve fibers is challenging because any single fiber is not able to transmit more than about a thousand spikes per second and cannot express amplitude directly as each spike is voltage-normalized (as a digital signal). To overcome this, each inner hair cell is connected to about 10 IHC-out fibers of different activation thresholds. IHC-out fibers tend to fire at particular phase of the stimulating mechanical vibration. So the interspike intervals tend to occur at integer multiples of the period of the tone. This phenomenon is called a *phase locking*. At the lowest amplitude, only the fibers with the low threshold respond. They however do not fire at every period of a sound wave. As the intensity increases, the fiber is more likely to fire within a phase-locked time window. With further increase of amplitude it eventually becomes saturated, firing at almost every

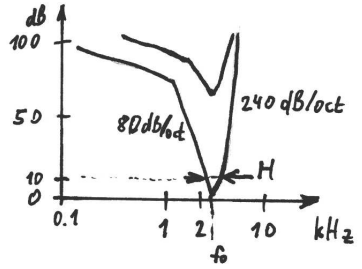


Fig. 38. *Typical threshold of high (lower) and low (upper) spontaneous-rate fibers for various frequencies. So called quality factor $Q_{10} = f_0/H$ is about 5 to 10 in human and 100 to 400 in SI-zone of mustached bat's cochlea.*

[16] This would be consistent with the finding that during prenatal development, OHCs are connected both to OHC-in and IHC-out channels, according to [1]. IHC-out synapses are later withdrawn but perhaps the useful ones could survive.

[17] Bipolar neurons have one input and one output axon and no dendrites.

opportunity. Before reaching this limit, another set of fibers with higher threshold becomes operational, so that the brain still receives meaningful information about intensity. The fibers with low threshold are called *high-spontaneous-rate fibers* because they spontaneously transmit random spikes due to thermal noise, even in silence[18]. *Low-spontaneous-rate fibers* are those less numerous but still important fibers which encode high amplitudes.

Phase locking works reliably only up to 2 or 3 kHz. Above it, firing becomes more and more random but its probability still remains a nondecreasing function of the sound's amplitude. In fig. 38 there is a plot of intensity of a sound wave of particular frequency which is needed to excite the fiber above its spontaneous level. On frequencies for which the phase locking works, its base period corresponds to that of the driving frequency. The plot closely corresponds with shape of the active wave (fig. 33) — ▷ 162 by measuring mechanical amplitude at a fixed place on the basilar membrane, while sweeping the frequency of the sound, we would get a similar plot (only upside-down reversed). Imagining this, we can see that the flat part of the curve is caused by a Békésy-tail of low frequency waves which are just transiting thru our measuring spot. With a raising amplitude, steepness of the curve deteriorates as can be seen from the upper curve which applies to low-spontaneous fibers. This is because the active mechanism becomes saturated and cannot keep up with higher amplitudes of the basilar membrane, making Békésy-waves more apparent. As a result, our hearing is less precise at very high intensities.

Using OHC-in fibers, brain can control blocks of adjacent outer hair cells, thereby setting amplitude of the active peak and, as we have seen in fig. 38, also its steepness. Its center frequency will probably be slightly affected, too. It is known that a narrow-band noise introduced into one ear affects activity of same-frequency OHC-in fibers in the other ear as well.

Finally, it should be noted that basilar membrane moves non-linearly. This non-linearity is stronger than the one introduced by lever mechanism in the middle-ear. It is most likely caused by an activity of OHCs because the membrane's motions become linear when these cells are destroyed. It is supposed that as OHCs become saturated they push and pull basilar membrane with less force then would be required for linear operation. This not only gives rise to a flattened filter's shape at high amplitudes (fig. 38), but also makes closely located peaks to influence one another in a complex way. This leads to more aggressive masking than the necessary one dictated by an uncertainty principle. It can even generate tones which did not exist in the original signal. For instance, so called *cubic tone* of frequency $2f_2 - f_1$ becomes audible when f_1 and f_2 are close enough (such that their active peaks touch). This tone really exists as a membrane vibration, which can be proved by measuring it as an oto-acoustic emission. Besides these adverse effects, non-linearity simplifies intensity encoding because the vibration of basilar membrane becomes amplitude-compressed, thus requiring less range.

[18] Their threshold is controlled by IHC-in fibers to some extent. When these are cut, spontaneous activity decreases together with attainable dynamic range. This might be a mechanism responsible for the typical reaction of IHC-out fiber to an onset of constantly loud sound — first, the firing rate is high but after 10–20 ms of constant fall it settles at about half of the original rate. This is called *adaptation* and probably works by lowering activity of IHC-in after the brain detects the sound onset. Note however, that this is only my speculation and there might by other explanations, mainly that the adaptation is innate to the OHC-out synapse itself and does not require IHC-out–brain–IHC-in-loop.

5.8.3 Auditory Pathways

Little is known about actual audio-processing in human brain, especially when compared to what is currently known about its video-processing. Here follows what I have been able to reconstruct from the literature (see fig. 39).

The cochlea (C) sends its output to the CN unit which is said to decode the duration and intensity of the sound. Specifically in PV-CN subunit there are so called *octopus cells* which detect onsets of sound on individual fibers. These neurons are very fast, able to respond to click trains at a rate of 800 Hz. In PV-CN there are also *chopper cells* which, when excited, tend to fire periodically within certain narrow range of frequencies. Their probability of firing is driven by collective activity of the set of neighboring fibers. When the channel they represent is amplitude modulated by a frequency that the chopper cell is tuned to, the cell tends to phase-lock on it, firing at instants of maximum spike density (i.e. at envelope peaks of the underlining sound). Otherwise the cell fires less frequently and more randomly. By having several of these cells connected into a coincidence detector (which is suspected to be located in IC), the brain can detect amplitude modulation of certain modulating frequency range for a given cochlear band. The circuit is repeated for many combinations of cochlear bands and modulating frequencies, thus making a 2D map of modulation intensity given carrier frequency and modulation frequency. See [2] for details. Note that amplitude modulation by frequency F_0 naturally arises in voiced phonemes. Moreover, with sophisticated mutual inhibition of chopper cells tuned to different modulating frequencies but belonging to a single cochlear channel, it might be possible to track multiple modulations simultaneously (even in single channel). This could be used for rudimentary source separation and segmentation.

The AV-CN subunit contains so called *bushy cells* acting as coincidence detectors of signals in several neighboring fibers (possibly even spanning few neighboring IHCs). That means that they fire only when representative portion of their inputs receive spike at the same time. By this mechanism, they remove noise from high spontaneous rate fibers. It is obvious that this can only work for phase-locked low frequency sounds. AV-CN sends its phased-locked low-frequency output to M-SO

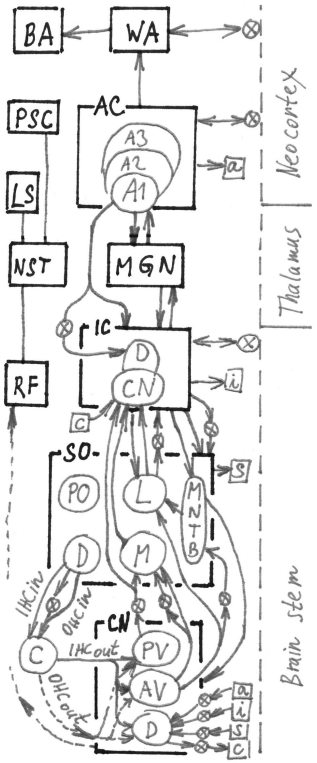


Fig. 39. *Simplified block diagram of auditory pathways. Dashed lines indicate non-myelinated fibers, arrows mean data flow direction. Due to left-to-right symmetry only the left side is drawn. The →⊗→-symbol denotes fibers crossing to the other side. Key: C = Cochlea, CN = Cochlear Nucleus, SO = Superior Olive, IC = Inferior Colliculus, MGN = Medial Geniculate Nucleus, AC = Auditory Cortex, WA = Wernicke's Area, BA = Broca's Area, RF = Reticular Formation, NST = Non-Specific Thalamus, PSC = Poly-Sensory Cortex, LS = Limbic System, M = Medial, L = Lateral, D = Dorsal, AV = Anterior Ventral, PV = Posterior Ventral, PO = Periolivary Nuclei, MNTB = Medial Nucleus of Trapezoid Body, CN = Central Nucleus of.*

on both sides of brain, whereas the high frequency channels are sent to both L-SOs. In addition to that, the paths to the contralateral L-SO go thru the MNTB unit.

The D-CN subunit is different. It resembles cerebellum by its architecture and wiring[19], getting some of the inputs — in fact a feedback — from higher levels of auditory pathways, including neocortex. Its neurons react very sensitively to notches in spectral envelope at frequencies above 8 kHz. It is believed that these cells in fact help to decode elevation of the sound source. The reflections off the complex shape of pinna result in different time-lags for the direct path and the reflected paths. As such, pinna acts like a direction dependent filter introducing zeroes in higher frequencies where the direct path sound and the reflected one tend to cancel each other. Elevation can be estimated, knowing these zeroes and azimuth information. This is happens in the IC unit. Whole CN unit contains about 8800 neurons.

Otherwise, each subunit of CN is organized such that the ordering of fiber's center frequencies remains same as in the auditory nerve, which in turn follows frequency ordering of IHCs in the cochlea. This ordering is preserved all the way up to the auditory cortex.

The output from CN goes to SO and IC units at the opposite side of brain, with the exception of fibers leaving AV-CN subunit which sends two channels of fibers, one going to the opposite side, the other staying at its original side. The channel which stays, divides into low-frequency fibers which enter M-SO and high-frequency ones which enter L-SO.

In M-SO, there are 15 500 neurons acting as delay lines and time-of-arrival comparators, comparing ear-to-ear phase of the sound, trying to laterally localize its source. For sounds located maximally sideways the delay is about 700 μs, but humans can detect delays as short as 10 μs. M-SO relies on phase locking which is why only low frequency fibers (up to 1.5 kHz) are used for it. The L-SO subunit also tries to perform lateral localization. It uses high frequency fibers, comparing intensities from left and right ear. At frequencies above 6 kHz, the head-cast acoustic shadow can make the difference as large as 20 dB. Recall that the elevation information is not available in SO, since the positions of the zeroes are being sent from D-CN directly to CN-IC.

In IC, the spatial localization of source is probably finished as all required information is available there (note that left and right ICs are directly connected). It is also known, that IC is involved in reflex which turns our head towards a source of sudden loud sound. It is hypothesized that IC is also involved in pitch detection. IC is composed of 392 000 neurons.

From the IC, the signal goes to MGN (composed of 570 000 neurons), which is a part of thalamus. It is composed of number of subunits which are not drawn in the diagram and their purpose is not fully understood yet.

In general, MGN relays the information to the *auditory cortex* (AC) which consists of concentrically shaped units A1, A2 and A3. Little is known about their function except that A1 is involved in detection of temporal patterns and that it is important for recognizing a rising pitch from the falling one. A2 is said to recognize learned patterns, assigning learned significance to them. It is known that for the right-handed, the right AC concerns more with melody and fine frequency localization, whereas the

[19] Consult [23] for early theory of cerebellar cortex.

left one does sequencing and phonetic processing. The ACs are connected via the corpus callosum. Together they are made of 10^8 neurons.

Wernicke's area (WA) is located close to the AC and it is involved in semantical decoding of spoken message. The left one does phonetic and semantic processing while the right one decodes prosody and emotional tone of speech. It is also active when we are singing or listening to a song. Left and right WAs are connected via corpus callosum channel. Subjects with damaged WA do not understand meaning of spoken language albeit they can fluently speak grammatically correct nonsensical sentences. Those who recovered, reported they found speech of others completely unintelligible and although they were aware that they are speaking, they could neither stop it nor understand their own words. Less severe damages to WA influence ability to find right names for semantical concepts. Sufferers often recall a word which sounds similarly and fits grammatically into the sentence (for instance, they can confuse *television* with *telephone*). This suggests that WA might be involved in word-to-meaning mapping.

Broca's area (BA) is responsible for speech production. It gets its input directly from WA as well as from many other parts of neocortex. In the right-handed, only the left BA is considered to be important for the language. It has variety of functions concerning language, namely assembling a sentence according to grammar rules before it is vocalized. Also, it is using grammatical information to decode the meaning of more complex sentences (such as those in passive voice). It is hypothesized that it also plays some role in decoding speech-related arm gestures. When damaged, patients know what they want to say but cannot get it out. They are able to understand simple sentences, but cannot speak fluently and grammatically. They also have a difficulty in word finding, articulation and comprehension of grammatically complex sentences both verbally and in writing.

When the link between BA and WA is damaged, patient loses ability to exactly repeat what was said to him. Other functions are nearly unaffected, except for the tendency to swapping or dropping phonemes or syllables, probably owing to lack of the fast feedback. The subject is able to hear the errors, often hopelessly trying to correct them.

As WA is closer to AC than BA, it is tempting to assume that during recognition, instead of ASR-like language model, semantics is used first, followed by syntactic disambiguation performed by BA. This is also supported by the fact that a damage to BA does not impact practical use of the language as much as a damage to WA. Nevertheless, the truth will probably be much more complicated.

This was a description of primary auditory pathway. There is also a secondary pathway, which goes from CN to *reticular formation* (RF) via slow non-myelinated fibers, continuing to *non-specific thalamus* (NST) ending in *poly-sensory cortex* (PSC) and *limbic system* (LS). In RF the signals are mixed from multiple sensors. This pathway affects attention by assigning priority to auditory, visual, tactile and other sensory information.

5.8.4 How it Might Work

It is clear from the number of feedback loops in fig. 39 that the hearing is an active process. This is somewhat similar to vision where we need feedback loops which keep our eyes aimed and focused on target. Only in case of sound it is not quite clear what

else than volume should be controlled. Anything written in *slanted font* in the rest of this section are my speculations.

It is likely that some of the loops are present only to compensate for parasitic effects caused by ear's construction — for instance it is plausible to assume that simultaneously with contraction of muscles in the middle ear, the activity of low frequency OHC-in and IHC-in fibers increases, jointly compensating for reduced transfer function of the middle ear. This would explain why we do not perceive attenuating effects of these muscles. Using this mechanism, the brain could also compensate for uneven equalization caused by reverberant environment. This would be somewhat similar to cepstral mean subtraction used in computer front ends.

Spiking rate of each fiber could be regarded as an output of a band-pass filter. Its frequency response is less steep towards lower frequencies (fig. 38) because the low frequency waves must travel across the high frequency regions of the basilar membrane causing some response there too. ▷ 165

This may be a problem in environment with very loud low frequency sound, which might overpower quiet high frequency sounds even after filtering. *To compensate for it, brain could turn the muscles on, which would reduce an amplitude of the disturbing low frequency wave that travels all the way along the cochlea. As the attenuation by the middle ear muscles is more effective at lower than higher frequencies, it would improve SNR.*

As a signal in OHC-in gets weaker, the OHCs are less likely to depolarize and so the active peak gets smaller. At the same time its slopes get flatter, widening the filter's band. The center frequency may also change a little. Filter with wider band can be made with shorter impulse response. So, by decreasing activity of OHC-in fibers while at the same time increasing that of IHC-in (which is to compensate for the loss of amplification) the brain could tradeoff time and frequency resolution according to signal it listens to. Note however that IHC-caused bandwidth changes cannot be too dramatic owing to the loss of amplification. For this reason I think it is not very likely that brain really uses something like this.

5.8.5 Binaural Beats

When we are listening to two sine waves of very close frequencies, we actually hear beats in amplitude of a single tone instead of two distinct steady tones. This is not surprising because $\cos(2\pi ft) + \cos(2\pi gt) = 2\cos(\pi(f+g)t)\cos(\pi(f-g)t)$. Here, the $\cos(\pi(f-g)t)$ represents the beating envelope of the tone. Note the the perceived frequency is $|f-g|$ not its half-value as the formula suggests. This is because the we cannot distinguish when the cosine is negative or positive. So the envelope we actually perceive is $|\cos(\pi(f-g)t)|$ which repeats with frequency $|f-g|$.

The beating occurs even if the two sines are of different amplitudes. Recalling 4.11 it is easy to derive, that ▷ 124

$$A\cos(2\pi ft)+B\cos(2\pi gt) \approx \cos(2\pi\max(f,g)t)\sqrt{A^2+B^2+2AB\cos(2\pi(f-g)t)} \quad (377)$$

where the envelope (which is the square root part) is exact (in fact it is the Hilbert envelope of the signal) whereas the real carrier wave is slightly phase modulated version of the presented one. The discrepancy becomes negligible for $A > 2B$ and so I stayed with easier formula. Fig. 40 gives an example of possible envelope shape. ▷ 171

What is interesting about this, is that we hear beats even if the left ear listens only to f, while — at the same time — the right one listens only to g. It should be noted that binaural beats only occur at low frequencies (up to about 1 kHz). *Therefore M-SO is probably responsible for the effect.* The effect is not perfect though, as the binaural beats are not perceived as sharp as monaural, at least by me. I arranged an experiment with $f = 500$ Hz, $g = 507$ Hz and the binaural beats with $A = B = 1$ sounded more like monaural with $A = 1$ and $B = 0.32$ to me.

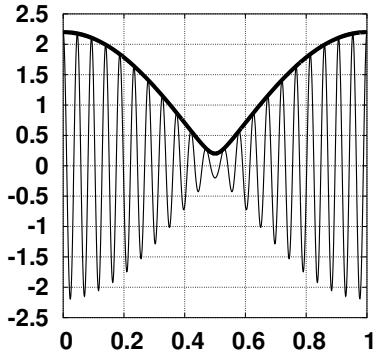


Fig. 40. *A beat for $A = 1$, $B = 1.2$, $f = 21$, $g = 22$ with the envelope (bold line).*

Nevertheless, this means that the spiked representations of the sound in the auditory nerves can be combined — at least to some extent — as if it was a real sound, namely that the spike density can be 'subtracted' so that the waves could (destructively) interfere with each other. In my opinion, this is used to increase quality of the sound from a desired direction by 'adding' the left and properly delayed right channel. The delay lines from M-SO could be used for this. However, for proper function, brain should probably also compensate for non-linearity of cochlea so that the addition would simulate interference precisely. An elegant way of achieving this would be using an IHC-in signal (and perhaps even OHC-in to some extent) to set left-right balance so that the loudness (as encoded on acoustic nerves) would be equal for the sound source being tracked. Imprecision in setting these levels or inability to subtract waves in spiked representation well enough might explain why we perceive beating of signals with $A = B$ as if $A \neq B$.

5.8.6 Haas Effect

The *Haas effect* or the *law of the first wavefront* is a binaural effect revealing how we locate sound source in enclosed reverberant environment, such as a cave or room. The effect occurs when the same sound arrives after 2 to 100 ms after the wavefront. Then the direction of the sound source is evaluated by the IC unit only from the first arriving sound. This way, it reduces the possibility that the sound location would get confused by the clutter of late reflections. The gating time is sound and room dependent, being up to 50 ms for speech and 100 ms for music. Moreover, the later waves are still used for hearing as they often provide better SNR than the direct one. Hass found that the effect takes place even if the late waves are 10 dB above the wavefront. In this case, however, the effect is reduced to delays of only 10 to 30 ms. For longer lags, two sounds are heard.

The Hass effect is exploited in movie-theater sound systems. The surround sound is delayed by 30 ms so that the spectators could determine the position of the sound source from wave generated by the front speakers while the surround speakers mounted on the walls provide the ambient feel and improve SNR.

5.8.7 Masking

Let's imagine for a while the cochlea as a bank of fixed linear filters. Then, its output fibers would ideally separate sine components of the sound into different channels. We

have already seen that two close pure tones lead to beating. This happens if they fall into a single channel. Because of uncertainty principle the frequency selectivity of each cochlear channel must be limited if we want to preserve time features as well. In fact, the filters (whose magnitude response looks as upside-down reverted plot of fig. 38) ▷ 165 are very closely spaced (there are 3500 of them), overlapping considerably. That is why the beating appears even if the respective tones hit different fibers, whose filters overlap sufficiently. The tone belonging to a different fiber is attenuated, so it does not influence the result much. But when its amplitude is high it might be comparable to the tone best fitting to that particular fiber and the beating occurs.

When the foreign frequency gets even louder, the beating envelope gets shallower (as in fig. 40). As the auditory nerve conveys amplitude information logarithmically encoded it may easily happen that these shallow variations in amplitude fall below the fiber's noise and become indistinguishable from signal caused by the louder tone alone. We say that the quieter tone has been *masked* by a the louder one, often called a *masker*.

So the masking manifests itself as local elevation of the threshold of hearing in vicinity of strong peaks in short-time frequency spectrum. It is more profound in case of narrow-band noises than pure tones because the noises do not lead to beating, which the brain uses to discover partially masked quiet tones.

The real cochlea performs more aggressive masking than the one just described. My explanation is that *the basilar membrane is non-linear and it seems that the OHCs in vicinity of a loud peak are likely to help with its amplification, thus being ineffective for weaker sounds on frequency at which they would normally respond. As a result, the amplification is reduced at the peak flanks leading to masking there.*

At the same time, the brain tries to circumvent masking as much as it can. For instance, if the masker arrives from different direction than the target signal, that is each ear receives the masker in different time relatively to the target, the brain selects clear parts from each ear, thus reducing the masking effect.

Similar effect exists in time domain, too. It is called *temporal* or *non-simultaneous masking* and it causes elevation of the hearing threshold of a given cochlear channel for about 30 ms before the sound's onset and about 100 – 200 ms after its end. *It seems that this mechanism tries to gate the clearest parts of the sound so that their subsequent processing would not get confused by reverb. It is similar to the Haas effect except that we want the loudest (best SNR) wavelet here, not necessarily the wavefront.*

It is likely that many levels of auditory pathways are involved in the temporal masking. Nevertheless, its rudimentary form is present already at the cochlea's output. Recall[18] that the OHC-out fibers fire rapidly at the onset of the sound, then adapting to ▷ 166 a lower rate. Also, after the sound ends, there is a recovery period of elevated threshold (the fibers even lose their spontaneous activity for a while). *Brain does not interpret this as a volume change, because it was him who sent the command over IHC-in fibers to do so. From his point of view it was only a change in a dynamic range, so that the anticipated reverberations would fall below the noise level, therefore being masked*[20].

