# Zpracování řečových signálů — studijní opora

Honza Černocký

Speech@FIT, Ústav počítačové grafiky a multimédií
Fakulta informačních technologií, Vysoké učení technické v Brně

http://www.fit.vutbr.cz/~cernocky

6. prosince 2006

# Obsah

# Kapitola 1

# Úvod

0:30

## 1.1 Slovo autora

Milí studenti ! Máte před sebou studijní oporu k předmětu "Zpracování řečových signálů" ZRE (ve starém studijním programu "Číslicové zpracování řeči" CZR). ZRE je standardně vyučován v prvním ročníku magisterského studia, je povinný na oboru Počítačová grafika a multimédia (MGM) a volitelný na ostatních magisterských oborech vyučovaných na FIT, tedy MIN, MIS a MPS.

Tato studijní opora byla původně koncipována jako pomůcka pro studenty distanční a kombinované formy studia, jako doplnění audiovizuálních záznamů přednášek a občasných konsultací. Doufám, že bude užitečná i pro studenty standardního presenčního studia, v žádném případě však nenahrazuje účast na přednáškách a nahlížení do skutečné odborné literatury ! Nemá ani v nejmenším ambice rovnat se dílům Josefa Psutky, Bena Golda a Nelsona Morgana a jiných klasiků – tyto knihy jsou k disposici v dostatečném množství v knihovně FIT.


Při studiu ZRE věnujte pozornost webové stránce tohoto kursu (i dalších řečových kursů):
`http://www.fit.vutbr.cz/~cernocky/speech`
a pokud Vás budou zajímat projekty či další spolupráce ve zpracování řeči, nezapomeňte navštívit stránky naší výzkumné skupiny **Speech@FIT**:
`http://www.fit.vutbr.cz/speech`

Přeji Vám krásný čas strávený se zpracováním řeči.

Honza Černocký

## 1.2 Základní údaje o kursu

### 1.2.1 Cíle předmětu


Seznámit studenty se základními charakteristikami řečového signálu v návaznosti na tvorbu a slyšení řeči lidmi. Popsat základní algoritmy analýzy řeči společné mnohým aplikacím. Podat přehled aplikací (rozpoznávání, syntéza, kódování) a informovat o praktických stránkách implementace řečových algoritmů.

### 1.2.2 Klíčová slova

Aplikace počítačového zpracování řeči, číslicové zpracování řečových signálů, tvorba a slyšení řeči, úvod do fonetiky, předzpracování a základní parametry, lineárně-prediktivní model, cepstrum, určování základního tónu hlasu, kódování - časová oblast a vokodéry, rozpoznávání - DTW a HMM, syntéza.

### 1.2.3 Získané dovednosti, znalosti a kompetence

Studenti se seznámí se základními charakteristikami řečového signálu v návaznosti na tvorbu a slyšení řeči lidmi. Pochopí základní algoritmy analýzy řeči

společné mnohým aplikacím. Získají přehled o aplikacích (rozpoznávání, syntéza, kódování) a o praktických stránkách implementace řečových algoritmů. Budou schopni navrhnout jednoduchý systém pro zpracování řeči (detektor řečové aktivity, rozpoznávač několika izolovaných slov), včetně implementace do aplikačních programů.

### 1.2.4 Piktogramy použité v této opoře

jsou uvedeny v tabulce 1.1

| | | | |
|---|---|---|---|
|  | Čas potřebný pro studium |  | Otázka, příklad k řešení |
|  | Cíl |  | Počítačové cvičení, příklad |
|  | Definice |  | Příklad |
|  | Důležitá část |  | Reference |
|  | Rozšiřující látka |  | Správné řešení |
|  | Obtížná část |  | Souhrn |
|  | |  | Slovo tutora, komentář |
|  | |  | Zajímavé místo |
|  | Pointer na web | | |

Tabulka 1.1: Význam používaných piktogramů

## 1.3 Závěr úvodu

Tato studijní opora vznikla v říjnu a listopadu roku 2006 (jako obvykle na poslední chvíli před termínem odevzdání) a Vy — studenti akademického roku 2006/07 — jste prvními, kdo ji dostane do rukou. Prosím všechny o připomínky, konstruktivní kritiku, korektury a morální podporu (alkohol nosit nemusíte).