[20] This conjecture predicts that the adaptation time and level will differ according to reverberant characteristics of the environment, which are believed to be estimated by the brain from, say, last 10 seconds of (binaural) sound. Namely, in non-reverberant environment the adaptation should be weaker than in highly reverberant environment. By not observing this phenomenon, the conjecture

Higher levels of processing could make the masking sharper, depending on context.

The masking is employed in sound compression formats such as the `mp3` which does not encode the information we cannot hear. Note however that some information we do not *hear* can still influence our judgment on source location owing to Haas effect. For this reason the `mp3` is considered inappropriate for binaural audio.

5.8.8 Off-Frequency Listening

Off-frequency listening is another tool our brain uses to reduce frequency masking. It improves signal to noise ratio (SNR) by listening from different fiber than the most active one (*it is probably also used to bridge damaged regions of cochlea*). From fig. 41 it is clear that if we listened using a filter centered at frequency S, we would get SNR1, whereas with off-frequency listening at $S + \Delta f$ we'd obtain better SNR2. Note that this is limited by thermal noise and depends on the fact that there are no other signals in the passband of the filter we listened with.

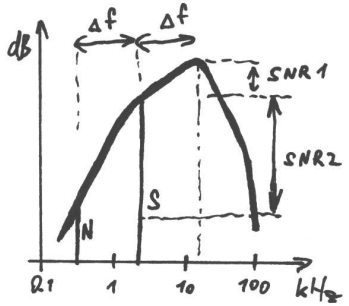


Fig. 41. *Simple off-frequency listening to a sine signal S, being disturbed by equally loud noise N.*

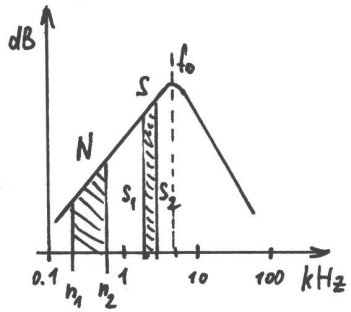


Fig. 42. *More realistic model of signals under idealized triangular filter.*

Concave shape of the filter's magnitude response in fig. 41 was important for the method to work. If the shape was linear (on a log-log paper) there would be no improvement at all. This is clear for pure sine signals. But it also holds for band-limited signals depicted in fig. 42. For sake simplicity, let us assume that N and S signals have rectangular power spectrum of unit height. The power of N filtered by the left flank of idealized triangular filter is

$$P(n_1, n_2) = \int_{n_1}^{n_2} 10^{\frac{1}{10}\left(\alpha 10 \log_{10}(f/f0) + \widehat{\beta}\right)} df = \beta \int_{n_1}^{n_2} \left(\frac{f}{f_0}\right)^{\alpha} df = \frac{\beta\left(n_2^{\alpha+1} - n_1^{\alpha+1}\right)}{(\alpha+1) f_0^{\alpha}} \quad (378)$$

where $\alpha > 0$ and $\beta = 10^{\widehat{\beta}/10}$ control the shape of left flank of the filter and f_0 selects its center frequency. When considering S to be of the same type as N, we get SNR as follows.

$$SNR = \frac{P(s_1, s_2)}{P(n_1, n_2)} = \frac{s_2^{\alpha+1} - s_1^{\alpha+1}}{n_2^{\alpha+1} - n_1^{\alpha+1}} \quad (379)$$

could be falsified. Note that IHC-in fibers could either directly quench the activity of OHC-out fibers at each onset of sound, or they could only somehow influence intrinsic adaptation constant of OHC-out synapses, in case the adaptation would be built into the synapse itself.

In another words, it does not depend on f_0 at all, hence it does not pay-off to listen from off-frequency fiber in this case. This also justifies importance of the curved shape of fig. 41. Fortunately it is easier to achieve this with digital filters than to approximate perfectly straight filter of fig. 42.

5.8.9 Recognition Performance of Humans

In Fletcher's experiments (one of which was already mentioned in 5.7) it was discovered that the probability of correct recognition of uniformly randomly selected CVC (consonant-vowel-consonant) syllable is c^2v, where c is a probability of correct recognition of a consonant, while v is a probability of correct recognition of a vowel [14]. At the same time, the probability of correct identification of random CV or VC syllable is cv. This formula holds regardless of the speaker and quality of the speech signal. Of course, the actual values of c and v change according to conditions, but the syllable error rate also changes such that the formula holds. ▷ 159

As correct recognition of a syllable means correct recognition of all its phonemes, this formula says that the probability of phoneme error is independent of recognition errors in its neighborhood. This is interesting because the phoneme realizations depend on one another due to coarticulation. Fletcher's result is interpreted such that our brain somehow resolves coarticulation in early stages of auditory chain (before employing language or phonetic model). This should give us subjective sensation of phonemes as being a basic units of speech.

When vowel and consonant errors are both counted, we obtain *phone error rate*. Humans can achieve phone error rate of about 1.5 % for nonsensical CVC syllables, under best recording conditions. This contrasts with today's most advanced phoneme recognizers which can barely achieve 20 % error rate [49].

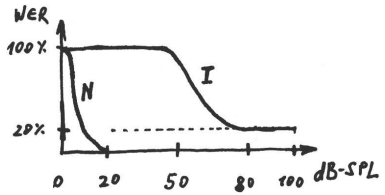


Fig. 43. *Word Error Rate (WER) of normal (N) and impaired (I) persons according to [1]. The impaired ones had their inner hair cells damaged, thereby relying on Békésy waves only.*

In [1] there is a summary of a remarkable study, comparing subjects with destroyed outer hair cells with normal people in the task of word recognition under various intensities of the sound. Subjects had to repeat single words (which were picked at random) which were played to them at different volumes. The experiment resulted in a plot shown in fig. 43. The loss of 50 dB amplification performed by OHCs is clearly apparent. Although the elevated threshold of hearing could be compensated by a hearing-aid, these people never reach perfect recognition as the normal ones, topping around 20 % WER, regardless of the volume. This is because their spectral resolution is limited as they perform spectral analysis by Békésy-waves only.

Coincidentally, todays best ASR systems have WER around 20 % in realistic speaker independent conditions such as in MALACH project. It should be stressed that this is pure coincidence, firstly because the tasks on which the error rates were obtained are very different (MALACH being a continuous speech recognition, while the impaired people were tested on single words without any context) and secondly because in ASR there are other problems that hurt performance, such as the out of vocabulary words, processing of strangely accented or non-grammatical speech none of which appeared in the human test.

165 ◁ It is nevertheless interesting to note that the Q_{10} factor (defined in fig. 38) which is almost 5 to 10 (increasing with frequency of the fiber) in healthy human, is worse than 2 to 6.5 in MFCC-HTK-FB24 frontend and about 3 to 1 (decreasing with frequency) for cochlea with destroyed outer hair cells.

5.8.10 Human Performance in Isolated Tasks

It was found that the human auditory system suppresses echoes of up to 30 ms (this corresponds to 10 m of path difference). These are not perceived, whereas the longer ones are heard as reverb.

As for frequency discrimination, it takes 15–20 ms of steady tone before we could fully determine its frequency. We can then discriminate two consecutive tones being 2 or 3 Hz apart as long as their frequencies lie below 3 kHz (*probably we are exploiting synchronized spiking of OHC-in fibers for that*). This precision is nearly independent of the frequency below 3 kHz, deteriorating considerably above.

5.8.11 Disclaimer

This physiological detour should not be treated as being very precise or up to date. In fact, some claims might be quite incorrect. It is based on what I remembered from lectures I attended, on literature [36, 26, 1, 32, 64, 6, 14, 2, 4, 3] and consultations with our leading neurophysiologists.

As there are contradictory claims in the literature and different authors sometimes use slightly different names for same things I tried to resolve these conflicts using common sense to get a unified picture of the subject. By this process, however, I might have introduced some errors. Especially any numbers (counts of neural fibers, slopes of the filters, etc.) should be treated with reasonable suspiciousness.

From the same reason, the plots are drawn by hand so that they would not evoke false feeling of precision. They are meant as an illustration of principle, only. I have no means how to verify them and the fact that the measurements vary greatly[21] from work to work forced me to either average them (if the difference was mild) or to chose the more plausible one (if they were far apart).

As this section was meant as an inspiration only, I hope it would not matter after all. Main contribution of this section should be seen in gathering available information at one place, concentrating on function instead of anatomy and presenting the matter in technical terms.

5.8.12 Acknowledgments

I would like to thank Prof. MUDr. Josef Syka, DrSc., dr.h.c. and Dr. Ing. Daniel Šuta from Institute of Experimental Medicine[22] for consultations which really improved quality of this section. Should there still remain any major errors, these were caused by me, not listening to their advices as carefully as I wished to.

[21] Partly, this is probably because some works still consider Békésy's waves only, hence getting much flatter filters.

[22] `http://www.iem.cas.cz/institute/`

6 NUFIBA Front End

In this chapter, I will describe the front end used in my recognizer. In comparison with traditional front ends it will be more complicated and will require more calculations when being run but still it will be an artificial construction. I don't think it has a sense to simulate biological hearing system other than for the purpose of designing better hearing aids or brain research. As we have seen in the previous chapter, the cochlea has its own construction problems[1] and many of its complicated features are only counteracting them in some way. It would be wasting of computational power to simulate for example the way how the amplitude gets encoded in auditory nerve fibers when the same information can be stored in single 32 bit number.

Nevertheless, I took into account overall properties of cochlea and early auditory pathways, when designing the front end.

6.1 Why Front End?

In this section I will lay down the reasons why I think it worths to invest an effort to development of better front end. I will also summarize properties I think it should have, according to the lessons learned from the previous chapters, using physiological data and performance of former ASR implementations as a guide of the new design.

(1) Although the error rate of contemporary recognizers is already quite acceptable, it rises rapidly when echo of the room changes to the one which was not in the training data or in a presence of noise, especially non-uniform and non-stationary. Current front ends do little about it, leaving most work to the acoustic model. Consequently this must be trained with many combination of possible voices, noises and echoes to work reliably, which is difficult to achieve. On the other hand, humans don't need to learn the language from scratch when they enter a new environment. Only short adaptation is required. This suggest that we don't need to update our 'speech models' at all. Instead, we only have to figure out how to translate distorted sounds into something we understand.

(2) Fletcher's experiments (5.7 and 5.8.9) suggest that humans outperform today's machines in phoneme recognition by an order of magnitude. Also it seems that brain can resolve coarticulation before assembling phonemes into words, that is before using any context [14]. Having such a front end would greatly simplified the acoustic model. ▷ 159

(3) Experiments [22] conducted with STRAIGHT vocoder suggest, that we perceive physical size of the sound source directly and don't need to perform VTLN (as described in 5.4) by adaptively changing the warping factor (which represents the size), based on previous utterance. Instead, it seems that somehow, our auditory system decodes both the sound type (size-normalized) and its size at once. Even ▷ 155

[1] Some receptors may malfunction or even die, yet the system as a whole must remain reasonably functional. When programming a computer, on the other hand, we may assume that the hardware is perfect because the redundancy can be solved at a different design level.

very short sounds (like single phonemes) are still comprehensible even if resynthesized with vocal tract lengths no human could physically have.

174 ◁ (4) Performance of subjects with damaged outer hair cells shows (see 5.8.9) that even our own 'mighty language model' cannot correct errors caused by reduced frequency resolution. In other words, inadequate front end cannot be salvaged by a language model of whatever quality.

(5) Those, who can read spectrograms, use features as short as 3 ms. Standard front ends smear everything shorter than 10 ms, thereby losing important information about consonants.

(6) It was found that the recognizer actually works better if it performs decoding governed by

$$\widehat{W} := \arg\max_W P(A|W)^\alpha P(W)^{1-\alpha} \tag{380}$$

14 ◁ 36 ◁ instead of formula (1) and the fudge factor α is about 1/17 (see 2.8.5). It can be interpreted that we prefer information from the language model over the information we actually hear. In spite of that, we obtain performance improvement, which means that language model was more useful than acoustic model. Therefore it would be natural to improve the acoustic model so that we could get closer to theoretically optimal $\alpha = 1/2$.

(7) According to empirical study in [55], the word error rate of the speech recognizer examined on testing data depends linearly on cross entropy (i.e. logarithm of perplexity) of its language model on that testing data. For their recognizer, they found the following dependence

$$\mathrm{WER}(H) \;\approx\; 34 - 5.4\cdot(7-H) \qquad [\%] \tag{381}$$

fitted from several points with H between 7 and 10 (each point obtained using a different language model), where H is a cross entropy in bits per word. Note that for all-knowing language model having $H=0$ it predicts WER $= -3.8$ % which is reasonably close to zero. This suggests that the linear dependence holds even for $H<7$. Bounds on H of human language model can be estimated by letting people play so called *Shannon game* (described in [39]). Shannon found out (and it was confirmed later by others) that the English language has an entropy somewhere between 0.6 and 1.3 bits per letter. With average word length (of a word randomly picked from text in an English book) being 5.1 characters this gives entropy of 3.1 to 6.6 bits per word. Now, in spite of using the optimistic value of 3 bit/word, the formula still predicts WER of 12.4 %.

So, even if we used the best language model we could ever hope for, the recognizer would still perform much worse than humans. This is my last argument why not to spend much effort on improving language models.

6.2 Design Objectives

From what was said until now, most objectives of the front end will come out as natural. As this is an experimental system, I will not attempt to sacrifice quality for speed. The optimizations could be tried later, once it will be clear whether this approach works.

So, for instance, the sampling frequency will be fixed at 48 kHz (running an external upsampler on files in different format) and the number of filter bank channels will be considerably high.

On the other hand, I will only address monaural inputs in order to save time for more important things. Even though multichannel processing can achieve blind source separation (provided we have more microphones than sources), which is tempting, a good single channel engine would be more practical because of telephony applications. Also the human performance does not drop considerably on monaural sound. Therefore, the search for the ultimate front end should be directed along the single channel route, adding more inputs later.

As pointed out in (4) and (5), high resolution in both the frequency and the time is important. Time events as short as 3 ms must be distinguishable. For this reason, the front end will not use fixed data chunking. Instead, it will be implemented as time-domain linear filterbank[2]. Of course, the 3 ms resolution would be attainable only for frequencies well above 300 Hz.

According to 5.8.9, the Q_{10} factor of each filter of the filterbank should be at least 6 and according to 5.8.8 the frequency response should be of concave shape to make ▷ 173 off-frequency listening possible.

In the real world, short-time features get smeared by the reverb. To make these of any use, the front end has to cope with echo somehow. This will be addressed in section 6.5. It will try to deconvolve the signal before it is ▷ 190 fed into the filterbank. By deconvolution I mean filtering the signal with an approximation of the inverse of the filter that caused the reverb. Unfortunately, it is nearly impossible to obtain this filter just from the smeared signal alone without any prior information, so we would have to get by with an approximation that might fail.

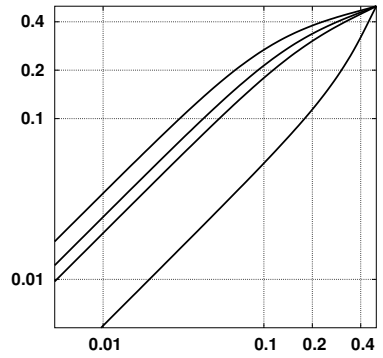


Fig. 44. *PMVDR frequency warping function (the same as in fig. 30) drawn on* ▷ 158 *a logarithmic frequency axis for values of α being 0.55, 0.42, 0.32 and -0.32 for the curves considered from the left to the right respectively. Note that $\alpha = 0.32$ shifts the spectrum one octave upwards and for* ▷ 65 *16 kHz sampling frequency the normalized frequency of 0.2 corresponds to 3.2 kHz.*

Should it happen, there has to be a fall-back strategy (using longer-lasting features) that will be used instead. In fact, this strategy will be just a special case of general system of reliability estimates based on SNR estimation which will run independently for each filterbank output channel. These reliability estimates will be passed to the acoustic model along with the features so that the decoding would not get distracted by interfering noises or missing information, as outlines in 3.3.14.

It should be stressed that the noise is not just the stationary humming of computer fans. It covers all non-speech sounds (or ideally all sounds except the speaker being tracked). Therefore the front end should be able to guess when two events in filterbank's output

[2] I'm not going to simulate frequency masking caused by nonlinear motion of the basilar membrane. Section 3.2.31 justifies this decision mathematically. Also in 5.8.7 we witnessed that the brain tries to ▷ 55 escape frequency masking whenever possible (contrary to temporal masking). ▷ 171

spectrogram came from a single source. As we only have a single channel, it cannot be solved exactly. Nevertheless there are features such as slight fast changes in voice modulation which are very source-specific and may help in tagging the spectrogram by sound source numbers. Another possibility lies in a side-effect of deconvolution — as the target speaker is focused, other sound sources (with different reverb impulse responses, as they are located elsewhere in the room) are likely to get defocused. Hence, sharp pulses would be likely from our speaker (once it has been focused).

Finally, traditional psychological theories [16] could be also tried. According to them, spectrogram events with high level of *similarity, proximity, common-fate* or *good continuation* tend to come from a single source. Unfortunately these theories are only qualitative and do not state how much of proximity and common fate is required for such a judgment nor how to measure 'similarity' and other features. Nevertheless, it could be parametrized somehow and trained from real data, at least in principle. The good news is that we could simulate the room, mix different clean recordings in many different ways, generating vast amount of training data, outwitting the data sparsity problem.

As in any modern front end, we have to tackle the problem of different vocal tract lengths. In accordance with (3), the features should provide two complementary
155 ◁ informations — the size of the sound source and its timbre. As pointed out in 5.4, α times smaller resonator leads to α times higher resonant frequencies.

If we constructed the filterbank so that the channel center frequency would depend on the channel number exponentially, then the change in size would lead to simple translation (on the channel-number axis). Thus, the easiest way of providing complementary size/timber information would be to compute the barycenter (which would indicate the size – the higher the number the smaller the sound source), shifting the spectrogram such that its new barycenter would be constant, then using that new spectrogram as a feature vector for subsequent processing.

As explained in 5.4, real people of different sizes are not just scaled copies of one another. Therefore, not all the phones get transposed by the same amount. Especially fricatives like /s/ and /š/ seem to be nearly speaker independent. So — after all — simple barycentric normalization might not be the best that could be done, but still might serve as a good starting point.

158 ◁ It is instructive to see how this is solved in PMVDR front end of 5.6. The warping function is bended (see fig. 44) but it is fairly linear in formant-dominated region of spectrum so it can transpose it by more than two octaves when going form $\alpha = 0$ to $\alpha = 0.55$. At the same time, the higher frequencies where the fricatives live, are affected only mildly. Nevertheless, this trick seems rather ad-hoc to me, so I decided not to use it at all.

As described so far, the front end produces two-dimensional images called *spec-*
149 ◁ *trograms*, such as those in fig. 25. Note that the low frequency bands tend to change more slowly than the high frequency ones. This comes from the fact that not only the center frequency of the bands are placed uniformly on the logarithmic frequency scale but that the whole transfer function of the filter remains constant under translation (transposition in musical terms) on the logarithmic scale. As a consequence the bandwidth to center frequency ratio is constant and so the bandwidth has to decrease
115 ◁ when going towards lower frequencies. It follows from the sampling theorem 4.10 that

the narrowband signal could be reconstructed from fewer samples and therefore cannot change so wildly as the wideband one.

Also note that the glottal frequency and its several first harmonics are clearly visible in the spectrogram, even for low-pitched voices. This is the downside of high Q_{10} factor, which prevents F_0 and $2F_0$ from occupying a single band, thus preventing creation of pulses known from FFT based spectrograms. The pulses are visible only in high bands where more integer multiples of F_0 fall into filter's passband and the pulses get created by interference.

The problem is that the first formant lies in the region of visible harmonics and has to be specially recovered from these F_0 harmonics tracks. On the other hand, high-pitched voices look the very same way, including pulsed nature of higher formants. This is an advantage over the block-based front ends that have to face the problem of voices of different pitch looking differently (c.f. 5.6).

Originally it was planned that the front end would also provide long timescale features (prosody), namely the prominent syllable detector [60] which could be helpful in inferring word boundaries in some languages (including Czech). Unfortunately, the work in this direction was suspended due to the lack of time.

Finally, I decided that the front end's output will be discrete. There are several reasons that led me to this decision, namely the research [54, 35, 27] showing that it is possible to match the performance of continuous probability density functions, also supported by the fact that PRML decoding (see (131) and 3.3.15) uses only 8 ▷ 66 ▷ 69 quantization levels as it was found that the improvement obeys the law of diminishing returns. The last reason is practical. The proposed front-end is likely to be power demanding. By making its output discrete, we can save some CPU cycles in the decoder, thus hiding the impact of the front end.

Having discrete outputs has a consequence that most of the online speaker adaptation will take place inside the frontend and the labels coming out of it would more or less represent fragments of phones rather then general sounds as is usual.

The question remains how to choose the mapping from spectrograms to the labels. First, let's assume that the effect of F_0 frequency (harmonics over the first formant and pulsed structure of higher formants) has been removed by interpolation between neighbors. The side-effect of this process — the measured value of F_0 — will be retained. Secondly, as the filterbank's highest channels run at the original sampling rate of 48 kHz, the spectrogram should be downsampled to, say, 600 Hz frame rate. It is still high enough to preserve 3 ms features. Then, each frame will be power-normalized, setting the power value aside for later use.

Finally a window of say 24 frames will be fed into the vector quantizer that will transform it into a single natural number. Then the window will move by 12 samples to generate the next output.

On top of that, there will be a mechanism of reliability estimation, working in intimate connection with the quantizer and the acoustic model, as explained later.

6.3 NUFIBA Architecture

The *NUFIBA* acronym stands for *Non-Uniform Filter Bank*, which means that the filters in question are not spaced regularly as in the FFT but their center frequencies

are spaced exponentially instead. So the filters are in fact regular (being a translated copies of one another) when drawn on a log-log paper and so the acronym may also mean *Naturally-Uniform Filter Bank*. But I like the first reading better as it permits possible future extension into truly non-uniform filters. These might have a sense as a way of optimization, trading-off between CPU demands and recognizer's performance, as it is common in the nature (non-uniform sensitivity of our cochlea, fovea of our eyes, and so on). However, for the time being let the filter spacing be naturally uniform in order to make things simple.

It soon became evident that for real world applications the filterbank has to be equipped with other accessories to make it work under harsh conditions. For this reason, the filterbank will be called the *NUFIBA-core* whereas *NUFIBA* refers to the front-end as a whole.

6.4 NUFIBA Core

Let us begin with description of the central part of the front end. It consists of the filterbank followed by Hilbert transforms whose output is then divided into amplitude and frequency components.

6.4.1 The Filterbank

This implementation of filterbank was inspired by [17]. Each channel is realized as a combination of low-pass and high-pass filters. The trick is that the low-pass filters can share each others resources, namely, that the bank of low pass filters can be implemented as a tapped cascade of low-pass sections fed from the side of the highest-frequency filter.

Moreover, whenever the normalized frequencies above 0.25 get attenuated below the noise-floor, the signal gets decimated by 2. This saves a lot of computations[3] and it also makes the filter design easier[4].

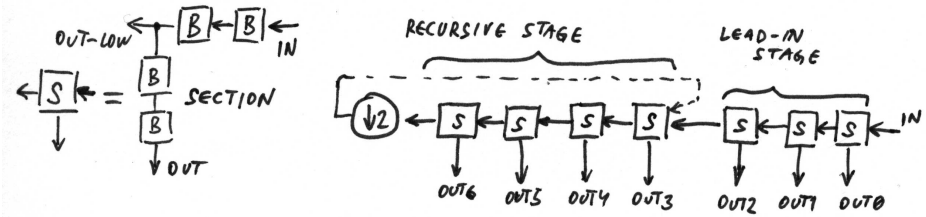


Fig. 45. *NUFIBA filterbank architecture. The left side explains how the section (S) gets assembled out of the blocks (B). On the right side, there are two NUFIBA stages. The data gets decimated after leaving the second stage to be fed into the identical copy of that stage (indicated by a dashed line). There it produces OUT7–10 and the leftover output gets decimated again to feed the next stage.*

Then at each tap, there would be connected a high-pass filter that would finish the job. The overall architecture is depicted in fig. 45. The filters belonging to each output are called a *section*, while the sections belonging to a single octave are called the *stage*. Actually, due to decimation, there are only two physical stages, the lead-in stage and

[3] Actually only up to twice as many operations as it took to compute the first octave will be required to compute all the octaves that follow.

[4] Because the filters do not need to be so steep (on linear frequency scale) as would otherwise be required for the lowest frequencies.

the recursive stage that follows. After the recursive stage, the signal is decimated by 2 and fed back into the copy of the recursive stage (with its own state memory but shared filter's coefficients, which allows for implementation via a cycle making the program code small thus reducing cache pressure). The last filter of the lead-in stage must be carefully designed so that its response after decimation would match the last filter of the recursive stage as closely as possible.

Each section is implemented as a cascade of *blocks* in the first canonical form (fig. 21), where for the low pass filter there are two of them and for the high-pass filter there are four blocks. ▷ 142

Entire filterbank is designed as a minimum phase filter (see 4.7) to ensure stability and earliest possible reaction (see 4.7.11). In the experimental system this is nevertheless unused as the outputs from the high-frequency channels are delayed[5] so that their peak response would line up with the low-frequency ones (consequently the filter is no longer ZP-canceled (see 4.4.6), after this delay has been applied). This measure simplifies development of the system because the instantaneous spectrum is then simply a column taken from the spectrogram. On the other hand, should we be designing a real-time system it might be advantageous to begin processing of the high frequency channels as soon as possible, using the low frequency ones only to disambiguate hard cases. The delay is not negligible. For NUFIBA configuration described below it is 303 ms at the lowest frequency channel (33 Hz). Generally, the delay halves by every octave so it is about 10 ms at 1 kHz. A filterbank with lower Q_{10} would have the delay shorter at the expense of worsened frequency resolution. Additionally, there is a delay caused by subsequent Hilbert transform, making up the total delay to be 329 ms. For off-line experimental system this is of no concern, though. ▷ 101 ▷ 107 ▷ 90

The transfer function H_k of the k-th channel must satisfy the following relation:

$$H_k(f) = H_{k+1}\left(f \cdot 2^{-1/CPO}\right) \qquad \forall\, f \in [0, 1/2] \tag{382}$$

where CPO is the number of channels per octave and the channels are numbered from the highest frequency one, starting by zero. Naturally, the exact equality would be most likely impossible to achieve so we get satisfied with an approximation here. The formula (382) ensures that the same sound transposed by $1/CPO$ octave generates identical but translated image of the original sound.

For the first tryout, I have chosen 12 channels per octave (so that each channel would correspond to a half-tone in the tempered musical scale). The shape of the filter was chosen to approximate a Gaussian (on linear frequency scale). This choice maximizes the product of time and frequency resolutions. Formally, the shape is given by the following formula

$$H_k(f) \approx \exp\left(-\frac{1}{2}\left(\frac{f - f_k}{\sigma f_k}\right)^2\right) \qquad \text{where } f_k = \frac{2^{-(1+k)/CPO}}{2} \tag{383}$$

where σ controls the trade-off between the time and frequency resolution and consequently determines the Q_{10} factor. The chosen value of $\sigma = 0.05$ translates to

[5] The required delays were computed from the group delay (225) at the center of each filter. ▷ 95

$Q_{10} = 6.52$ which leads to bandwidth at 10 dB attenuation to be $B_{10} = 0.222$ octave, while at the -80 dB it is $B_{80} = 0.63$ octave.

To allow the digital filter some freedom in approximating the target shape, we don't insist on exact shape below -80 dB as long as the transfer function stays below and goes below -96 dB at the right side so that the aliasing noise would stay under the noise floor of 16 bit numbers.

110 ◁ It follows from 4.7.16 that the phase response of a minimum phase filter is completely determined by its magnitude response (up to a constant phase shift). The question is what happens when we squeeze the response according to (382). Ideally, we would like to see that the phase response has been squeezed the same way. Theorem 4.7.16 states that the following holds for the original magnitude and phase response H_L and φ:

$$\mathrm{sgn}^2 \odot \mathcal{F}^{-1}(\varphi) = -i\,\mathrm{sgn} \odot \mathcal{F}^{-1}(H_L) \tag{384}$$

Now, squeezing the responses according to (382) and padding the gap with something close to zero, so that the resulting response would be in $\mathcal{S}_D$, we get $\widehat{H}_L$ and $\widehat{\varphi}$. Obviously, the signal $\mathcal{F}^{-1}(\widehat{\varphi})$ is somehow upsampled version of $\mathcal{F}^{-1}(\varphi)$ and likewise for $\widehat{H}_L$.

Now observe that the equality no longer holds exactly because the values of neighboring samples were mixed by convolution with an interpolating filter, a digital filter
119 ◁ approximation of (301), during upsampling. The exact span of the interpolator depends on how we did the padding. In combination with the change of sign introduced by multiplication with the sgn sequence, this creates discrepancy near zero indexes.

On the other hand, it suggests that if we did less radical padding the effect of convolution and consequent mixing around zero could be limited and the equality would hold at least approximatively.

In any case, this means that if the squeezing preserves the shape of H exactly, it inevitably introduces an error into the phase. This is a second reason why the response shape is left free below 80 dB — it creates a room for phase correction. Otherwise, the phase would be completely determined by (384).

6.4.2 Searching for the Filterbank

The NUFIBA filterbank is fully described by the list of poles and zeroes in its respective blocks. Each block is characterized by a pair of complex conjugated poles and a pair of complex conjugated zeroes. This comprises two complex numbers of absolute value less than 1 (a minimum phase property). Those numbers are kept in polar representation for each block.

To obtain the filter, I specified the cost function that measures how far the current the configuration of zeroes and poles is from the desired shape (383) and used modified[6] Brent's method [51] for multidimensional search to find a minimum. The Brent's method performs a search similar to conjugate gradients but it does not need derivative of the cost function, which makes it easier to use.

The cost function first computes complex frequency response of entire filterbank on a grid of 'measuring frequencies'. Then, for each band, it calculates the distance (in decibels) from the desired shape (383) weighted so as to neglect differences from it under -80 dB as described above.

[6] My modification of Brent's method allowed to constrain the search onto an interval in each dimension. This was used to ensure that zeros and poles stay within the unit circle.

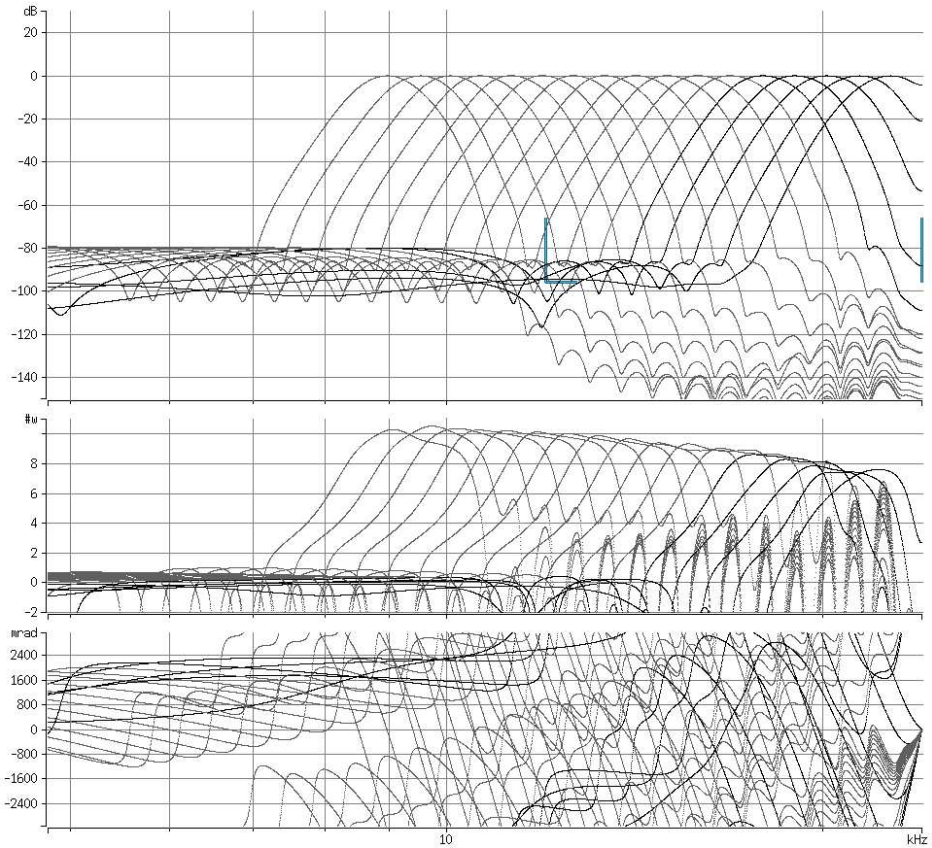


Fig. 46. *NUFIBA amplitude response, wave delay and phase response. Five darker lines on the right comprise the lead-in stage, while the lighter ones constitute the recursive stage.*

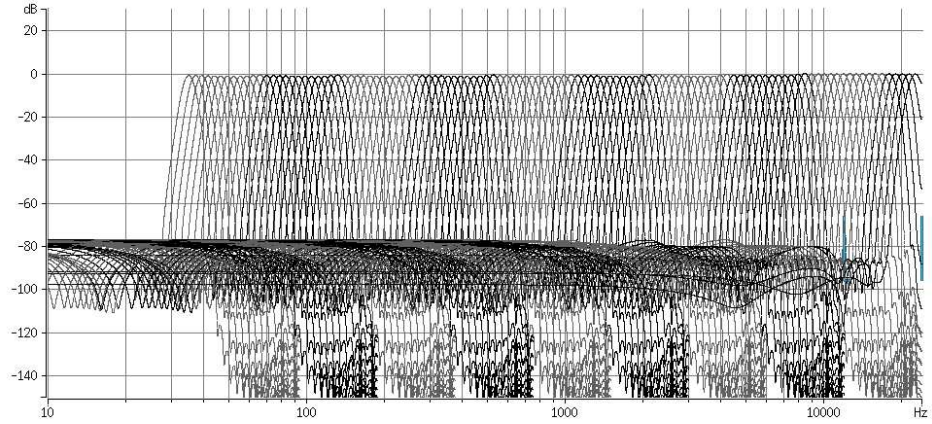


Fig. 47. *All NUFIBA channels for 9 recursive octaves. Compare with fig. 28.* ▷ 152

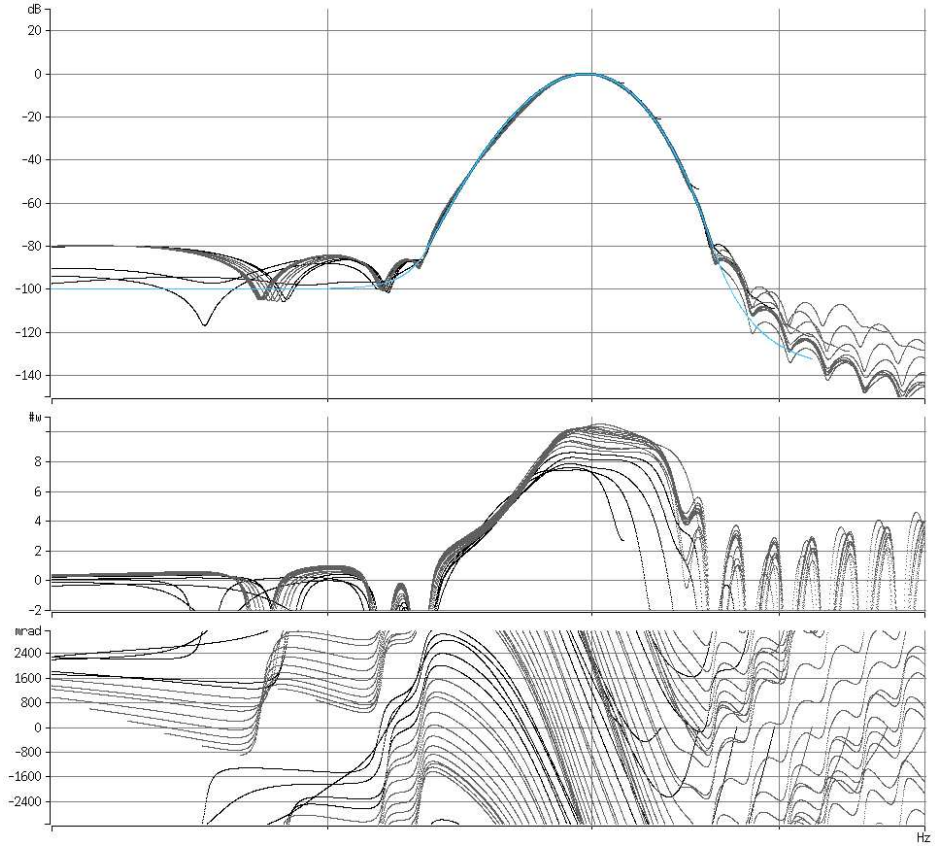


Fig. 48. *The channels of fig. 46 drawn over one another so that the departure from the target shape would become visible.*

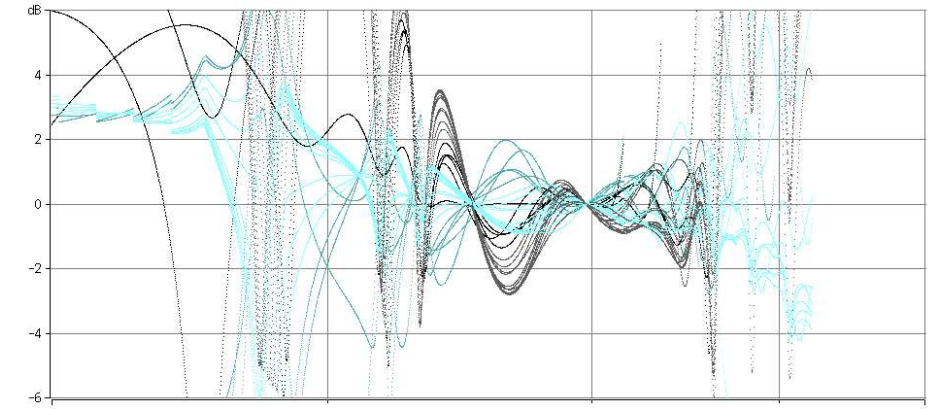


Fig. 49. *Difference from the target shape and from the average shape (light lines).*

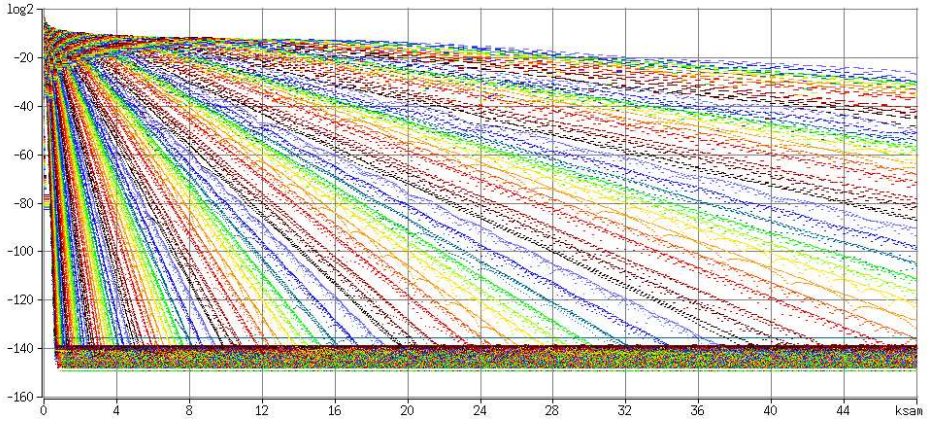


Fig. 50. *NUFIBA decay test. On the horizontal axis there is time measured in samples (so 48 k corresponds to 1 second) whereas on vertical axis there is $\log_2(|A_k(t)|)$, where A_k denotes k-th NUFIBA output channel. We can see that the slowest channel decays below 2^{-20} in less than a second.*

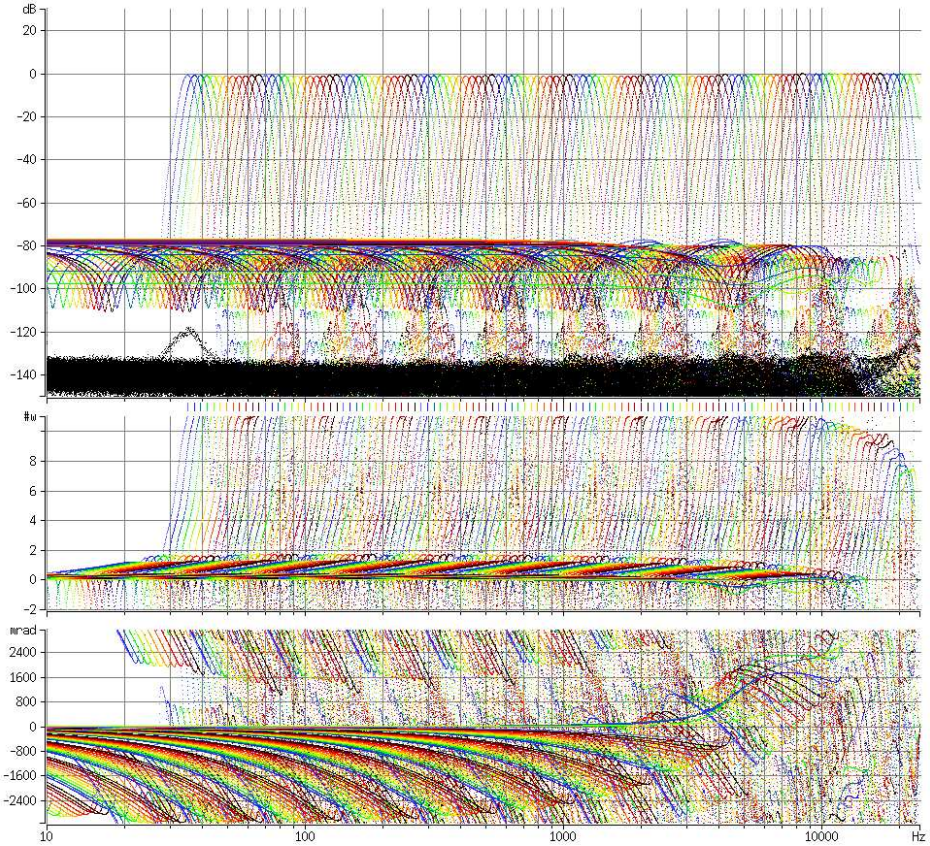


Fig. 51. *Color lines represent individual filters, while the black dots represent measured noise floor.*

182 ◁ This is added to similar distance from the right half of (383), which is used to constraint the shape of low-pass filter taps.

Finally, neighboring filter shapes are moved (on a log scale) to lie on each other and a distance of the complex responses is calculated for each measuring frequency. The result is added to the total cost. This has to force phase responses to try to approximate one another, as discussed above. A similar technique is used to match the last filter of the recursive stage with the last filter of the lead-in stage.

The number of channels in the lead in stage was chosen to be 5. With $CPO = 12$ this makes 17 segments each made of 2 low-pass and 4 high-pass blocks. This leads to $17{\cdot}6{\cdot}4 = 408$ dimensional search. Fortunately, many dimensions are nearly independent, so it is possible to run local searches along these to quickly obtain initial guess for the final global search. The optimization took about a day and the resulting filters can be
184 ◁ seen in figures 46–49.

6.4.3 Testing the Filterbank

144 ◁ As pointed out in 4.13.6, the roundoff errors can be tricky in IIR filters, so it is a good practice to test the filter prior to actually using it. The testing program feeds the filterbank with the unit impulse signal $\vec{1}$. The outputs of all channels are then plotted as a logarithms of their absolute value. This is shown in fig. 50. This plot can be used to judge evenness of the channel's decay rate as well as the filter's floor level, where the limit cycling takes place. For our purpose the filterbank passes this test.

The second test feeds the filterbank with a sine signal of specified frequency and measures its output (after waiting some time for the transient response to vanish). It
184 ◁ plots the frequency response of the bank (which should be the same as in fig. 47) and the level of non-sine signal find in the output caused by roundoff errors. It can be seen from fig. 51 that it is well below the quantization noise.