Chtěl bych poděkovat především členům naší výzkumné skupiny za vytrvalou podporu a kritiku, která vedla k postupnému zvyšování úrovně slajdů k ZRE — ty jsou základem tohoto textu.

Velký dík patří také Slávkovi Křenovi za poskytnutí stylu do LaTeXu.

# Kapitola 2

# Program a organisace kursu

## 2.1 Kdo Vás bude učit

- přednášky: Honza Černocký a další lidé ze Speech@FIT (LVCSR, syntéza) — když tomu někdo rozumí lépe než Černocký (což je dost často ;-)
- snaha o externisty: podle dosažitelnosti např. Dr. Milan Jelínek [Univ. of Sherbrooke, Kanada]: kódování řeči, Pavel Cenek [FI MU]: dialogové systémy, VoiceXML.
- počítačová cvičení: asistenti a doktorandi Speech@FIT. Pokud budete potřebovat přiřadit jméno cvičícího či přednášejícího jeho xichtíku, pohleďte na `http://www.fit.vutbr.cz/speech`
  (jsou tam ale možná i věci zajímavější než naše xichty...).

## 2.2 Organisace kursu

- **Přednášky**
- **Počítačové laboratoře** — Matlab a C pod Linuxem, účast není kontrolována a hodnocena, ale vřele ji doporučujeme, protože v poč. laboratořích budou zadány a diskutovány projekty, a ty hodnoceny jsou ;-)
- **Projekty**, které jsou dva: výpočet LPC parametrů pro kódování řeči a tvorba Viterbiho dekodéru pro rozpoznávání řeči. U obou dostane každý individuální zadání (1×WAV pro kódování, několik WAV pro rozpoznávání). Výsledky+zdrojáky se budou odevzdávat přes WIS, probíhá automatická kontrola + korelace zdrojáků pro zamezení plagiarismu....
- **numerické cvičení** – jedno ke konci semestru, příklady na LPC, DTW, HMM, příklady budou ale i součástí přednášek.

## 2.3 Program přednášek

P1 Organisace kursu, aplikace, vědy.

P2 Číslicové zpracování řečových signálů: záznam řečového signálu - vzorkování, kvantování. Získání spektra řeči - Fourierova transformace - spojitý čas, co se děje, když navzorkujeme. Diskrétní Fourierova transformace. Náhodné signály, spektrální hustota výkonu. Úprava řeči - filtrace. Frekvenční charakteristiky filtru.

P3 Předzpracování řeči: střední hodnota, preemfáze, rámce, základní parametry parametry, Spektrogram. Tvorba řeči: Řečové ústrojí a jeho signálový model - hlasivky a artikulační trakt vs. buzení a filtr. Základní charakteristiky v čase a ve spektru: vliv buzení a filtru. Formanty. Co je vidět na short-term

a long-term spektrogramu. Jak od sebe oddělit buzení a filtr: cepstrum + MFCC.

P4 Lineárně-prediktivní model: Na co nám to vlastně je ? Chceme pouze charakteristiky hlas. traktu, ne buzení ⇒ aplikace v kódování a rozpoznávání. Předpověď následujícího vzorku z předcházejících - lineární predikce (LP). Chyba LP. Získání chyby LP jediným filtrem. Určení modelu artikulačního ústrojí pomocí pomocí LP analýzy. Spektrum pomocí lineární predikce. Parametry odvozené z LP - LAR a LSF, které se používají v kodérech. LPC-cepstrum.

P5 Určování základního tónu. Terminologie. Charakteristiky základního tónu mužů, žen a dětí. Využití v systémech zpracování řeči. Metody založené na autokorelační funkci. Normalizovaná cross-correlation function (NCCF). Dlouhodobý prediktor a cepstrální analýza pro určení základního tónu. Spolehlivost a problémy detektorů základního tónu.