6.4.4 Hilbert Transforms

Each output channel of the filterbank is connected to the quadrature filter which recovers analytic signal from it. The reason for this is twofold. First, it is used to compute an envelope of each channel. Second, the phase information can be used on its own to further refine the spectrogram or to measure frequency of steady tones beyond the precision of NUFIBA channel spacing.

Each channel's quadrature filter is implemented by the Hilbert transform (as ex-
124 ◁ plained in 4.11). However, as the channels are already narrowband, we do not need true
125 ◁ Hilbert filter (309) which would be problematic because of its slow convergence. Equivalent function can be achieved with something that only approximates the Hilbert filter well over the channel's passband. This technique can reduce the number of filter taps from several hundreds that would be needed for true Hilbert to just 11. The downside is that each channel must have its own (different) filter centered over its passband. But this is a little price to pay considering that it even does not affect the running time.

So how do we find the Hilbert filter? By a brute force search again. Now we are going to search for a FIR filter with odd impulse response (so that its spectral response would be purely imaginary). The 11-tap filter needs just 5 parameters to be determined (the middle tap is always 0), which is pretty easy. Note that we do not insist on even taps to be zero as it is in (309). The rationale is that we in fact search

for (309) convolved with some band-pass filter and the convolution smears the (309)'s response, making even taps possibly non-zero.

The cost function being minimized first evaluates the Hilbert filter response on a grid in the desired interval of frequencies and measures maximum positive and negative error in decibels. These two are squared and summed to form the basic cost. This is further penalized if there are too large ripples (not only in the band of interest) or if the response goes negative.

To help the optimizer from getting lost when it is far from the optimum, the penalty is multiplied with mean quadratic error of the filter's amplitude response from the desired shape (i.e. 0 dB) in the desired frequency range — this guides the search when the penalties are high.

The result of the search is summarized in the following program's output. Note that the filters in the lead-in stage receive rather inaccurate filters. Fortunately, the recursive stage is acceptable. Also note that this is a maximum error, often found at the edges of the interval (where the signal gets weak anyway), so the filters are not as unusable as they look at the first sight. There is response of one of them in fig. 52, to ▷ 189 illustrate the matter.

```
22653.0Hz: maximum error in 17750.4Hz..23326.5Hz is 34.508dB=5213.9766%
21381.6Hz: maximum error in 16754.1Hz..22690.8Hz is  2.564dB= 34.3403%
20181.5Hz: maximum error in 15813.8Hz..22090.8Hz is  1.674dB= 21.2515%
19048.8Hz: maximum error in 14926.2Hz..21524.4Hz is  1.198dB= 14.7913%
17979.7Hz: maximum error in 14088.5Hz..20989.8Hz is  0.815dB=  9.8369%

16970.6Hz: maximum error in 13297.8Hz..20485.3Hz is  0.723dB=  8.6748%
16018.1Hz: maximum error in 12551.4Hz..19544.2Hz is  0.335dB=  3.9358%
15119.1Hz: maximum error in 11847.0Hz..18447.2Hz is  0.125dB=  1.4468%
14270.5Hz: maximum error in 11182.0Hz..17411.9Hz is  0.027dB=  0.3162%
13469.5Hz: maximum error in 10554.4Hz..16434.6Hz is  0.006dB=  0.0680%
12713.6Hz: maximum error in  9962.1Hz..15512.2Hz is  0.002dB=  0.0223%
12000.0Hz: maximum error in  9402.9Hz..14641.6Hz is  0.000dB=  0.0029%
11326.5Hz: maximum error in  8875.2Hz..13819.8Hz is  0.001dB=  0.0116%
10690.8Hz: maximum error in  8377.1Hz..13044.2Hz is  0.001dB=  0.0164%
10090.8Hz: maximum error in  7906.9Hz..12312.1Hz is  0.002dB=  0.0202%
 9524.4Hz: maximum error in  7463.1Hz..11621.0Hz is  0.008dB=  0.0958%
 8989.8Hz: maximum error in  7044.2Hz..10968.8Hz is  0.014dB=  0.1606%
```

6.4.5 Running the NUFIBA-core

The NUFIBA-core consists of the nufiba filter followed by the Hilbert filters, producing complex-valued analytic signals. The samples of the analytic signal c_n are then recomputed into log-amplitudes a_n and normalized log-frequencies b_n as follows.

$$\begin{aligned} a_n &= 10 \log_{10}(1 + |c_n|^2) \\ b_n &= \log_2 \left(\frac{\mathrm{Arg}(c_{n+1}/c_n)}{\pi\, 2^{-(1+k)/CPO}} \right) \end{aligned} \tag{385}$$

where n represents discrete time and k is the NUFIBA channel number (0 corresponding to the highest one). In the spectrograms so far, a_n was drawn. The quantity b_n measures

how far off the channel's center frequency the signal is. This is expressed in octaves, so 0 means exactly the channel's frequency, +1 one octave higher, -1 one octave lower. Due to narrowband nature of NUFIBA, useful range of b_n is about $(-0.3, 0.3)$. Beyond this the attenuation is too high for the output to be reliable (because of noise).

The advantage of the relative scale is that if the F_0 has certain frequency modulation, its m-th harmonics is going to vary in frequency m-times as much but when expressed by means of b_n, the same sequence of numbers appears[7].

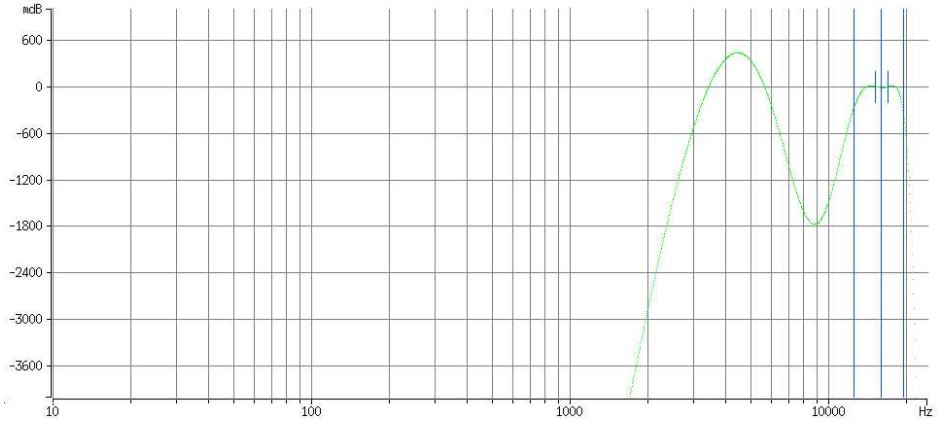


Fig. 52. *Amplitude response of Hilbert filter attached to NUFIBA channel number 6, the one with center frequency 16018.1Hz. The blue lines indicate the width of the Gaussian shaped NUFIBA filter, the short lines are offset from the center by the NUFIBA channel spacing, thus denoting centers of the closest channels. The phase response is $-\pi/2$ everywhere.*

Before passing the data to the subsequent processing, the spectrogram gets downsampled by 64 (for the highest channels) so that the resulting frame rate would be 750 Hz. Note that for the lowest channels this does not mean downsampling at all because their sampling frequency is even lower than that (their samples just get repeated so that the entire spectrogram would be of a single timescale).

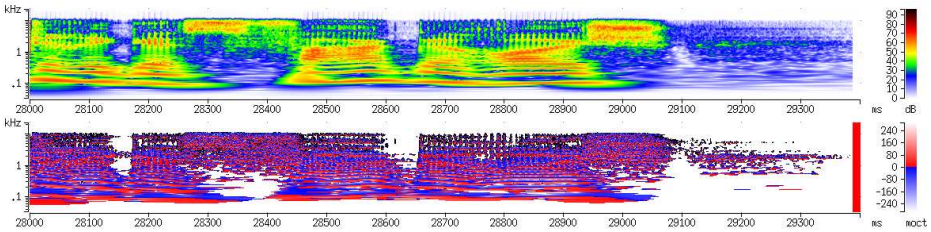


Fig. 53. *Example of amplitude (a_n) and frequency (b_n) NUFIBA outputs.*

The downsampling works by selecting the highest value in the interval in question for a_n signals (to retain high SNR), and by LMS fitting of line segment thru the b_n samples. The average value of b_n's is used as the downsampled output, but only when their mean square difference from the fitted line is below $1.5/CPO$ (which is 0.125 in our case). If it is above, artificially high value (say 1) is used, which leads to rejection of the value as noise in later processing.

[7] Apart from minor complication that the signal gets downsampled every octave and this must be accounted for.

In figure 53 there is a frequency spectrogram (the lower one) together with its amplitude counterpart. See how the formants that cannot be easily distinguished from each other in the amplitude spectrum get separated in the frequency spectrum. Also note that low intensity formants are clearly visible in the frequency spectrum too. Unfortunately, there is also much noise there and so the frequency spectrum cannot be used directly.

6.4.6 Notes on Future Filterbank Designs

Using higher Q_{10} would be possible but firstly, I did not want to stretch filter design to the limits (steeper filters either require more zeros and poles or cannot capture the required shape precisely) and secondly, the problem of visible harmonics of F_0 would be even worse, perhaps interfering even with F_2 formant.

On one hand we need wide filters so that F_1 would be directly readable but on the other hand we need high Q_{10} to be able to separate close higher formants. The question arises whether it would not pay off to use 2 filterbanks, one with high Q_{10} and the other with low. Then, after finding F_0 we could choose the resolution that best suits the channel we are about to extract.

This is what I had in mind in section 5.8.4 when I was speculating about the role of OHC-in signals. If these could manipulate Q_{10} of the filters involved, the listener could tune his filters to the speaker so that the formants would be optimally visible just before the bands of F_0 harmonics would appear. In any case, human filterbank has lower Q_{10} at lower frequencies, thereby mitigating the problem of F_0 harmonics on average. ▷ 169

Future NUFIBA implementations may use several filterbanks with different Q_{10} filters, or these could be really non-uniform as in the real cochlea, or we could just try to combine raw outputs from the neighboring channels to obtain wider bands when these are needed. The mixing coefficients would be variable, serving the (theorized) purpose of OHC-in fibers.

The last remark concerns the implementation. The future filterbank might try to use differences of the neighboring low pass filter taps to obtain initial high-pass filtering for free. The idea is that it would need fewer high-pass blocks then. However, the search for the filterbank's parameters will be more demanding (actually I have already tried this but have not succeeded).

6.5 Blind Focusing

This section concerns with detailed description of how the echo suppression works. First we need a little more math.

6.5.1 Definition *Circulant Matrix*

Circulant C is defined as an $N \times N$ matrix such that $C_{ij} = c_{(i-j)\%N}$, where the vector c is its zeroth column. We write $\mathrm{circ}(c)$ for such a circulant generated from c. Let $\mathrm{circ}_M(c)$ denote $N \times M$ stripe made from first M columns of $\mathrm{circ}(c)$. It is also referred to as a *circulant* despite not being a square matrix.

6.5.2 Definition *Toeplitz Matrix*

An $N\times N$ matrix T such that $T_{ij}=t_{i-j+N-1}$, where t is its upper-left circumferential vector of length $2N-1$ is called the *Toeplitz matrix*. In another words, Toeplitz matrices have constant diagonals whose values are determined by the vector t. For such a T we write $\mathrm{toep}(t)$. We also write $\mathrm{toep}_M(t)$, to denote $N\times M$ Toeplitz stripe $\mathrm{toep}_M(t):=t_{i-j+M-1}$. In this case, t would have only $M+N-1$ elements.

6.5.3 Example Every circulant is also a Toeplitz matrix but not the contrary.

$$\mathrm{circ}\left(\langle 1,2,3\rangle\right)=\begin{pmatrix}1&3&2\\2&1&3\\3&2&1\end{pmatrix}\qquad \mathrm{toep}\left(\langle 1,2,3,4,5\rangle\right)=\begin{pmatrix}3&2&1\\4&3&2\\5&4&3\end{pmatrix}\tag{386}$$

6.5.4 Notation Let us use $\mathrm{diag}(d)$ for a diagonal matrix D s.t. $D_{ii}=d_i$.

The reason why we get so interested in circulant and Toeplitz matrices is that they allow elegant formulation of convolution. For vectors x and y, the matrix product
73◁ $\mathrm{circ}(x)y$ is equal to $\mathrm{circ}(y)x=x*y$, where $*$ denotes cyclic convolution defined in 4.1.5. Moreover, for an N-tap FIR filter h and $M+N-1$ long vector x being an excerpt from a signal $\widehat{x}$, the M-element vector $\mathrm{toep}_N(x)h$ in fact represents those samples of signal $\widehat{x}*h$, whose value does not depend on values of $\vec{x}$ beyond the excerpt (i.e. those for which h remained inside x while computing convolution).

6.5.5 Theorem *Circulant gets diagonalized by the discrete Fourier transform. Formally, for any vector $c\in\mathbb{C}^N$ the following holds.*

$$\mathrm{circ}(c)=\overline{F_N}\,\mathrm{diag}(d)F_N\qquad \textit{where}\quad d=\sqrt{N}F_N c\tag{387}$$

73◁ **Proof** See 4.1.6, which is an equivalent proposition, already proven. Q.E.D.

6.5.6 Note The last theorem shows that multiplying the vector by a circulant can be done quickly via the FFT. The same holds for its inversion (if it exists): $\mathrm{circ}^{-1}(c)=\overline{F_N}\,\mathrm{diag}^{-1}(d)F_N$. Although general Toeplitz matrix cannot be diagonalized the way circulants can, we can still multiply Toeplitz times vector quickly by realizing that

$$\mathrm{circ}\begin{pmatrix}0\\c_1\\\gamma\\c_2\end{pmatrix}=\begin{pmatrix}\mathrm{toep}\begin{pmatrix}c_2\\0\\c_1\end{pmatrix}&\mathrm{toep}\begin{pmatrix}c_1\\\gamma\\c_2\end{pmatrix}\\\mathrm{toep}\begin{pmatrix}c_1\\\gamma\\c_2\end{pmatrix}&\mathrm{toep}\begin{pmatrix}c_2\\0\\c_1\end{pmatrix}\end{pmatrix}\tag{388}$$

where c_1 and c_2 are $N-1$ dimensional vectors, γ is a number and the whole matrix is $2N\times 2N$ circulant. Therefore 6.5.5 can be used to carry-out multiplication with N-dimensional vector of our interest, padded by N trailing zeroes, tossing away the part of the result we are not interested in.

Sometimes it also comes in handy to approximate Toeplitz matrix T by circulant C of the same dimensions, such that the quantity $\sum_{ij}|C_{ij}-T_{ij}|^2$ would be as small as possible. In fact, any matrix can be approximated by circulant in this manner and the optimum is remarkable: The best approximating circulant can be obtained from the source matrix A by averaging along its (circulant) diagonals. Formally:

6.5.7 Theorem *For any matrix $A\in\mathbb{R}^{N\times N}$ we have*

$$\left(\arg\min_{\vec{c}}\sum_{ij}\left|\,\mathrm{circ}(\vec{c})_{ij}-A_{ij}\right|^2\right)_p=\frac{1}{N}\sum_i A_{i,(i-p)\%N}\tag{389}$$

Proof Differentiating by c_p and setting the result to zero, we get

$$\frac{\partial}{\partial c_p}\sum_{ij}\left|c_{(i-j)\%N}-A_{ij}\right|^2=\sum_{ij}2(c_{(i-j)\%N}-A_{ij})\cdot I\big((i-j)\%N=p\big)\tag{390}$$

where $I(P):=P\,?\,1:0$ is the indicator function. This immediately gives the following.

$$Nc_p=\sum_{(i-j)\%N=p}A_{ij}\tag{391}$$

Q.E.D.

6.5.8 Noise Canceling

Now we are far enough to reveal where we are heading to. First we are about to investigate noise canceling. This is not a blind technique but the rationale is to develop technology that will be used in blind focusing later. The noise canceling is even worthwhile by itself in free-microphone dialog systems where it can suppress the computer's voice, so that the computer would not hear itself.

Consider the following situation. We have a real-valued signal a being transmitted thru the loudspeaker to the room, where it gets recorded by a microphone as y. Let us suppose that the recorded signal consists of external sounds z and reverberated copy of a, here modeled as $x*a$, where x is FIR approximation of the room's impulse response[8]. Therefore $y=x*a+z$. Here, the a is the *noise* that we want to cancel in order to learn about z. We don't know x but, fortunately, we know the original a. We are going to find $\widehat{x}$ such that $\widehat{x}*a$ best approximates y in the least squares sense.

$$\widehat{x}=\arg\min_x\left|\left|\,\mathrm{toep}_N(a)x-y\,\right|\right|^2\tag{392}$$

The noise canceled signal $\widehat{z}$ will then be $y-\mathrm{toep}_N(a)\widehat{x}$. Let us first develop how to solve $\arg\min_x||\,\mathrm{circ}_N(a)x-y\,||^2$, which is easier. It can be even the method of choice if we knew that a is close to zero near its ends and so the wraparound would not cause harm.

6.5.9 Solving Tall Circulant Systems of Equations

Let $A:=\mathrm{circ}_N(a)$ be $M\times N$ circulant matrix. Then $||Ax-y||^2=x^{\mathrm{T}}A^{\mathrm{T}}Ax+y^{\mathrm{T}}y-2x^{\mathrm{T}}A^{\mathrm{T}}y$. Setting the gradient by $\vec{x}$ to zero, we get:

$$\vec{0}=\nabla_x||Ax-y||^2=2A^{\mathrm{T}}Ax-2A^{\mathrm{T}}y\tag{393}$$

So for finding minimum we have to solve the following system of linear equations.

$$A^{\mathrm{T}}Ax=A^{\mathrm{T}}y\tag{394}$$

Note that the multiplications by vector and even the product $A^{\mathrm{T}}A$ can be performed quickly because A is circulant. The latter follows from the following observation.

[8] Think of x being about 2 seconds long but still much shorter than the duration of signal a.

6.5.10 Observation *Let $a \in \mathbb{R}^M$. Then for $N \leq M$ the $N \times N$ matrix*

$$(\mathrm{circ}_N(a))^{\mathrm{T}} \mathrm{circ}_N(a) \tag{395}$$

is a Toeplitz matrix.

Proof Using the fact that

$$\mathrm{circ}_N(a) = \mathrm{circ}(a) \begin{pmatrix} I_N \\ O_{(M-N)\times N} \end{pmatrix} \tag{396}$$

we get

$$\mathrm{circ}_N(a)^{\mathrm{T}} \mathrm{circ}_N(a) = (\, I_N \quad O\,) \,\mathrm{circ}(a * a) \begin{pmatrix} I_N \\ O \end{pmatrix} \tag{397}$$

which means that the product is the upper left corner of certain circulant matrix, that is obviously a Toeplitz matrix. Q.E.D.

Now we see that the $A^{\mathrm{T}}A$ matrix is Toeplitz and we need to solve that system of equations. If it was circulant, we could use its inversion right away, but this is not usually applicable to Toeplitz.

6.5.11 Positive Semidefinite Toeplitz Solver

For this reason, I fell back to general method of conjugate gradients (consult [40] for painless introduction into the topic). However, $A^{\mathrm{T}}A$ is often ill conditioned a so the method needs to be preconditioned, otherwise it would be too slow. Fortunately, circulant approximation 6.5.7 suits this purpose well. Writing B for $A^{\mathrm{T}}A$ and C for its circulant approximation according to 6.5.7 the algorithm that converges to x, a solution of (394) can be stated as follows.

$$\begin{aligned} r &:= y - Bx \\ d &:= C^{-1}r \\ &\text{repeat until convergence:} \\ \alpha &:= \frac{r^{\mathrm{T}}C^{-1}r}{d^{\mathrm{T}}Bd} \\ x &:= x + \alpha d \\ w &:= r - \alpha Bd \\ \beta &:= \frac{w^{\mathrm{T}}C^{-1}w}{r^{\mathrm{T}}C^{-1}r} \\ d &:= C^{-1}w + \beta d \\ r &:= w \end{aligned} \tag{398}$$

Note that C^{-1} is also a circulant and that it can be quickly precomputed. However, the *Untransformed Preconditioned Conjugate Gradient Method* (398) needs C to be symmetric and positive definite as explained in [40]. So in fact we take $\widehat{C}^{-1}$ instead of C^{-1} in (398), where $\widehat{C}$ is obtained from $C = \overline{F}\,\mathrm{diag}(h)F$ by by setting zero h_k's to 1 and non zero ones to $|h_k|$. We don't need to worry about the symmetry because $A^{\mathrm{T}}A$ is already symmetric, which is preserved by the approximation 6.5.7.

Also note that we need two matrix multiplications per iteration. The first one is Bd (where B is Toeplitz) and the second (circulant) one is $C^{-1}w$. The third, $C^{-1}r$, is in fact $C^{-1}w$ from the previous iteration. So only two multiplications have to be performed (the first one taking $4N\log_2(2N)$ operations, whereas the second one only $2N\log_2 N$).

The recurrence should stop when $r^{\mathrm{T}}C^{-1}r$ falls below $\varepsilon^2\widehat{r}^{\mathrm{T}}C^{-1}\widehat{r}$, where ε is required precision of the outcome and $\widehat{r}$ is the initial value of r. The result will then be stored in x. Note that it does not make much sense using $\varepsilon < \sqrt{\varepsilon_m}$, where ε_m is the machine epsilon introduced in 4.13.5. ▷ 143

Empirically, it takes tens to several hundreds iterations of (398) before x converges to 6 significant digits (starting from $\vec{0}$), depending on the condition number of $A^{\mathrm{T}}A$.

6.5.12 Solving Tall Toeplitz Systems of Equations

Let us return to our original problem of solving

$$\mathrm{toep}_N(a)x \approx y \tag{399}$$

in the least square sense (392), where $a \in \mathbb{R}^{M+N-1}$ and $y \in \mathbb{R}^M$. The method works by extending Toeplitz stripe on its top to the circulant stripe and extending the right hand side as well, so we would have

$$\mathrm{circ}_N(a)x = \begin{pmatrix} z \\ y \end{pmatrix} \qquad \text{where } z \in \mathbb{R}^{N-1} \tag{400}$$

If z was already set to $(I_{N-1}O)\,\mathrm{circ}_N(a)$ multiplied by the solution x, then the solution of (400) computed by (394) would give the correct answer to the original Toeplitz problem.

Of course we don't have such z but we can obtain it in an iterative way, starting from initial guess x and $q := x$, using the following iteration:

$$\begin{aligned} z &:= (I_{N-1}\ O)\,\mathrm{circ}_N(a)\big(x + \beta(x-q)\big) \\ w &:= \arg\min_x \left|\left|\mathrm{circ}_N(a)x - \begin{pmatrix} z \\ y \end{pmatrix}\right|\right| \qquad \text{solved as (394) by (398)} \\ q &:= x \\ x &:= w\frac{y^{\mathrm{T}}\,\mathrm{toep}_N(a)w}{||\,\mathrm{toep}_N(a)w||^2} \end{aligned} \tag{401}$$

The parameter $\beta \in [0,1)$ controls so called *Richardson Extrapolation*, that accelerates the convergence. I ramp it from 0.4 to 0.93 in first few iteration, letting it 0.93 for the rest. The method usually converges in 10 to 20 iterations but note that each iteration of (401) involves possibly hundreds of iterations of (398). The last line of (401) scales the vector w to $x = \alpha w$ so that $||\mathrm{toep}_N(a)x - y||$ would get as small as possible.

6.5.13 Blind Focusing

The previous method can be used to remove known computer's voice from the recording and it actually works quite well suppressing it by 50 to 70 dB, depending on quality of the loudspeakers. But it does not help in removing echo from the recordings because it does not know the impulse response of the room.

With a single microphone the situation is helpless in this respect. On the other hand, the reverb is a real problem as it smears fine formant movements as can be seen in fig. 59, making phonemes look similar to one another. ◁ 203

So, as we know that the exact solution is impossible, we might be grateful for an approximative one. Similar problems in computer vision led to the development of so called *blind deconvolution* techniques. These try to estimate the repairing impulse response from general knowledge of the distortion mechanism and general knowledge of the class of clean signals. The method of *Blind Focusing* works similarly, belonging among blind deconvolution techniques.

The idea is that even if we cannot know true impulse response we still know that the restored signal should be a voiced phoneme from time to time and voiced phonemes tend to be of pulsed nature. Provided that we knew true positions of glottal pulses we might want to find an impulse response that, when convolved with the smeared signal, reflows the energy towards these pulses.

Of course we would not know exact positions of these pulses, but we can first guess them from F_0 (that can be measured with some success even in highly damaged recording), focus the signal somewhat and repeat the whole process. In later stages we may even use NUFIBA output to further improve these positions.

Let $a \in \mathbb{R}^{M+N-1}$ represent the measured signal, $x \in \mathbb{R}^{N-1}$ the reconstruction impulse response and $w \in [0,1]^M$ the weighing vector of desired glottal pulse positions. Then the method will maximize the following

$$\widehat{x} := \arg\max_x \left|\left| w \odot (\mathrm{toep}_N(a) \cdot x) \right|\right|^2 \qquad \text{with respect to } \left|\left| \mathrm{circ}_N(a) \cdot x \right|\right|^2 = ||a||^2 \tag{402}$$

The first part favors those vectors x that move the signal $a * x$ towards regions of high w_n values (note that $w_n \in [0,1]$). The second part then ensures the conservation of energy (the first part would diverge without it). The reason for using circulant matrix for that is that Toeplitz matrix could hide up to $N-1$ samples depending on particular value of x, which would be incompatible with $||a||^2$ that uses entire a for energy calculation. Circulant also leads to faster algorithm.

The downside of using circulant is that it mixes data from the opposite ends of vector a. But this can be avoided by defining the first $N-1$ places in a as zeroes. Then the wrap-around would not harm. This measure does not affect the first formula at all because the corresponding w_ns will be also set to 0 so that this part of a would be ignored. Moreover, the Toeplitz matrix in the first formula can also be superseded by circulant, which only makes the vector w longer by $N-1$ elements. These have to be placed before the original w and set to 0. So, the new w has $2N-2$ zeroes at the beginning. Having this in mind, we can rewrite (402) into the following equivalent form.

$$\widehat{x} := \arg\max_x x^{\mathrm{T}} A^{\mathrm{T}} W A\, x \qquad \text{with respect to } x^{\mathrm{T}} A^{\mathrm{T}} A\, x = a^{\mathrm{T}} a \tag{403}$$

where $A = \mathrm{circ}_N(a)$ and $W = \mathrm{diag}(w)$ and $w \in \mathbb{R}^{M+N-1}$ with first $2N-2$ places set to 0, and first $N-1$ places of a being zeroes.

Now we use the method of Lagrange multipliers[41] to locate the maximum. In accordance with (316), there is a single constraint function $g_1(x) = x^{\mathrm{T}} A^{\mathrm{T}} A\, x - a^{\mathrm{T}} a$, hence the set of allowed solutions is $A = \{x \in \mathbb{R}^N \mid x^{\mathrm{T}} A^{\mathrm{T}} A\, x = a^{\mathrm{T}} a\}$. For the method to work we need to know that $\nabla g_1[A] \not\ni \vec{0}$. If the set $\nabla g_1[A] = \{2A^{\mathrm{T}} A\, x \mid x^{\mathrm{T}} A^{\mathrm{T}} A\, x = a^{\mathrm{T}} a\}$ contained $\vec{0}$ then $x^{\mathrm{T}} A^{\mathrm{T}} A\, x = ||a||^2 = \vec{0}$, which means that we can use the method only on non-zero input signals a. For those the Lagrange multipliers give the following system of equations (in x and λ) as a solution. ▷ 129 ▷ 129

$$A^{\mathrm{T}} W A\, x = \lambda A^{\mathrm{T}} A\, x \qquad\qquad x^{\mathrm{T}} A^{\mathrm{T}} A\, x = a^{\mathrm{T}} a \tag{404}$$

Any solution of (404) represents a stationary point of the original constrained problem, which includes both its minimum and the maximum. Also note that the value of λ is bounded between 0 and 1, which can be seen from multiplying (404) by x^{T} from the left.

$$x^{\mathrm{T}} A^{\mathrm{T}} W A\, x = \lambda x^{\mathrm{T}} A^{\mathrm{T}} A\, x = \lambda a^{\mathrm{T}} a \tag{405}$$

Writing b for Ax, this gives

$$\lambda = \frac{b^{\mathrm{T}} W b}{b^{\mathrm{T}} b} = \frac{b^{\mathrm{T}} W b}{a^{\mathrm{T}} a} \tag{406}$$

Since $W = \mathrm{diag}(w)$ and $w_k \in [0,1]$, we can see from the first equality that $\lambda \in [0,1]$. The second equality provides interpretation of λ as a ratio of energy of the focused signal multiplied by the weight to the original signal. By (402) we want the solution with highest possible λ.

The equation (404) is known as the *Generalized Eigenproblem*, described in [65]. For regular $A^{\mathrm{T}} A$ it can be reduced to the following equivalent ordinary eigenproblem.

$$\left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}} W A\, x = \lambda x \tag{407}$$

6.5.14 Theorem *(Power Iteration) For symmetric positive semidefinite matrix M, the iteration*

$$\vec{v}_{n+1} := \frac{M \vec{v}_n}{||M \vec{v}_n||} \tag{408}$$

converges towards the eigenvector belonging to the highest eigenvalue for any starting vector $\vec{v}_0$ that is not perpendicular to that highest eigenvalue eigenvector.

Proof Due to symmetry of M, its eigenvectors fill the whole space and can be chosen to be mutually orthogonal. Let us have them stored as columns in orthonormal matrix V. Then $\Lambda := V^{\mathrm{T}} M V$ is diagonal matrix $\mathrm{diag}(\lambda)$ of M's eigenvalues. Due to M being positive semidefinite all $\lambda_k \geq 0$. The recurrence can then be rewritten into

$$\begin{aligned} \vec{w}_n &:= M^n \vec{v}_0 = V \Lambda^n V^{\mathrm{T}} \vec{v}_0 \\ \vec{v}_n &:= \frac{\vec{w}_n}{||\vec{w}_n||} \end{aligned} \tag{409}$$

It is obvious now, that the highest eigenvalue dominates the Λ^n very soon (actual rate depends on its ratio with the second highest eigenvalue). Then, if $V^{\mathrm{T}}v_0$ has a non-zero component for the highest eigenvalue eigenvector, it would be selected by $V\Lambda^n$ in $V\Lambda^n V^{\mathrm{T}}v_0$. Q.E.D.

Although both $A^{\mathrm{T}}A$ and $A^{\mathrm{T}}WA$ are symmetric and positive semidefinite their product might not be symmetric. Nevertheless we can, at least theoretically, factor $A^{\mathrm{T}}A$ as $V\,\mathrm{diag}(\lambda)V^{\mathrm{T}}$, where $\lambda_k \geq 0$. As we assumed in the beginning that $A^{\mathrm{T}}A$ is
42 ◁ regular we even have $\lambda_k > 0$. Now, defining[4] regular matrix $R := V\,\mathrm{diag}(\lambda n.\sqrt{\lambda_n})$, we can write $A^{\mathrm{T}}A$ as RR^{T}. Multiplying (407) from the left by R^{T} and defining $y := R^{\mathrm{T}}x$ we obtain

$$R^{-1}A^{\mathrm{T}}WAR^{-\mathrm{T}}\,y = \lambda y \tag{410}$$

which is an ordinary eigenproblem with symmetric matrix $M := R^{-1}A^{\mathrm{T}}WAR^{-\mathrm{T}}$. The symmetry implies that all its eigenvalues are real[9]. Moreover they are all in the interval $[0,1]$ because (410) is equivalent with (404) for which it was shown. Therefore M is positive semidefinite and we can use 6.5.14 to find the solution. n-th iteration (before normalization) reads as

$$\begin{aligned}\vec{x}_n &= R^{-\mathrm{T}}M^nR^{\mathrm{T}}\vec{x}_0 = R^{-\mathrm{T}}R^{-1}A^{\mathrm{T}}WAR^{-\mathrm{T}}M^{n-1}R^{\mathrm{T}}\vec{x}_0\\ &= (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}WA\left(R^{-\mathrm{T}}M^{n-1}R^{\mathrm{T}}\vec{x}_0\right) = \left((A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}WA\right)^n\vec{x}_0\end{aligned} \tag{411}$$

This shows that even the direct iteration (that does not need R) is convergent. In fact, all that complications with R were done only to prove that it is so. One more nuisance must be dealt with. Often $A^{\mathrm{T}}A$ is singular and the above theory would fail. The remedy is in taking $A^{\mathrm{T}}A + \delta I$, where $\delta > 0$ is small number, instead of it. As $A^{\mathrm{T}}A$ was symmetric positive semidefinite, adding a small diagonal to it makes it positive definite and therefore regular[10]. It even speeds up the convergence of computing $(A^{\mathrm{T}}A)^{-1}$ by
193 ◁ (398). However, it must not be exaggerated – too large δ leads to wrong solution (usually the focusing filter comes out as very narrowband, turning the restored signal into incomprehensible whistling). Note that the presence of δ changed the energy conservation condition into

$$||\,\mathrm{circ}_N(a)x||^2 + \delta||x||^2 = ||a||^2 \tag{412}$$

Values of up to about 2 % of $(A^{\mathrm{T}}A)_{00}$ seem to work well on real sound signals.

6.5.15 Blind Focusing Algorithm

Let me summarize the blind focusing algorithm here. Its inputs are $a\in\mathbb{R}^M$ — the signal, $w\in[0,1]^M$ — the weighing vector, $N < M$ — the size of the output, $\alpha > 0$ — the relative regularization factor (value of 0.02 works well), $\beta\in[0,1]$ — the Richardson extrapolation strength (0.4 seems to work fine) and $x\in\mathbb{R}^N$, $x \neq \vec{0}$ — the initial guess.

[9] For matrix $A = A^{\mathrm{H}}$, we have $\lambda = \overline{\lambda}$ for its every eigenvalue because $x^{\mathrm{H}}x\lambda = x^{\mathrm{H}}Ax = x^{\mathrm{H}}A^{\mathrm{H}}x = (Ax)^{\mathrm{H}}\,x = (\lambda x)^{\mathrm{H}}x = \overline{\lambda}x^{\mathrm{H}}x$. Therefore every λ is real.

[10] Symmetric positive semidefinite matrix M can be written as $V\,\mathrm{diag}(\lambda)V^{\mathrm{T}}$, where V are orthonormal matrices of its eigenvectors and $\lambda_k \geq 0$. Obviously $M + \delta I = V\,\mathrm{diag}(\lambda)V^{\mathrm{T}} + \delta VV^{\mathrm{T}} = V(\mathrm{diag}(\lambda) + \delta I)V^{\mathrm{T}}$, which makes all its eigenvalues positive, thus $M + \delta I$ is regular.

It produces output $x\in\mathbb{R}^N$ such that $\mathrm{circ}_N(a)x$ would represent the focused signal. The first $N-1$ places of w should be always zero, so that not yet fully started convolution $a * x$ would not bias the results. For the same reason, the initial $N-1$ samples long segment of the focused signal $\mathrm{circ}_N(a)x$ should be discarded because of wrap-around. The algorithm proceeds as follows.

$$\begin{aligned}A &:= \mathrm{circ}_N\begin{pmatrix}O_{(N-1)\times 1}\\ a\end{pmatrix}\\ W &:= \mathrm{diag}\begin{pmatrix}O_{(N-1)\times 1}\\ w\end{pmatrix}\\ B &:= A^{\mathrm{T}}A\\ \delta &:= \alpha B_{00}\\ y &:= x\end{aligned} \tag{413}$$

repeat until convergence:

$$\begin{aligned}\star\qquad & \text{get } z \text{ by solving } (B+\delta I)z = A^{\mathrm{T}}WA\left(x+\beta(x-y)\right) \text{ using (398)}\\ & y := x\\ & x := \frac{z}{||z||}\end{aligned}$$

6.5.16 Note The method depends on the fact that all eigenvalues lie in $[0,1]$. By taking $B + \delta I$ instead of B these may change and we have to check the impact of this. Writing (404) with δ-regularization, multiplied by x^{T} from the left, we get

$$x^{\mathrm{T}}A^{\mathrm{T}}WA\,x = \lambda x^{\mathrm{T}}A^{\mathrm{T}}Ax + \lambda\delta x^{\mathrm{T}}x \tag{414}$$

Which implies

$$\lambda = \frac{x^{\mathrm{T}}A^{\mathrm{T}}WA\,x}{x^{\mathrm{T}}A^{\mathrm{T}}A\,x + \delta x^{\mathrm{T}}x} \tag{415}$$

The denominator is even larger than in (406), which means that $\lambda\in[0,1]$. Thus it is safe to use δ-regularization in (413).

6.5.17 Dual Blind Focusing

As already mentioned, the Lagrange multiplier method captures the maximum as well as minimum of (402). It means that we can tackle the focusing task from the opposite end as well. Instead of marking regions in time whereto the energy of $a * x$ should flow, we mark the regions that we want to avoid. This can be achieved by searching for the eigenvector belonging to the least eigenvalue.

All eigenvalues of $M := (B+\delta I)^{-1}A^{\mathrm{T}}WA$ are between 0 and 1 by (415). So are all eigenvalues of $I - M$ with the distinction that the least eigenvalue of M got transformed into the greatest eigenvalue of $I - M$, which enables us to adopt the algorithm (413) for finding the least eigenvalue eigenvector. All that has to be done is inserting the following line right after the $\star$-labeled line.

$$z := \left(x+\beta(x-y)\right) - z \tag{416}$$

This make z equal to $(I-M)\big(x+\beta(x-y)\big) = \big(I-(B+\delta I)^{-1}A^{\mathrm{T}}WA\big)\big(x+\beta(x-y)\big)$, which guides (413) towards the least eigenvalue eigenvector.

Note that the dual approach for $\delta = 0$ is formally equivalent to the original method used with $I - W$ because

$$\left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}(I-W)A = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}A - \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}WA = I - \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}WA \tag{417}$$

But as W is required to have its first $2N-2$ places zero, the two methods are in fact different.

Practically, the dual method seems promising as it does not need exact positions of glottal peaks in W, getting by with marked regions of silence. These are easier to extract from the reverberated sound. Unfortunately, I found that the method converged much more slowly than the primal algorithm 6.5.15 – probably the eigenvalues of (404) are gathered near 0, slowing down convergence. So will be subject of future research, namely if there is a meaningful way of combining primal and dual methods. The current system uses primal method only.

196 ◁

6.5.18 Limitations

Both the noise canceling and focusing assumes linearity (which usually holds in case of rooms and modern recording equipment), constant sampling frequency (noise canceling additionally requires input to be sampled synchronously with the output[11]) and that the impulse response does not change over time. The last requirement is the most problematic. If anything moves in the room the impulse response changes. Slight changes may be tolerated but as these accumulate over time we cannot use very long input signals a. The input has to be cut in segments of reasonably constant reverb. But this effectively limits the length of reconstruction response x as this has to be at least 10 times shorter that signal a.

In fact, I witnessed noticeable overtraining of noise canceling even if the response was 20 times shorter than the signal. The impulse response $\widehat{x}$ sort of learned how to create not only $a*x$ but also an approximation of y thru interference with a. So in the result $y - a*\widehat{x}$, a part of the y signal was missing. I could tell that by listening to $a*\widehat{x}$ in which traits of y could be heard from time to time.

Another limiting factor is that many sounds do not come directly from the microphone but from audio files, such as `mp3`. Due to compression, these damage fine structure of reverberations by non-linear and time-variant distortion, which further limits effectiveness of the method.

All these factors limit practical length[12] of x to about 0.1–1 s, depending on sound quality. For longer reverberations different method, such as time masking, has to be used. Nonetheless even the masking profits from partial sharpening that the blind focusing is able to provide.

[11] For that reason, noise canceling would fail if the input was recorded with physically distinct sound card than the output was generated. Each of the cards has its own oscillator and these are likely not to run at precisely same frequency.

[12] Note that x is a FIR filter, and therefore it undoes IIR smearing, which can have much longer effect the the length of the FIR, depending on closeness of its poles to the unit circle. Also note that this IIR is causal minimum phase filter (section 4.7).

101 ◁

6.6 Driving the Focuser

This section answers wherefrom we obtain the weighing vector w.

6.6.1 Pulse Shape

Suppose we knew instants of glottal closures. How should we shape w so that the blind focusing would concentrate the energy of the signal around these instants? Obviously, w should be composed of pulses, each having its maximum at the instant of glottal closure.

But as we can expect that the there will always be errors in determining exact closure instants we also need sufficiently forgiving pulse shape for which these small errors would not lead to a dramatic change of the outcome.

I will provide heuristic argument, why the shape of the pulse should be a falling exponential, by investigating what happens with two glottal pulses when weighing function w gets shifted for the second one by small Δt (shifting w as a whole does not harm at all — as long as x is long enough it just shifts is in the same direction to compensate). As (402) in fact maximizes (406) we can define the following cost function

▷ 195 ▷ 196

$$\varphi(x,\Delta t) := \frac{||(a*x)\odot w(\Delta t)||^2}{||a*x||^2} \tag{418}$$

where $w(\Delta t) := s + s[-T_0-\Delta t]$ is a sum of two instances of the shape s we are looking for. The first one is centered at zero, while the second at T_0 with the ability to move by Δt. Now we want a shape that preserves the following property

$$\varphi(x,0) < \varphi(z,0) \;\Rightarrow\; \varphi(x,\Delta t) < \varphi(z,\Delta t) \tag{419}$$

for every reasonably small value of Δt. This would ensure that the best x remains stable despite small perturbations. Let us require even more, namely

$$\frac{\varphi(x,0)}{\varphi(z,0)} = \frac{\varphi(x,\Delta t)}{\varphi(z,\Delta t)} \tag{420}$$

Expanding the right side, writing α for the left side, we get

$$\alpha||(a*z)\odot(s+s[-T_0-\Delta t])||^2 = ||(a*x)\odot(s+s[-T_0-\Delta t])||^2 \tag{421}$$

Rearranging and assuming that the two copies of s do not interfere (being zero everywhere except close vicinity of the glottal pulse), we get

$$\alpha||(a*z)\odot s||^2+\alpha||(a*z)\odot s[-T_0-\Delta t]||^2 = ||(a*x)\odot s||^2+||(a*x)\odot s[-T_0-\Delta t]||^2 \tag{422}$$

Now assuming that $s_n = \beta^{-n}$ for a short interval after zero, being zero elsewhere and that x and z are already so close to the solution that the signals $a*x$ and $a*z$ are practically zero except certain interval after the glottal pulse we can write $(a*z)\odot s[-T_0-\Delta t]$ as $(a*z)\odot s[-T_0]\beta^{\Delta t}$. Using the following definition

$$\begin{aligned} A_1 &:= ||(a*x)\odot s||^2 & A_2 &:= ||(a*x)\odot s[-T_0]||^2 \\ B_1 &:= ||(a*z)\odot s||^2 & B_2 &:= ||(a*z)\odot s[-T_0]||^2 \end{aligned} \tag{423}$$

(422) leads into

$$\alpha B_1 + \alpha B_2 \beta^{2\Delta t} = A_1 + A_2 \beta^{2\Delta t} \tag{424}$$

or

$$\alpha B_2 \left(\frac{B_1}{B_2} + \beta^{2\Delta t} \right) = A_2 \left(\frac{A_1}{A_2} + \beta^{2\Delta t} \right) \tag{425}$$

The last formula holds for $\Delta t = 0$ as it is equivalent to (420), under the assumption of locality of s, $a * x$ and $a * z$. Moreover let us assume that the first pulse $a * x$ is very similar to the second one $(a * x)[-T_0]$, and likewise for z. Then $A_1/A_2 \approx B_1/B_2$ and both sides of (425) could be approximately canceled regardless of value of Δt, provided that it was reasonably small. This finishes heuristic argument why I use decaying exponentials in the weighing vector w.

Few practical things remain to be said. Firstly, the focusing performance was found to be nearly independent of the rate of fall β. It is currently set to decay to $1/32$ of the starting value after 10 ms. Moreover the whole w is delayed by 1 ms. This is to accommodate for errors in glottal pulse timing. As the response x is finite it has $x_n = 0$ for all $n < 0$. If there would be systematic error in glottal pulse tracking it could mean that the optimal solution would like to use negative indices in x. Shifting w by 1 ms creates 1 ms safety margin at the beginning of x for that.