P6 Kódování řeči I: Cíl kódování. Bitový tok, objektivní a subjektivní kvalita. Dělení kodérů podle bit. toku a kvality. Kódování signálu v časové oblasti. Vokodéry - LPC. Vektorové kvantování v kódování řeči.

P7 Kódování II. - aplikace CELP, Kódování v GSM.

P8 Úvod do rozpoznávání – úkol, klasifikace: isolated/connected/LVCSR, speaker dep/indep. Základní funkční bloky. Detekce řečové aktivity pro izolovaná slova.

Rozpoznávání založené na vzdálenostech řečových rámců - různé definice vzdáleností. lineární úprava, dynamické programování (Dynamic Time Warping DTW).

P9 Skryté Markovovy modely (Hidden Markov models HMMs): proč to děláme a vztah s DTW, modelování, Gaussova rozložení, sekvence stavů. Pravděpodobnost promluvy podle sekvence stavů, Baum Welchova a Viterbiho pravděpodobnost. Trénování modelů: Baum Welch, rozpoznávání - Viterbi. Token passing. Spojená slova.

P10 HMM II. Plynulá řeč s velkým slovníkem: Rozpoznávání pomocí menších jednotek - fonémy...

Fonetická stavba jazyka. Samohlásky a souhlásky, charakteristiky, dělení fonémů. Mezinárodní normy pro označování fonémů: IPA a SAMPA, TIMIT. Koartikulace.

A zpět do rozpoznávání: context-dep. triphones. Velký slovník, language modeling.

P11 Parametry pro rozpoznávání. Co od nich chceme - potlačení pitche, dekorelace, souvislost se spektrální obálkou. Jak to děláme a co používáme: LPCC, MFCC, pro dekorelaci PCA, LDA, HLDA, pro menší závislost na kanálu normalizaci. Další triky s features - delta, delta-delta. “Hot-topics v parametrizaci”: TRAPs a FeatureNet, neuronové sítě.

P12 Syntéza řeči: Struktura syntezátoru. Převod textu do mluvené podoby: text-to-speech. Text normalization. Prozodie (melodie, přízvuky, časování) v syntéze řeči. Jednotky pro syntézu - ruční a automatický výběr (corpus-based). Generování signálu v časové a frekvenční oblasti. Metody PSOLA a HNM. Aplikace. SW pro syntézu: EPOS, MBROLA, Festival.

P13 Průmyslové nasazení systémů pro zpracování řeči: Pavel Cenek [OptimTalk.com]. Dialogové systémy, VoiceXML, gramatiky pro popis rozpoznávačů, atd.

## 2.4 Numerická cvičení


Numeriko je jen jedno — 3 hodiny pro všechny: číslicový filtr, LPC - výpočet filtru, DTW, HMM.

## 2.5 Počítačové laboratoře

Počítačových laboratoři je 6 a jsou "proloženy" dvěma projekty:

L1 Zpracování řeči v Matlabu: čtení/zápis zvukových souborů, základní operace, ukládání, nahrávání. Zpracování signálů v Matlabu: návrh filtru, póly, nuly frekvenční charakteristiky, filtrace, Spektrální analýza: Fourierka, PSD.

L2 hraní se zvukem v C — vstup a výstup zvuku, dělení na rámce, výpočet základních parametrů, ladění C pomocí Matlabu.

L3 LPC v C: úkolem je napsat korelaci, algoritmus Levinsona a Durbina a funkci na výpočet energie. Check s Matlabem pro zvukový soubor. Příprava na kódování - uložení do jasné struktury.

**PROJEKT 1** : LPC v C — úkolem je dopsat to, co bylo zadáním poč. cvičení, uložení do binárního souboru. Bude k disposici dekodér v Matlabu, který soubor načte, bude možné ověřit, že to skutečně "hraje". Projekt se nebude zabývat detekcí základního tónu. Bude k disposici referenční WAV a referenční soubor s parametry pro ladění. Každý pak dostane osobní WAV, vygeneruje soubor s parametry, tento se bude aut. kontrolovat.

L4 NCCF a detekce základního tónu v Matlabu a v C, prahy. Kdo bude chtít: vyhlazení mediánovým filtrem, přidání do struktury z projektu, kompletní kodér.