6.6.2 Finding the Pulses

The pulses are found by the following procedure. First, the simplified implementation of the YIN pitch tracker [13] is run on the input to assign estimated F_0 frequency to every 10 ms block of the signal. These blocks are then examined to remove values based on too weak or too loud (clipped) signal. These block will be marked as unvoiced. This is also done with the consecutive blocks whose F_0 changes only by ± 1 when expressed as a period in number of samples (at 48 kHz sampling rate). No human is able to produce such stable F_0 but it can 'appear' in the signal due to strong resonant echo. If it was used to create w it would baffle the focuser completely — that is why it has to be removed. Then, the outliers are removed by interpolation with their neighbors. After that, blocks with $F_0 > 640$ Hz are marked as unvoiced. Finally, contiguous regions are identified and shortened by 10 ms on both ends because it is believed that upon the start and at the end the signal to noise ratio is worse than in the middle.

97 ◁ The estimated glottal periods are then used in selection where to place glottal closure instants as follows. First the signal is preemphasized by (232). Then the absolute value is taken of its samples and it is processed by a floating thresholding that works the following way. If the current threshold t_n is greater than current signal x_n it falls exponentially as $t_{n+1} := \alpha t_n$, where α is set to decay to $1/2$ over the period time supplied by the pitch tracker. Whenever $x_n > t_n$, the t_n is set to x_n and output spike is produced as $z_n := x_n$ (at other instants the output z_n is zero). At the same time all outputs z_m for $m < n$ that lie below a line that crosses a zero at 40 % of the period duration (but at least 1 ms) are reset to 0. This removes initial oscillation retaining only the main peak.

Then a checking phase is run that searches for glottal periods with more then one non-zero z_k. If such a period is found the it gets square-rooted. As the analyzing window slides by 10 ms steps, it may mean for long periods that a single value of z_k gets square-rooted several times in a row. This measure has to limit the impact of errors in F_0 tracking.

In the next step, non-zero values of z get changed to their logarithms, so that the pulses with less amplitude (and hence worse SNR) would be taken proportionally to their information content.

Finally the exponential decays are added after the non-zero z_k's to create w, as described in previous section and whole w is delayed by 1 ms.

As can be seen, this method is rather ugly and should be replaced by something theoretically grounded. Certainly, there is a room for improvement and further research. For the time being it should be regarded as a baseline procedure which nevertheless works as will be demonstrated in the following section.

6.6.3 Demonstration

Usually, the initial estimate of glottal closures tends to be very unreliable. Consequently, the improvement is often hardly visible at first. To overcome this, the focusing has to be performed in several stages using the result from the preceding stage for creating better w to perform new focusing on the original signal (i.e. not the one from the previous stage — this is only used for better estimate of glottal closures). After repeating this 5 or 6 times, the improvement can be clearly seen (and heard).

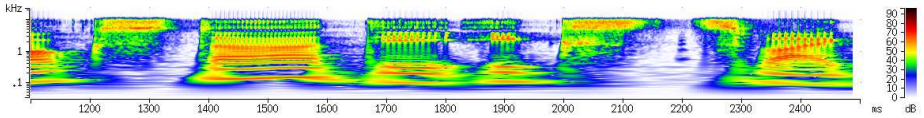


Fig. 54. *Original signal. It is actually longer than what fits in here, lasting almost 30 seconds. These 30 seconds are used in the focusing as a single block without any further subdivision.*

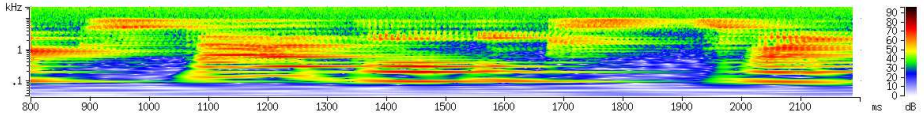


Fig. 55. *The same signal artificially distorted by simulated echo and additive Gaussian noise with $\sigma = 120$ (the samples are regarded as 16 bit signed integers). Formant movements became unclear.*

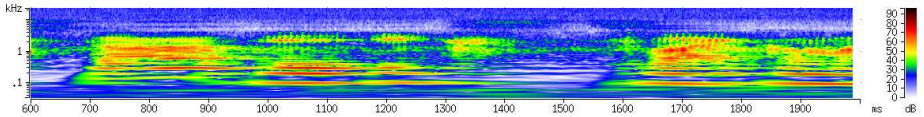


Fig. 56. *Focused with $N = 4096$, $\alpha = 10^{-6}$ and $\beta = 0.4$ by 4 stages, totally using 352 iteration of (413) each using about 300 iterations of (398) on average.* ▷ 198 ▷ 193

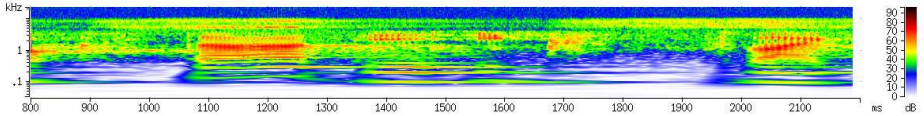
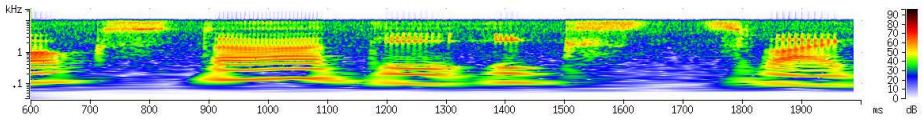


Fig. 57. *Upper bound on performance of the focuser. The glottal pulse positions were obtained from the clean signal. It took 35 iterations of (413), each involving about 245 iterations of (398) with regularization factor $\alpha = 0.02$, Richardson extrapolation factor $\beta = 0.4$ and the length of the filter $N = 16384$ samples.*



192 ◁ 194 ◁ 193 ◁ ***Fig. 58.*** *Absolute limit on performance of any linear method that minimizes mean square error. The original signal was used as y in (392) to plot $a * x$. For $N = 16384$, it took 29 iterations of (401), each using 420 iterations of (398) on average.*

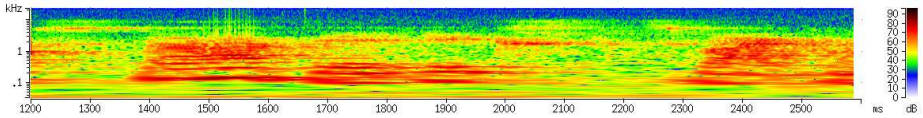


Fig. 59. *This is how the sound re-recored with omnidirectional microphone looks like. The reverb and noise completely took over the formant structure.*

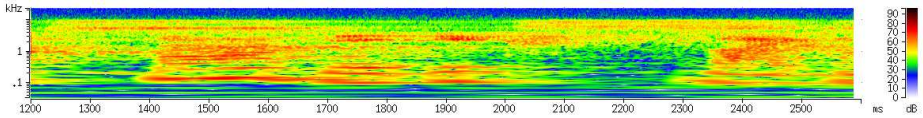


Fig. 60. *Focused by 6 stages, N=16384.*

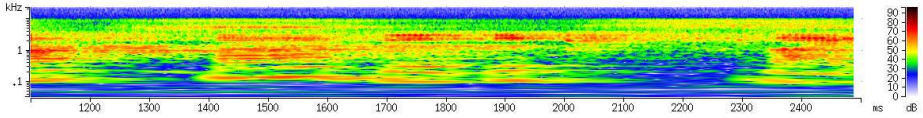


Fig. 61. *Upper bound on performance of the focuser, $N = 16384$, $\alpha = 10^{-6}$, $\beta = 0.4$.*

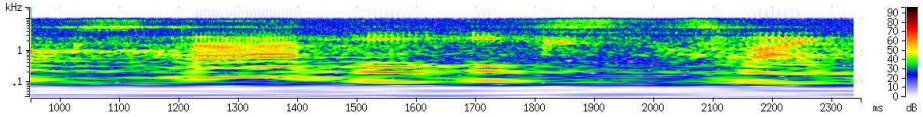


Fig. 62. *Absolute limit on any mean square error method. For $N = 16384$ it took 28 iterations of (401), each requiring 200 iterations of (398) on average.*

The following spectrograms show the original clean recording, artificially distorted version thereof and finally the same recording aired to the room and recorded again together with heavy reverb and background hum of the computer.

It can be seen that the method improves readability of quick formant glides, whereas the quality of fricatives seems to get worsened. Probably there should be different mode of w-shaping when dealing with fricatives. Unfortunately, this (apart from low speed) currently limits practical usefulness of this method.

6.7 Noise Following

Since learning of all possible combinations of all noises and all reverberations and all utterances is intractable, we need a way to treat missing information when only separate learning of signals and channel distortions would be used. Ideally the channel distortion should be learned during the recognition phase so that it would quickly adapt to any channel.

Current solution is still far from this ideal. It just tracks minimum of each NUFIBA channel 800 ms to the past and anytime the signal comes closer than 3 dB to this limit,

it is marked as noise. This is usually only effective against constant periodic noises, such as 50 Hz mains buzz.

6.8 Smoothing

Before the spectrogram can be used, we have to remove effects of F_0 from the it first. This is achieved by tracking F_0 and running cosine window along each channel, tuned according to current F_0 such that the it would act as a notch filter for F_0. Moreover the averaging is being done before the logarithm is taken from the NUFIBA output. For that reason it works as if it performed soft-max function in dB scale, effectively removing glottal pulses from higher formants. Towards lower frequencies the method also tracks F_0 harmonics, linearly interpolating the space between them, down to the second harmonics (the first one is discarded).

This procedure transforms the original pulsed spectrogram into smooth image of moving formants. Note that F_0 value is retained sideways as it is needed to tell voiced/unvoiced phones apart and in tonal languages its value even determines the phone.

6.9 MMI Criterion of the Output Alphabet

Let $W \xrightarrow{\mathcal{C}_\mathcal{A}} A \xrightarrow{\mathcal{D}_\Lambda} \widehat{W}$ be a Markov chain of random variables (see (103) and 3.2.35 to refresh what the Markov chain is). The chain represents the channel over which the words are being sent (for sake of simplicity we are assuming these to be independent). The random variable W represents the source of the words. The first arrow indicates the channel, which involves not only the propagation of sound in the air and its recording but also entire action of the NUFIBA front end. That is why the channel is parametrized by the NUFIBA output alphabet $\mathcal{A}$. Random variable A represents the sequence of symbols from $\mathcal{A}$ that was generated by the given input word. The channel $\mathcal{D}_\Lambda$ is the speech recognizer which translates A into $\widehat{W}$. ▷ 55 ▷ 56

Now we want to make a selection of $\mathcal{A}$ such that the probability of error would be minimal. From Fano's inequality 3.3.11 we get that the probability of the word error is bounded from below by ▷ 63

$$P_e \geq \frac{H(W) - H(P_e) - I(W;A)}{\log_2 \#W} \geq \frac{H(W) - 1 - I(W;A)}{\log_2 \#W} \tag{426}$$

Therefore, it seems reasonable to choose the alphabet $\mathcal{A}$ so as to maximize $I(W;A)$. Of course it does not say anything about the maximum probability that the error occurs. It does not guarantee better performance, it merely removes obstacles for it.

6.9.1 Vector Quantizer

Due to the lack of time, the MMI criterion was not used and ordinary K-means vector quantizer was used instead.

The letters of the acoustic alphabet are created by clustering the vectors made of 20 consecutive slices of the spectrogram. The windows moves by 10 slices in runtime.

This leads to 75 acoustic letters per second. Each output letter of $\mathcal{A}$ is generated by standard k-means clustering with Euclidean distance:

$$\rho(A, B) = \sum_{ij} |A_{ij} - B_{ij}|^2 \tag{427}$$

where A and B are slabs of 20 consecutive frames. The pitch information that was kept aside is added as an extra coordinate after that.

6.10 My Contribution

The things presented in this chapter are my work, this chapter being central to this thesis. Although the filterbank approach is not really a new idea, its combination with Hilbert transforms, echo suppression, noise floor tracking and discrete output alphabet creates a genuine combination that has not been tried yet, as far as I know.

Moreover the method of blind focusing via generalized eigenvectors is all my invention, probably new to signal processing.

Finally, although the MMI criterion for acoustic alphabet selection has been already published in [54], it was derived there by low level analysis of neural network they used as a classifier and it depended on certain plausible assumptions.

My explanation, which uses Fano's inequality is fully general, independent of the classifier type and other special assumptions.

7 Acoustic Model

The acoustic model is based on a believe that more important things in speech happen when the sound is changing than when it is steady. That change, being sort of a gesture of our articulation organs. And gestures needs movement, even if we just wave our hand. For this reason, I do not model individual phonemes but rather transitions from one phoneme to another. I call these transitions *diphones*. So, each model in fig. 63 represents one such transition. The steady parts of phonemes are mainly captured by the first loop of each model and for that reason, that loop may be shared among those that transit from the same phone. This loop is intended to correspond to the middle a phone. The other four states model the transition itself. Each diphone model's end is connected to every diphone beginning by a null arrow. These represent "language model", which is equivalent to 4-gram (4-phone) model.

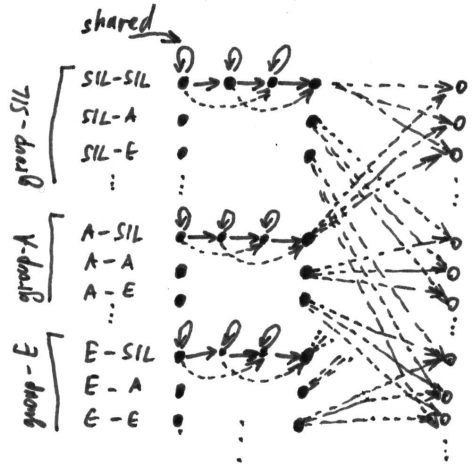


Fig. 63. *Diphone acoustic model. Filled circles represent states, each outline circle represent the corresponding state on the left (as if it was drawn on a cylinder). Each transition from one phoneme to another (including silence) has its own HMM. HMMs whose beginning belongs to identical phone are grouped together, sharing the loop in the first state by tying. Each solid arrow represents all symbols of the acoustic alphabet, each of them having attached the probability of traversing that arrow and emitting that symbol. Dashed arrows represent null transitions (with assigned probability of this happening). The sum of the probabilities of all letters on all arrows emanating from a single state plus the sum of probabilities of null arrows emanating from this state is one.*

8 MMI Clustering and Class-Based Language Model

This chapter closesly follows my article [44]. This is also the only part of my thesis that has been implemented so cleanly that I dared to release[1] it under the GPL licence as a language processing tool.

8.1 MMI Classes

Now a classical method of data clustering, called *Maximum Mutual Information Clustering* was introduced in [47] in a context of language modeling. The original article contained some cues concerning its implementation. These are carried out in detail here, together with some new tricks. The results of the test run on 110M words long Czech National Corpus are briefly described then.

The background idea of Maximum Mutual Information Clustering is in an intuition that a given word is more 'interchangeable' with some words than with the others. Therefore it should be possible to define word classes, lumping together the words which often appear in similar contexts. Then, we could work with these classes instead of words in estimation of n-gram probabilities. As there will be fewer classes than words, we can expect the probability estimates to be more reliable than those that are
19 ◁ using words directly. Recall formula (5).

8.1.1 The Model

More formally, let us assume, that we are given the set V of possible words, together with a joint probability $P_0(v_k, v_{k-1})$, meaning how probable it is to encounter a fixed pair of consecutive words (v_k, v_{k-1}) in typical input text $\langle v_0 \dots v_M \rangle$. Then, for sake of simplicity, we define probability $P(w_0 \dots w_N)$ of a word sequence $\langle w_0 \dots w_N \rangle$ to be

$$P(w_0) \cdot P(w_1, \dots, w_N | w_0) := P(w_0) \cdot \prod_{k=1}^{N} P_1(w_k | C(w_k)) \cdot P_2(C(w_k) | C(w_{k-1})) \quad (428)$$

where C is the class function $C : V \to \mathcal{P}(V)$ satisfying $C(x) \cap C(y) = \emptyset$ for any x, y such that $C(x) \neq C(y)$, and $\bigcup_{x \in V} C(x) = V$. Probabilities concerning the set of words are defined as one would expect:

$$\begin{aligned} P_c(A, B) &:= \sum_{a \in A,\, b \in B} P_0(a, b) \quad \text{for sets } A, B \subseteq V \\ P_1(x \mid C(x)) &:= \frac{P_c(\{x\}, V)}{P_c(C(x), V)} \qquad P_2(C(x) \mid C(y)) := \frac{P_c(C(x), C(y))}{P_c(V, C(y))} \end{aligned} \quad (429)$$

[1] `http://ufal.mff.cuni.cz/tools.html/mmi.html`

Our goal is to select function C so as to maximize $P(w_0 \dots w_N)$ for some heldout data $\langle w_0 \dots w_N \rangle$. This is equivalent with maximizing logarithm of $P(w_1 \dots w_N \mid w_0)$, which leads to

$$\begin{aligned} L(C) &:= \sum_{k=1}^{N} \Big(\log_2 P_1(w_k \mid C(w_k)) + \log_2 P_2(C(w_k) \mid C(w_{k-1})) \Big) = \\ &= -\left(-\sum_{k=1}^{N} \log_2 P_c(\{w_k\}, V) \right) + \sum_{k=1}^{N} \log_2 \frac{P_c(C(w_k), C(w_{k-1}))}{P_c(C(w_k), V) \cdot P_c(V, C(w_{k-1}))} \\ &= N \cdot \Big(I\big(C(w_0), \dots, C(w_{N-1}) \,;\, C(w_1), \dots, C(w_N)\big) - H\big(w_1, \dots, w_N\big) \Big) \end{aligned} \quad (430)$$

where

$$I\big(C(w_0) \dots C(w_{N-1}); C(w_1) \dots C(w_N)\big) = \frac{1}{N} \sum_{k=1}^{N} \log_2 \frac{P_c(C(w_k), C(w_{k-1}))}{P_c(C(w_k), V) \cdot P_c(V, C(w_{k-1}))} \quad (431)$$

is cross mutual information and $H\big(w_1 \dots w_N\big) = -\frac{1}{N} \sum_{k=1}^{N} \log_2 P_c(\{w_k\}, V)$ is cross entropy. Now it can be clearly seen where the name for the method came from. All we have to do is to find the mapping C maximizing mutual information of some training data (where we have estimated P_0) versus some (different) heldout data, and the result will be one that maximizes probability of heldout data in the framework of our model.

But, unfortunately, I am not aware of any reasonably fast algorithm, achieving this. It is clear that we have to back slightly off the optimality requirement for sake of practical feasibility. The next section explains how this can be done.

8.2 MMI clustering method

The MMI clustering method was introduced in [47] and is also described in [39] The key idea is to use the same data for P_0 estimation as well as C selection. This greatly simplifies algebra and also alleviates the need for probability smoothing required by the original formulation (which was needed there so that we would not get $\log_2(0)$ somewhere on the heldout data). On the other hand it is not very natural solution and has to be commented, at least: Working on the single data set means that the optimal classes become the singleton classes, one for each word. But this is not what we want. Here, the second idea takes place: Instead of maximizing $L(C)$ on the heldout data, we will try to keep $L(C)$ as high as possible for the preselected number of classes, putting aside the question of how do we discover the right number of them. Still it is too difficult to be done on a computer except for very small input. So we back-off from the optimality even more and instead of trying to find the right classes, we will use eager solution working in a bottom-up way, building a forest of classes (eventually ending with one big classification tree).

Suppose we have the input text $\vec{w} = \langle w_0 \dots w_N \rangle$ and define co-occurrence matrix c_{yx} in the following way:

$$c_{yx} := \#\{k \in [1, N] \cap \mathbb{N} \mid w_k = y,\ w_{k-1} = x\} \quad (432)$$

Then we assign $P_0(y,x)$ to be c_{yx}/N. For a fixed input text $\vec{w}$ and fixed class mapping $C : V \to \mathcal{P}(V)$ we define I to be N-times the mutual information, i.e. $N \cdot I\big(C(w_0)\dots C(w_{N-1})\,;C(w_1)\dots C(w_N)\big)$, which can be expanded as follows.

$$I := N \cdot I(\dots;\dots) = \sum_{X,Y \in \mathrm{Rng}(C)} \left(\sum_{\substack{x \in X \\ y \in Y}} c_{yx} \right) \log_2 \frac{P_c(Y,X)}{P_c(Y,V) \cdot P_c(V,X)} \tag{433}$$

Note that we treat $0 \log_2 0$ as 0 here, to make (433) equivalent to the original cross-entropy formulation (with P_0 estimated on the same data). We will only need special case of (433), for C that maps each word to its singleton class, i.e. $C(v) = \{v\}$. This gives the following formula.

$$I := \sum_{x,y \in V} c_{yx} \log_2 \frac{N \cdot c_{yx}}{c_{y\bullet} \cdot c_{\bullet x}} \tag{434}$$

8.2.1 Basic algorithm

The following pseudocode describes how a simple clustering can be done. Its input is considered to be the matrix c_{yx} set up by (432) and indexed by words $w \in V$. For sake of simplicity we treat the words as numbers from 1 to[2] A (the resulting classes will then have numbers from $A+1$ on). Function `I(c)` is defined by formula (434).

Note that not all the words are being classified. This is because rarely appearing words (less than `T`-times, `T` being around 10) are too sparse to be classified reliably. Nevertheless, they are still in c_{yx} matrix to help the classification of other words. For sake of brevity, the algorithm does not build the tree, it only prints the history of merges as the tree can be easily reconstructed from it.

```
set(int) active={ y | (sum a : (c[y,a]+c[a,y])) >= 2*T }
for(n=A+1; #active>1; n++){
  {l,r}=argmax {y,x}, x in active, y in active : I(merge(c,y,x,n))
  c=merge(c,l,r,n);   active \= {l,r};   active U= {n}
  output("merging %d and %d into %d", l,r,n)
}
```

where `merge()` is

```
merge([M,M]int c, int k, int l, int n):[M,M]int
{
  for x in M \ {k,l} {
    c[n,x]=c[k,x]+c[l,x]    /* set M is assumed to be large enough */
    c[x,n]=c[x,k]+c[x,l]    /* to hold all the new classes */
  }
  c[n,n]=sum (a,b) in {k,l}*{k,l} : c[a,b]
  for a in M { c[k,a]=c[l,a]=c[a,k]=c[a,l]=0 }
  return c
}
```

[2] Unlike in the rest of this book, this matrix does not get indexed from 0. It stems from the implementation which uses zero index for special purposes. Although it could be easily reindexed here I did not want to introduce formal discrepancy between the algorithm and its implementation as this that could possibly lead to erros later, when someone would try to change the program.

8.3 Optimizations

The above algorithm has a complexity of $\Omega(C^5)$ where $C =$ `#active` is the number of words being classified. Optimizations are therefore necessary. The first step is to minimize the loss of $I()$ occurring after the merge instead of maximizing total $I()$. Although it seems to be a minor modification at the first sight, it turns out that the $I_{loss}()$ can be precomputed into an array and only slightly changed upon each merge, leading to a considerable speedup.

$$\begin{aligned} I_{loss}(c,l,r) := I(c) - I(\texttt{merge}(c,l,r,n)) = \sum_{x,y \in V} c_{yx} \log_2 \frac{N \cdot c_{yx}}{c_{y\bullet} \cdot c_{\bullet x}} \\ -\Bigg(\sum_{\substack{x \in V \setminus \{r,l\} \\ y \in V \setminus \{r,l\}}} c_{yx} \log_2 \frac{N \cdot c_{yx}}{c_{y\bullet} \cdot c_{\bullet x}} + \sum_{x \in V \setminus \{r,l\}} (c_{lx} + c_{rx}) \log_2 \frac{N(c_{lx} + c_{rx})}{(c_{l\bullet} + c_{r\bullet}) \cdot c_{\bullet x}} + \\ + \sum_{y \in V \setminus \{r,l\}} (c_{yl} + c_{yr}) \log_2 \frac{N(c_{yl} + c_{yr})}{c_{y\bullet} \cdot (c_{\bullet l} + c_{\bullet r})} + \\ + (c_{ll} + c_{lr} + c_{rl} + c_{rr}) \log_2 \frac{N \cdot (c_{ll} + c_{lr} + c_{rl} + c_{rr})}{(c_{l\bullet} + c_{r\bullet}) \cdot (c_{\bullet l} + c_{\bullet r})} \Bigg) \end{aligned} \tag{435}$$

After rather technical manipulations this can be simplified into

$$\begin{aligned} Q(c_{l\bullet}, c_{r\bullet}) + Q(c_{\bullet l}, c_{\bullet r}) + Q(c_{ll}, c_{lr}) + Q(c_{rl}, c_{rr}) - Q(c_{ll} + c_{rl}, c_{lr} + c_{rr}) \\ - \sum_{y \in J_Y} Q(c_{yl}, c_{yr}) - \sum_{x \in J_X} Q(c_{lx}, c_{rx}) \end{aligned} \tag{436}$$

where $Q(a,b) := R(a+b) - R(a) - R(b)$ and $R(x) := x \log_2(x)$ for $x > 0$ and $R(0) := 0$. The sets J_Y and J_X can be any supersets of $\{y \mid c_{yl} \cdot c_{yr} \neq 0\}$ and $\{x \mid c_{lx} \cdot c_{rx} \neq 0\}$, respectively[3]. Note that once we precompute $c_{\bullet x}$ and $c_{y\bullet}$ into suitable arrays all the terms in the formula except the last two, can be evaluated in constant time. This already has a complexity of $\mathcal{O}(A)$ for one evaluation of I_{loss} (where the naive implementation took $\mathcal{O}(A^2)$). Moreover, the sets J_X and J_Y over which we are summing usually have much lower cardinality then A, leading to yet more improvement.

The algorithm can now precompute I_{loss} into an array for all $C(C-1)/2$ pairs using formula (436) (this amounts to $\mathcal{O}(AC^2)$ operations), and then it can start iterations as before but instead of recomputing I every time, it would simply select the pair with minimal I_{loss}. Let this pair be (l,r). Then it would $\texttt{merge}(c,l,r,n)$ and compute I_{loss} of the new class n with all other classes (search for minimal I_{loss} takes $\mathcal{O}(C^2)$, merging takes $\mathcal{O}(A)$, and all new I_{loss} values require $\mathcal{O}(AC)$ — doing it C times leads to complexity of $\mathcal{O}(AC^2)$).

Last thing that must be done is the correction of all other I_{loss} values. It must be done since as l and r classes no longer exist (they were merged into a new class n), the value of $I_{loss}(c,a,b)$ might have changed. Let the result of $\texttt{merge}(c,l,r,n)$ be denoted

[3] This freedom is caused by the fact that $Q(0,x) = 0$ and $Q(x,y) = Q(y,x)$

by $\widehat{c}$. Now we are about to compute what correction to $I_{loss}(c,a,b)$ has to be added to make it $I_{loss}(\widehat{c},a,b)$:

$$\begin{aligned} I_{correction}(c,l,r,a,b) &:= I_{loss}(\widehat{c},a,b) - I_{loss}(c,a,b) = \\ Q(\widehat{c}_{a\bullet},\widehat{c}_{b\bullet}) + Q(\widehat{c}_{\bullet a},\widehat{c}_{\bullet b}) + Q(\widehat{c}_{aa},\widehat{c}_{ab}) + Q(\widehat{c}_{ba},\widehat{c}_{bb}) - Q(\widehat{c}_{aa}+\widehat{c}_{ba},\widehat{c}_{ab}+\widehat{c}_{bb}) \\ -\sum_{y\in V\cup\{n\}} Q(\widehat{c}_{ya},\widehat{c}_{yb}) - \sum_{x\in V\cup\{n\}} Q(\widehat{c}_{ax},\widehat{c}_{bx}) - \Big(Q(c_{a\bullet},c_{b\bullet}) + Q(c_{\bullet a},c_{\bullet b}) + Q(c_{aa},c_{ab}) \\ +Q(c_{ba},c_{bb}) - Q(c_{aa}+c_{ba},c_{ab}+c_{bb}) - \sum_{y\in V} Q(c_{ya},c_{yb}) - \sum_{x\in V} Q(c_{ax},c_{bx})\Big) \end{aligned} \tag{437}$$

Since $\{a,b\}\cap\{l,r\}=\emptyset$, many terms cancel out (see the definition of `merge()`), leading to

$$\begin{aligned} I_{correction}(c,l,r,a,b) &= \sum_{y\in V} Q(c_{ya},c_{yb}) + \sum_{x\in V} Q(c_{ax},c_{bx}) - \sum_{y\in V\cup\{n\}} Q(\widehat{c}_{ya},\widehat{c}_{yb}) \\ &- \sum_{x\in V\cup\{n\}} Q(\widehat{c}_{ax},\widehat{c}_{bx}) = Q(c_{la},c_{lb}) + Q(c_{ra},c_{rb}) + Q(c_{al},c_{bl}) + Q(c_{ar},c_{br}) \\ &-Q(\widehat{c}_{na},\widehat{c}_{nb}) - Q(\widehat{c}_{an},\widehat{c}_{bn}) = Q(c_{la},c_{lb}) + Q(c_{ra},c_{rb}) + Q(c_{al},c_{bl}) + \\ &+Q(c_{ar},c_{br}) - Q(c_{la}+c_{ra},c_{lb}+c_{rb}) - Q(c_{al}+c_{ar},c_{bl}+c_{br}) \\ &= U(c_{la},c_{ra},c_{lb},c_{rb}) + U(c_{al},c_{ar},c_{bl},c_{br}) \end{aligned} \tag{438}$$

where

$$U(a,b,c,d) := Q(a,c) + Q(b,d) - Q(a+b,c+d) \tag{439}$$

$I_{correction}(c,l,r,a,b)$ has to be added to every pair of $I_{loss}[a,b]$ array (a,b such that $\{a,b\}\cap\{l,r\}=\emptyset$) just before merging l with r. This amounts to $\mathcal{O}(C^2)$ operations per iteration. Combining it with complexity estimates already done we have that the total complexity of the algorithm just sketched is $\mathcal{O}(AC^2)$.

8.4 Implementation tricks

To further cut down the execution time (although the worst case complexity measured by means of A and C stays the same[4]) certain tricks are needed. As already hinted, the input data is preprocessed such that numbers are substituted for words. The second idea concerns the matrix c_{yx}. It is typically quite sparse, so it worths to implement it via a hash table.

8.4.1 Hash table

Note that it has to be a special table, since we need to be able to walk thru columns and rows as well as to delete elements no longer needed after the merge. Memory demands also impose some constraints on the form the hashing table should have. For instance, implementing the row/column walking ability via a linked list would be quite expensive, considering that on a 64 bit machine each pointer occupies 8 bytes and we would probably need two of them. The solution, I've chosen, is using an array (indexed by y) of hashing tables indexed by x. So for each line of the matrix we have an extra hash table. Their entries contain only the key x (32 bit integer) and the counter (32 bit integer) holding the value of c_{yx}. Collisions are resolved by double hashing. Thus, each bigram theoretically occupies 8 bytes of the memory. In practice it is more, since for the hashing to be fast we have keep say 60% of the table unused[5]. According to test runs, this leads to an average collision rate of less than 2 collisions per access[6] which is acceptable. Note that construction of such a table is a three phase process. In the first phase we read all the input data and count unigram frequencies. These frequencies are used as upper bounds for row sizes of the table. In the second phase, we allocate such a table and fill it with bigrams. The only purpose of this table is to count the true number of bigrams appearing in the respective rows. Finally this table is deleted and the right-sized table is build which has its row sizes selected such that they will be filled at 40%, unless they have less than two elements — in such cases[7] they are allocated tight, since there would be no speedup from an empty space, anyway.

8.4.2 Loss Computation and Merging

For I_{loss} computation as well as for merging we need to walk thru rows and columns. Walking along the row is done simply by reading valid entries of the hash table (60% items read are unused but it is acceptable). To walk thru a column, we need special array, one for each column holding y-indexes of entries in that column. The walk is then performed by look-up of y-indexes followed by search in the y-th hash for the key x. Therefore, it is slower than a row-walk.

For I_{loss}, it would be nice to walk only over the intersection of sets of indexes of non-zero entries of the respective rows/columns. But keeping track of the all $\mathcal{O}(C^2)$ intersections would be very expensive. Therefore we only keep the track of number of non-zero items stored in a given row/column and select the shorter one for walking. While it is walked the corresponding data from the other column are being looked-up in the hash.

Merging l with r (where $l<r$) is done such that it reuses index l for the new class (instead of introducing new index n as in the basic algorithm). As we still want the new index in the output, we need to maintain a translation array. Note that on each merge, the number of items in rows and columns may decrease, so after the number of items stored in any given hash table falls below say 1/5 of the original filling, the whole table is rehashed to be smaller (this makes row-walks faster and it is also taking the advantage of CPU's caches, so the extra work pays off). Note that when merging

[4] In fact, it will even be worse than that, if we consider that the worst case behavior of hashing is worse than $\mathcal{O}(1)$. In the following I will simply ignore that possibility since it is very unlikely to happen, thus not affecting the average performance.

[5] I know that it is not very good performance and that hashing functions exist, behaving well down to 30% of unused slots of the table. But those are more complicated, one method for instance requires size of the table to be a prime twin. Definitely there is a room for improvement. But there are other things to improve that would result in much noticeable speedup, so this does not worth to be changed, now.

[6] This means that to store/retrieve single item to/from the hash, the array (representing the hash) has to be accessed less than 3 times on average.

[7] These typically occupy half of the rows of the table due to the Zipf's Law.

the rows, the number of entries of the resulting row l can generally increase. It is implemented such that a new (optimal sized) hash is created and the source hashes are unallocated after the data has been moved.

8.4.3 Loss-Correction Computation

Symmetries of $U()$ can be used to spare some evaluations of it. It is easy to see, that

$$\begin{aligned}
&U(a,b,c,d) = U(c,d,a,b) = U(b,a,d,c) = U(d,b,c,a) = U(a,c,b,d) = U(b,c,d,a)\\
&U(0,b,c,d) = Q(b,d) - Q(b,c+d) = R(c+d) + R(b+d) - R(d) - R(b+c+d)\\
&U(0,0,c,d) = 0\\
&U(0,b,c,0) = -Q(b,c)
\end{aligned} \tag{440}$$

Corrections from $U(c_{la}, c_{ra}, c_{lb}, c_{rb})$ are processed separately from $U(c_{al}, c_{ar}, c_{bl}, c_{br})$ since the first traverses rows while the other columns of c_{yx} (a and b is changing, l, r will be merged). To make it fast, we first gather for $x \in$ `active` those pairs (c_{lx}, c_{rx}) having at least one of their member non-zero; we can also precompute $Q(c_{lx}, c_{rx})$. This way we get the set X containing those x-es appearing in the non-zero pairs. Then we compute the value of $U()$ for each $a < b$, $a \in X$, $b \in X$. We are using identities (440) to suppress evaluation of $U()$ in cases when

$$(c_{la} = 0 \text{ and } c_{lb} = 0) \quad \text{or} \quad (c_{ra} = 0 \text{ and } c_{rb} = 0) \tag{441}$$

In cases where $c_{la} = 0$ or $c_{ra} = 0$ simplified formula is used.

This trick proved itself to be crucial for the speed. The same program without it can run 4 days while it can only take 30 minutes with it (observed on part (1M words) of the Czech National Corpus (number of unigrams $A = 120187$, number of bigrams $B = 592188$, and the number of words being classified was $C = 10612$)). This optimization was not mentioned in [47] nor in [39].

8.4.4 Miscellaneous

There are also some auxiliary arrays there. As already noted, we have an array of arrays used to walk thru columns of the main table. We also have an array of lengths of those arrays. Then, there are mapping arrays translating internal numbering (which originates due to index re-usage) into external numbering and another one which translates it into numbering used by triangular I_{loss} matrix. Used by I_{loss} computations, there are arrays of row and column sums $c_{y\bullet}$, $c_{\bullet x}$ as well as $R(c_{y\bullet})$, $R(c_{\bullet x})$ which saves some evaluations of the $\log_2$ function. Also note that all of the column-walking-arrays may need an update when merging two rows. As this would be too expensive, lazy implementation is used which require some bookkeeping (I will not describe it here since it is marginal and quite lengthy).

8.5 Tests

These early experiments were only meant to test the performance of the program. First, 1.2M words of collected Shakespeare's work were processed in 6 minutes (on 2.4 GHz machine, using single CPU), using less than 200 MB of RAM, yielding to 2.8k words classified. Next, larger input of 120M words of the Czech National Corpus were processed. It took 14 hours and 2 GB of RAM (on the same machine) to classify 10k most common words. Below, some selected parts of the tree are presented:

At the first sight they look quite convincing, but occasionally we can see weird classes (like `naději/pozor` or `bankovnictví/plavání`). I suppose that this is caused by classes which have very small (maybe even empty) sets J_X and J_Y in formula (436). Then, especially for rarely occurring words, I_{loss} becomes very small although the words being eventually merged have little in common. As the loss is very small such pairs are likely to be formed early as the program runs. They may form misleading classes which further spoil classification of other words. ▷ 210

This effect, hugely amplified, can be observed on a short data (having, say, 20k words of length) where we set `T` to be 1. Although for `T` high enough (such that there are few words to classify, say $C = 70$) the classes are acceptable, if we try to classify all the words, the result looks completely arbitrary. Even the words that were classified sufficiently well when C was 70 are wrong now with respect to one another.

8.6 My Contribution

In this section I described how the MMI method can be efficiently implemented. Most of the tricks were already mentioned in [47]. However, slightly different and more compact formulas were found — symmetric formula (436) for I_{loss} and formulation of $I_{correction}$ using U-function (439). Their impact is twofold. At first, analysis of symmetries (440) leads to further savings in computation time, since it turns out that many results are zero (what was not directly visible in the formulas of the original paper). Secondly, they yield to deeper insight into the numbers according to which the classes are selected. It

210 ◁ was noticed that from (436) it follows that the criterion which eliminates rarely used words from classification should take the size of sets J_X and J_Y into the account, not only the total number of occurrences of a given word.

9 Experiments

Due to the lack of time I resorted to carry out only the basic experiment. It tries to estimate the performance of the phoneme recognizer on the Switchboard-1 corpus. Note that this simple recognizer does not benefit from the MMI class language model as its 'language model' concerns histories of individual phonemes.

9.1 Switchboard-1 Corpus

Despite being outdated, I chose this corpus as I was well familiar with it and it had a force-aligned phoneme stream suitable for training. I am well aware that relatively clean but band limited telephone speech will not enable the NUFIBA to show its strengths. But time waits for nobody.

The Switchboard-1 telephone corpus consists of 2438 spontaneous telephone conversations between two speakers. For each conversation, two speakers were carefully selected from a group of them such that the selection would maintain 'uniform' distribution of different types of voices across the corpus as well as speakers' gender. The conversations take from 50 seconds to 10 minutes, being 6 minutes 22 seconds on average.

Audio files are presented in so called Sphere files (having `.sph` extension) which compose of an ASCII header with various information about the speakers, file size, file compression format and so on, followed by a binary data usually in μ-law non-uniform quantization sampled at 8 kHz. Each side of the conversation has its own channel, so that it can be resolved what they said even if talking over each other, which nevertheless was quite rare. Unfortunately, the separation is not complete and there is an audible cross-talk from the other channel. It differs from recording to recording and perhaps even within single recording because adaptive canceling of section 6.5 turned out to ▷ 190 be ineffective to suppress it (but it may as well be caused by non-linear quantization effects or by a combination of both).

In its totality the corpus represents 259 hours[1] of speech during which 3 008 999 words[2] were uttered out of the vocabulary of 26 357.

Together with the raw audio data, there is a 'true transcript' available, made by human annotators. On top of that, these transcripts were force-aligned[3] with the audio as words and even as phonemes (so that if there are several possible pronunciation of some word, the force alignment gives us the most likely one on each occurrence of that word in the corpus).

[1] Or 518 hours of audio but as one speaker usually listens to the other, half of the files is actually a silence.

[2] In fact, some files were damaged and/or not transcribed at all, therefore the true number of words might be slightly higher.

[3] Force-alignment means that the recognizer is run in a special mode in which it is given the correct answer. Its only task is to find most likely path thru the models, thereby giving us times where each word (or even phoneme) have most likely occurred. This could be done by connecting all the HMMs of the correct words one after the other and running the Viterbi search. Consult [39] for more details.

Apart from force aligned true transcripts, there is also an output of the recognizer (unfortunately the documentation does not indicate what was used as a training data for it). Comparing these two transcripts, we obtain the following tables by running HTK `HResults` program which computes WER_1 error rate. Note that each conversation side was taken as a 'sentence' here, and that is why the sentence error rate is so high. The other reason is that the data come from year 2002. Todays state of the art recognizer would surely do better.

```
,-------------------------------------------------------------.
| HTK Results Analysis at Tue Jul 10 00:03:45 2012            |
| Ref: swb1_true.mlf                                          |
| Rec: swb1_asr.mlf                                           |
|=============================================================|
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-------------------------------------------------------------|
| Sum/Avg | 4856  |  78.45  14.75   6.79   6.25  27.80 100.00 |
`-------------------------------------------------------------'
```

So the WER is nearly 28 % and the phoneme error rate is the following:

```
,-------------------------------------------------------------.
| HTK Results Analysis at Tue Jul 10 00:17:07 2012            |
| Ref: swb1p_true.mlf                                         |
| Rec: swb1p_asr.mlf                                          |
|=============================================================|
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-------------------------------------------------------------|
| Sum/Avg | 4856  |  85.95   7.40   6.66   6.30  20.35 100.00 |
`-------------------------------------------------------------'
```

It is interesting that the error rate of ASR phoneme recognition (performed via a force-alignment to the words recognized) is about 20 %, which is close to the current performance of phoneme recognizers [49] working directly. As it is likely that these make different kinds of errors, the actual phone error rate achievable with current technology would be even lower, provided that these two were combined.

9.2 Experimental Setup

The experiment involves training and testing phoneme recognizer with NUFIBA front end and acoustic model of fig. 63. Originally it ran full NUFIBA including its blind focuser but as it soon become clear that it is too slow to meet the deadline, the focuser was disabled and the experiment rerun from the beginning. The impact of this should not be too high as most Switchboard recordings suffer little echo.

206 ◁

The corpus was randomly split into two halves, one intended for training, the other for testing. The split was performed on coversations not on conversation halves, which means that both communication directions of a single recording go either to the training part or to the testing part.

Due to problems with the crosstalk, the data from the other conversation side is used to identify true silence. Whenever there is no speech in both channels at the same time, it is considered a silence. This prevents the model for silence to be trained with attenuated speech[4] due to crosstalk, which would reducing the performance of the resulting engine. There may be noise or even non-speech sounds there during the silence — this is an intended behavior as the model for silence has to capture all non-speech sounds. Only then it will be helpful in separating speech and non-speech. Note that due to the way the corpus was split, using the data from the other conversation side does not lead to mixing of training data with the test data.

9.2.1 Training

First the acoustic alphabet is created by k-means clustering, as described earlier.

Then the initial training is performed using the Baum-Welsh algorithm, diphone by diphone taken from force aligned human transcripts. This procedure only trains the diphone models but not the null arrows that form a "language model". These are estimated separately from the transcripts, using Kneser-Ney smoothing.

Then the second phase of training procceds as follows. The data is chunked into consecutive word pairs[5] and for each of them the sequence of phonemes from the transcript is laid down as a sequence of diphones, their HMMs get connected by the null arrows (analogically to fig. 7) and, again, the Baum-Welsh is used to train the recognizer on these chunks. Note that the silence is trained on selected regions free of crosstalk, as explained earlier.

▷ 30

9.2.2 Model Size and Memory Requirements

There are 46 phones in the Switchboard corpus, including 3 different types types of silence. Making all pairs this yields 2116 diphone HMMs. Real count is smaller because only the HMMs corresponding to diphones actually occuring in the training data were retained.

Each diphone HMM consists of 4 states with totally 6 outputting transitions out of which 1 is shared among a group of diphones starting by the same phone. Each outputting transition should be thought of as a bundle of $\#\mathcal{A}$ outputting arrows, one for each letter of the acoustic alphabet $\mathcal{A}$. This requires $\#\mathcal{A}\cdot 46\cdot(1+46\cdot 5) = 10626\cdot\#\mathcal{A}$ parameters.

Then, there is a bigram language model comprised of $2116^2 = 4477456$ null transitions. Additionally there are two null transition per diphone HMM, making up for a total of 4481688 null transitions.

For $\#\mathcal{A} = 500$ it gives $5313000 + 4481688 = 9794688$ parameters, which is 37.3 MB if `float` numbers are used to represent these values.

9.2.3 Testing

Unfortunately, the training is still running as of now. That is why I cannot provide any results, yet. As soon as these will be avaiable I am going to publish them at `http://195.113.26.193/~klusacek/asr.html`.

[4] Though sometimes clearly visible in spectrograms.

[5] The two-word window moves by one word each step.

10 Conclusions

I feel the most embarrassed that I did not manage to finish this work in time, despite the effort. For this reason, the thesis cannot be ended by any clearcut conclusions. At least not before the experiment, which is still running, finishes.

Anyway, even in the absence of happy end, the thesis has its bright moments. Namely:

(a) It explains why it worth to invest an effort to the development of better front end instead of a language model. See discussion in section 6.1. Naturally, the credibility of these arguments depends on the result of the experiment.

(b) There are two theoretical results there. The first is called response decomposition theorem 4.9 and it is a formula that allows to separate zeroes and poles inside the unit circle from those that lie outside, without the need of root finding. There still nevertheless remains a lot of work to be done before it could be used numerically. The second theoretical result concerns blind focusing 6.5, which is a novel blind deconvolution technique intended to make fast formant glides visible even in reverberated recordings. This method is already usable but should be further optimized as it is very slow. It has been even finally excluded from current experiment, for its low speed (nevertheless, it was not enough or happened too late to meet the deadline).

(c) There is one practical result, the language processing tool for MMI clustering, released under GPL. See chapter 8.

(d) In section 5.8 there is a summary of what is currently known about the human hearing system. Unlike most physiological literature, it is written in technical terms, having speech recognition in mind.

(e) Finally I tried to make this book self-contained preffereably using only elementary mathematics so that it could be used as an introductory text into speech recognition or basic signal processing.

For me, the lesson learned from this endeavor is that building the speech recognizer from scratch is more demanding, then I imagined at the beginning. I would incline to say it may be even hard but perhaps I am just getting old or fed up with the subject.

11 Acknowledgments

I would like to express my gratitude to all who helped me with this work, one way or the other. Unfortunately it is impossible to name them all here. So I will try not to forget at least some of them, in what follows.

First, I would like to thank my advisor for his generosity, letting me explore wichever direction in my research I liked to. My second thank is for Libuše Brdičková, the secretary of our department, for shielding me from bureaucracy. Third, I reckon endless patience of Prof. Jarmila Panevová, waiting for me to submit the thesis. My thanks also go to Prof. MUDr. Josef Syka, DrSc., dr.h.c. and Dr. Ing. Daniel Šuta from the Institute of Experimental Medicine for consultations concerning section 5.8.

Then, I want to thank doc. Zdeněk Žabokrtský, Ph.D. for thorough proofreading, Nino Peterek, Ph.D. for helping me with running the HTK `HResults` program and proofreading, Mgr. Petr Kaňka for proofreading and Filip Uhlík, Ph.D. for proofreading and stimulating discussions.

Finally, I must not forget my aunt Ing. Libuše Ramešová who took high resolution photograph of the university logo — the one at the book cover — and Mgr. Petr Gadas who edited it into the final form.

My thanks, not connected only with this work, go to numerous authors of free software namely the `GNU/Linux` system and `gcc` compiler, `gnuplot` plotting program and TEX typesetting system. Without them, the work would have looked much differenty or it would not exist at all. Thank you gentlemen.

And, above all, I want to thank my family especially my mother for encouraging me to “finish this damn work already”.

12 List of Symbols

Herein follows the list of symbols that are considered more or less global in this book. It is intended as a quick reference for the reader wanting to refresh the definition of a particular symbol.

	V	Usually a set of words. The vocabulary.
	V^k	The k-tuple of words from V (or items from general set V).
	ε	The empty string, $\{\varepsilon\} = V^0$.
	V^*	The set of all strings over V, that is $\bigcup_{k=1}^{\infty} V^k$.
	V^+	The set of all non-empty strings over V, i.e. $V^+ = V^* \setminus \{\varepsilon\}$.
	$\#V$	Cardinality (the number of elements) of set V.
	$\vec{x}$ or x	Finite-dimensional vector x over $\mathbb{C}$ indexed from 0, understood as $1 \times N$ column matrix. The arrow above x gets usually dropped in cases where there is no danger of confusion. x_n denotes the n-th element of $\vec{x}$, whereas $\vec{x}_n$ is the n-th vector in the list. See 4.1.
71 ◁		
27 ◁	$\#\vec{x}$	Dimension of vector $\vec{x}$, see footnote[13].
	$\langle a_0, a_1, \ldots \rangle$	Column vector or infinite sequence made of the elements in the list.
	$\overline{x}$ or $\overline{A}$	Complex conjugation of number or vector x or matrix A.
	A^{T}	Transpose of matrix A, see section 4.1.
	A^{H}	Hermite conjugate of matrix A, $A^{\mathrm{H}} := \overline{A}^{\mathrm{T}}$. See 4.1.
	$x^{\mathrm{H}}y$	Scalar product of complex column vectors x and y, see 4.1.
	$\|\|x\|\|$	2-norm of vector x defined as $\|\|x\|\| := \sqrt{x^{\mathrm{H}}x}$, see 4.1.
	I or I_N	Identity matrix $(N \times N)$.
	O or O_N	All-zero matrix $(N \times N)$.
191 ◁	$\mathrm{diag}(\vec{v})$	Diagonal matrix constructor, see 6.5.4.
190 ◁	$\mathrm{circ}(\vec{v})$	Circulant matrix constructor, see 6.5.1.
191 ◁	$\mathrm{toep}(\vec{v})$	Toeplitz matrix constructor, see 6.5.2.
41 ◁	$\mathcal{P}(A)$	The set of all subsets of the set A. See footnote[2].
	$\bigcup A$	Union of all elements (taken as sets) of the set A. See footnote[2].
40 ◁	$f[A]$	Image of set A, see 3.1.5.
	$f^{-1}[A]$	Preimage of set A, see 3.1.5.
	$\mathrm{Rng}(f)$	Range of function f, see 3.1.5.
	$\mathrm{Dom}(f)$	Domain of function f, see 3.1.5.
42 ◁	$\lambda x.T$	Lambda abstraction, discussed in footnote[4].
35 ◁	$a\,?\,b : c$	See footnote[20].
	$\mathbb{N}$ or $\mathbb{N}_0$	The set of all natural numbers, including 0.
	$\mathbb{N}_1$	Natural numbers starting by 1 (i.e. $0 \notin \mathbb{N}_1$).
	$\mathbb{N}^k$	The set of all k-tuples of natural numbers.
	$\mathbb{Z}$	The set of all integers.
	$\mathbb{Z}_k$	The set of integers modulo $k \in \mathbb{N}_1$, i.e. $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$.
	$\oplus$	Exclusive or, as a function $\mathbb{Z}_2 \times \mathbb{Z}_2 \to \mathbb{Z}_2$.
	$\mathbb{R}$	Real numbers.
	$\mathbb{C}$	Complex numbers, i denotes the imaginary unit.
	$\lfloor x \rfloor$	For $x \in \mathbb{R}$, the $\lfloor x \rfloor$ is the highest integer less than or equal to x.
	$a \,\%\, b$	Modulo operation[2]: $a \,\%\, b := a - b \cdot \lfloor a/b \rfloor$ for any $a \in \mathbb{Z}$, $b \in \mathbb{N}_1$. ▷ 73
	$\circ$	Function composition $A \circ B := \lambda x.A(B(x))$. See footnote[6]. ▷ 43
	$\bullet$	Einstein's summing notation. For a vector v the expression $v_\bullet$ is a shortcut for $\sum_{i=0}^{\#v-1} v_i$. Likewise, for a matrix A the $A_{i\bullet}$ stands for the sum of the i-th row. See footnote[9] and section 8.2. ▷ 19 ▷ 208
	$\mathcal{O}(g)$	$f \in \mathcal{O}(g)$ iff $\exists x_0, c \in \mathbb{N}_1 : \forall x > x_0 : f(x) \le cg(x)$
	$\Omega(g)$	$f \in \Omega(g)$ iff $\exists x_0, c \in \mathbb{N}_1 : \forall x > x_0 : f(x) \ge cg(x)$
	$\mathrm{Arg}(c)$	Angular part of the complex number, $\mathrm{Arg}(Ae^{i\varphi}) = \varphi$ for $\varphi \in (-\pi, \pi]$.
	Arg_C	See footnote[26]. ▷ 113
	$\mathrm{sinc}(x)$	See 4.10.4. ▷ 117
	$\mathrm{sgn}(x)$	Sign function. $\mathrm{sgn}(x) := (x > 0\,?\,1 : (x < 0\,?\,-1 : 0))$.
	$\mathrm{sgn}_{\mathcal{S}}(x)$	Modified (1-periodic) sign function, see footnote[36]. ▷ 125
	$\mathrm{atg}(x)$	Arctangent, $\mathrm{atg} : \mathbb{R} \to (-\pi/2, \pi/2)$.
	$\mathrm{atg2}(y, x)$	Arctangent of two variables, $\mathrm{atg2} : \mathbb{R} \times \mathbb{R} \to (-\pi, \pi]$. It is defined as follows: $\mathrm{atg2}(y, x) := \mathrm{Arg}(x + iy)$.
	$\mathcal{T}_s^z(\vec{a})$	The set of all paths in the trellis compatible with the observation $\vec{a}$. See formula (11). ▷ 23
	$\Pr_\Omega$	Probability distribution on probability space Ω, as defined in 3.1.1. ▷ 39
	$\mathcal{E}(A)$	Expectation of random variable A, see 3.1.9. ▷ 42
	$\mathrm{Var}(A)$	Variance of random variable A, see 4.12.1. ▷ 127
	$\sigma(A)$	Standard deviation of A, see 4.12.1.
	$\perp\!\!\!\perp$	Conditional independence relation (72). ▷ 44
	$H_\Omega(X)$	Entropy of random variable X, defined in 3.2.1. ▷ 46
	$H(P\|\|Q)$	Cross Entropy, see 3.2.16. ▷ 53
	$D(P\|\|Q)$	Relative Entropy also known as the Divergence, see 3.2.17. ▷ 53
	$I(X;Y)$	Mutual information, see 3.2.19. ▷ 54
	$X \to Y \to Z$	Markov chain, see (103). ▷ 55
	F or F_N	$N \times N$ matrix of discrete Fourier transform, see 4.1.1. ▷ 72
	f^+, f^-, $f^\pm$	Rectified function $f : \mathbb{R} \to \mathbb{C}$, see 4.2.2. ▷ 77
	σ	The set of all signals, $\sigma \subset (\mathbb{Z} \to \mathbb{C})$. See 4.2.11. ▷ 80
	σ_F	The set of all finite signals $\sigma_F \subset \sigma$.
	σ_I	The set of all invertible signals $\sigma_I \subset \sigma$. See 4.3.9. ▷ 86
	σ_D	Set of impulse responses achievable by a digital filter. See 4.9. ▷ 112
	$\mathcal{S}$, $\mathcal{S}_F$, $\mathcal{S}_I$, $\mathcal{S}_D$	Sets of spectra of signals living in sets σ, σ_F, σ_I and σ_D, respectively.
	$\mathcal{F}$	Fourier series $\mathcal{F} : \sigma \to \mathcal{S}$. See 4.2.15. ▷ 81
	$\vec{0}$	Zero signal $\vec{0} := \lambda n.0$ or all zero vector. See 4.2.16. ▷ 81
	$\vec{1}$	Unit impulse signal $\vec{1} := \lambda n.(n = 0\,?\,1 : 0)$. See 4.2.16.
	$*$	Convolution, see 4.3.1 and (180) for its spectral counterpart. ▷ 83 ▷ 84
	$\odot$	Pointwise multiplication, see 4.3.3 and also 4.1.7. ▷ 84 ▷ 74
	$a[n]$	Time shift $a[k]_n = a_{k+n}$, see 4.3.4. ▷ 85
	ρa	Time reversal of signal a, see 4.3.5 ▷ 85
	$\overline{a}$	Complex conjugate of signal a, see 4.3.6 ▷ 85
	$\mathcal{I}$, $\mathcal{X}$	Direct and crossed phase-minimizing conversion. See 4.9.
	$\mathcal{D}_k$	Decimation operator $\mathcal{D}_k : \sigma \to \sigma$, where $k \in \mathbb{N}_1$, see 4.10.2. ▷ 116
	$\mathcal{Q}$	Quadrature filter $\mathcal{Q} : \sigma \to \sigma$, see 4.11. ▷ 124
	$\mathcal{H}$	Hilbert transform $\mathcal{H} : \sigma \to \sigma$, see (309). ▷ 125

13 References

[1] http://cochlea.org/.

[2] http://diwww.epfl.ch/lami/team/vschaik/eap/am.html.

[3] http://en.wikipedia.org/wiki/Auditory_pathways.

[4] http://en.wikipedia.org/wiki/Broca%27s_area.

[5] http://en.wikipedia.org/wiki/Complementary_distribution.

[6] http://en.wikipedia.org/wiki/Stevens_power_law.

[7] http://en.wikiquote.org/wiki/Fred_Jelinek.

[8] http://heavens-above.com/solar-escape.asp?/.

[9] http://www.youtube.com/watch?v=JJ_zlj4pFV0.

[10] http://www.youtube.com/watch?v=KyLqUf4cdwc
http://www.youtube.com/watch?v=YozgvRvjCtI.

[11] *TM Synchronization and Channel Coding — Summary of Concept and Rationale.* The Consultative Committee for Space Data Systems, June 2006.

[12] CRAY-1 COMPUTER SYSTEM — Hardware Reference Manual 2240004, Nov. 4, 1977.

[13] Alain de Cheveigne, Hideki Kawahara. YIN, a Fundamental Frequency Estimator for Speech and Music. *Acoustical Society of America*, pages 1917–1930, 2002.

[14] Jont. B. Allen. How Do Humans Process and Recognize Speech? In *IEEE Transactions on Speech and Audio Processing*, volume 2, pages 567–577, No.4, October 1994.

[15] Andrew J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, Vol. IT-13:260–269, April 1967.

[16] Barry Arons. A Review of the Cocktail Party Effect. Technical report, MIT Media Lab.

[17] F. Baumgarte. A Computationally Efficient Cochlear Filter Bank for Perceptual Audio Coding. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 5:3265–3268, 2001.

[18] Behzad Akbarpour, Sofiene Tahar. Error analysis of digital filters using HOL theorem proving. *Journal of Applied Logic*, 5:651–666, January 2007.

[19] C. E. Shannon. A Mathematical Theory of Computation. *The Bell System Technical Journal*, Vol. 27:379–423, 623–656, July and October 1948.

[20] Ciprian Chelba and Frederick Jelinek. Structured Language modeling for Speech recognition.

[21] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2005.

[22] D. R. R. Smith, R. D. Patterson, R. Turner. The Processing and Perception of Size Information in Speech Sounds. *Acoustical Society of America*, pages 305–318, 2005.

[23] David Marr. A Theory of Cerebellar Cortex. *Journal of Physiology*, 202:437–470, 1969.

[24] Teuvo Kohonen et al. Status Report of the Finnish Phonetic Typewriter Project. In *ICANN-91, Espoo, Finland*, pages 771 – 776, June 24-28 1991.

[25] F. Hayes-Roth, V.R. Lessen. Focus of Attention in the HEARSAY-II System. In *IJACIS*, 1977.

[26] Hugo Fastl and Eberhard Zwicker. The Ear's Own Nonlinear Distortion.

[27] J. T. Foote. Discrete MMI Probability Models for HMM Speech Recognition.

[28] R.W. Hamming. *Digital Filters.* Prentice-Hall, New Jersey, 1977.

[29] H. Heřmanský H.Bourlard and N. Morgan. Towards Increasing Speech Recognition Error Rates. 1996.

[30] H. Heřmanský. Perceptual Linear Predictive Analysis of Speech. In *Audio Signal Processing in Humans and Machines, Lecture 17*, 1995.

[31] Hynek Heřmanský and Sangita Sharma. Temporal Patterns (TRAPS) in ASR of Noisy Speech. In *Proc. ICASSP*, pages 289–292, 1999.

[32] M. Kössl I. J. Russell. Micromechanical Responses to Tones in the Auditory Fovea of the Greater Mustached Bat's Cochlea. In *Journal of Neurophysiology*, pages 676–686, 1999.

[33] J. Matoušek, J. Nešetřil. *Kapitoly z diskrétní matematiky.* KAM Series 95-299, 1995.

[34] J. R. Bellegarda. Exploiting Latent Semantic Information in Statistical Language Modeling. *Proceedings of the IEEE*, August 2000.

[35] D. Willett J. Rottland, Ch. Neukirchen. Performance of Hybrid MMI-connectionist / HMM Systems on the WSJ Speech Database 1.

[36] J. Syka, L. Voldřich, F. Vrabec. *Fyziologie a patofyziologie zraku a sluchu.* Avicentrum, 1981.

[37] Jaroslav Hašek, vypráví Jan Werich. *Osudy dobrého vojáka Švejka.* Supraphon, 1967.

[38] E.T. Jaynes. *Probability Theory the Logic of Science.* Cambridge University Press, 2003.

[39] F. Jelínek. *Statistical Methods for Speech Recognition.* The MIT Press, 1997.

[40] Jonathan Richard Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain.* School of Computer Science, Carnegie Mellon University, Pittsburgh, August 1994.

[41] Joseph Santos-Sacchi . `http://www.yaleearlab.org/intranet/images/besame.wmv`.

[42] David Klusáček. Pronunciation Modelling in Speaker Detection. Technical report, Charles University in Prague, Faculty of Mathematics and Physics, ÚFAL, 2003.

[43] David Klusáček. Optimal Detection in Case of Sparse Training Data. In *Proceedings of ODYSSEY04* , pages 97–104, 2004.

[44] David Klusáček. Maximum Mutual Information and Word Classes. In *WDS'06 Proceedings of Contributed Papers*, pages 185–190. MatfyzPress, Charles University, MFF UK, Trója, Prague, 2006.

[45] L. J. Wang, A. Kuzmich, A. Dogariu. Gain-Assisted Superluminal Light Propagation. *Nature*, Vol. 406:277–279, July 2000.

[46] L. R. Bahl, P. F. Brown, P. V. de Souza, R. L. Mercer. Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. In *ICASSP 86, TOKYO*, pages 49–52, 1986.

[47] Robert L. Mercer. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18:467–479, December 1992.

[48] C.M.Bishop M.Svensén. The Generative Topografic Mapping. In *Neural Computation 10*, volume 1, pages 215 — 234, 1998.

[49] P.Matějka P.Schwars and J. Černocký. Towards Lower Error Rates in Phoneme Recognition.

[50] J. Psutka. *Komunikace s počítačem mluvenou řečí.* Academia, 1993.

[51] R. P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[52] Dabbala Rajagopal Reddy. *An Approach to Computer Speech Recognition by Direct Analysis of the Speech Wave.* PhD thesis, Stanford, CA, USA, 1966.

[53] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker Verification using Adapted Gaussian Mixture Models. *Digital Signal Processing*, pages 19 — 41, 2000.

[54] G. Rigoll and C. Neukirchen. A New Approach to Hybrid HMM/ANN Speech Recognition Using Mutual Information Neural Networks. In *Advances in Neural Information Processing Systems 9, NIPS*96*, pages 772–778. The MIT Press, 1996.

[55] S. F. Chen, J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling, 1999.

[56] S. Kanthak S. Molau and H. Ney. Efficient Vocal Tract Normalization in Automatic Speech Recognition.

[57] N. Fakotakis T. Ganchev and G. Kokkinakis. Comparative Evaluation of Various MFCC Implementations on the Speaker Verification Task.

[58] T. Gold, R. J. Pumphrey, I. Hearing. The Cochlea as a Frequency Analyzer. In *Proc. R. Soc. Lond. B Biol. Sci.*, pages 462–491, 1948.

[59] T. M. Cover, J. A. Thomas. *Elements of Information Theory.* A John Wiley & Sons, 2006.

[60] Fabio Tamburini. Automatic Prosodic Prominence Detection in Speech using Acoustic Features: an Unsupervised System. In *EUROSPEECH, Geneva*, 2003.

[61] J. H. L. Hansen U. H. Yapanel and Satya Dharanipragada. A New Perceptually Motivated MVDR-Based Acoustic Front-End (PMVDR) for Robust ASR. 2004.

[62] V. Goel and W. J. Byrne. Minimum Bayes Risk Automatic Speech Recognition. *Computer Speech and Language*, pages 115–135, 2000.

[63] David Verstraeten, Benjamin Schrauwen, and Dirk Stroob. Isolated Word Recognition Using a Liquid State Machine.

[64] Georg von Békésy. Concerning the Pleasures of Observing, and the Mechanics of the Inner Ear. *Nobel Lecture*, December 11, 1961.

[65] W. T. Vetterling W. H. Press, S. A. Teukolsky and B. P. Flannery. *Numerical Recipes in C.* Cambridge university Press, 1992.

[66] Y. Bengio, R. E. Ducharme and Pascal Vincent and Departement d'Informatique Et Recherche Operationnelle and Centre De Recherche Mathematiques. A Neural Probabilistic Language Model. In *Journal of Machine Learning Research*, pages 1137–1155. MIT Press, 2001.

[67] Y. Normandin. *Hidden Markov Models, Maximum Mutual Information and the Speech Recognition Problem.* Ph.D. thesis, McGill University, Montreal, 1991.

[68] U. H. Yapanel and J. H. L. Hansen. Towards an Intelligent Acoustic Front-End for Automatic Speech Recognition: Built-in Speaker Normalization (BISN). 2004.

David Klusáček

New Methods in Statistical Speech Recognition

Version 1.0
July 2012

Illustrations by David Klusáček
Cover graphics by David Klusáček
Cover photograph by Libuše Ramešová

This book was typeset in TEX typesetting system using `cseplain` init file. It is typeset in Czech modification of BaKoMa free fonts, a Postscript Type 1 implementation of Computer Modern and $\mathcal{AMS}$-TEX fonts. Bookbinding by

Knihařství Rak
Dittrichova 7
120 00 Praha 2
`kniharstvirak.cz`