L5a Příprava na rozpoznávání: LPCC, MFCC, ukládání souborů s parametry (pro trénování HMM nebo jako reference pro DTW), příprava na volání rozpoznávače.

L5b DTW – demo v Matlabu, úkolem bude napsat funkci v C nebo ji aspoň někde najít a nainterfacovat, rozpoznání několika souborů.

L6a HTK: prototypy, trénování, rozpoznávání, vyhodnocení. Rozpoznávač českých "ano"/"ne" nezávislý na řečníkovi, vyhodnocení chybovosti.

L6b HMM rozpoznávač: Seznámení s Viterbiho dekodérem - čtení modelů, Poskytneme funkci na MFCC, ověření s HTK, že to počítá věrohodnosti a rozpoznává stejně.

**PROJEKT 2** napsat Viterbiho dekodér (nelekejte se, má to asi 10 linek v C...), dáme modely a opět pár referenčních WAVek s věrohodnostmi a výsledky rozpoznávání. Pak dostane každý svou osobní sadu WAVek, do WISu se budou odevzdávat rozp. výsledky + věrohodnosti + zdrojový kód Vašeho dekodéru.


Jako rozšiřující látku pro zájemce doporučuji rovněž *staré laboratoře*:

x1 Dekodér LPC v C.

x2 Zpracování zvuku on-line. Z tohoto Vás nebudeme hodnotit, ale pozorně poslouchat by měl každý, kdo bude chtít dělat cokoliv se zvukem (rozpoznávání, kódování) *on-line*. Na základě SW Petra Schwarze je vysvětlen on-line vstup zvuku ve Win a Linuxu, dvou-vláknová struktura (jedno vlákno nahrává, druhé pracuje s rámci) a jak to celé napsat v C++.

x3 Syntéza: databáze s fonetickými značkami, pak syntéza z textu pomocí konkatenace: Konkatenace signálů s opravou energie.

## 2.6 Hodnocení

| co | za kolik |
|---|---|
| 2 projekty po 15-ti bodech | 30 |
| půlsemestrálka - pouze teoretické otázky | 20 |
| semestrálka - teorie i příklady | 50 |
| celkem | 100 |

- Obě zkoušky se všemi materiály...
- bodování počítačových cvik jsme zrušili.
- budou se hodnotit projekty. Ke každému se bude ve WISu odevzdávat soubor s výsledky (automatické hodnocení) + zdrojové kódy a základní dokumentace (korelační analýza, aby to jeden nenapsal všem ;-).

## 2.7 Referenční literatura

- Psutka J.: Komunikace s s počítačem mluvenou řečí. Academia, Praha, 1995.
- Gold B., Morgan. N.: Speech and audio signal processing, John Wiley & Sons, 2000.
- S. Young, J. Jansen, J. Odell, D. Ollason, P. Woodland: The HTK book, Entropics Cambridge Research Lab., Cambridge, UK. Výborný úvod do HMM, ke stažení po registraci zdarma na `http://htk.eng.cam.ac.uk/`


- `http://www.fit.vutbr.cz/~cernocky/speech/`
- staré verze některých textů v
  `http://www.fit.vutbr.cz/~cernocky/oldspeech/`
  (pozor, tento materiál je poskytován zcela bez záruky a bez podpory a může obsahovat ukrutné blbosti !).

# Kapitola 3

# Úvod do automatického zpracování řeči

## 3.1 Definice

DEF

"Automatické zpracování řeči umožňuje hlasovou komunikaci mezi lidmi navzájem (kódování) nebo mezi člověkem a strojem."

## 3.2 Vědy ve zpracování řeči

Zpracování řeči je pluri-disciplinární obor, využívající poznatků věd přírodních, technických i humanitních.

- **fysiologie:** porozumění funkci hlasového a sluchového ústrojí, pomoc při tvorbě různých modelů.
- **akustika:** studuje fyzikální mechanismy tvorby a slyšení řeči.
- **zpracování signálu:** mnoho zúčastněných oblastí: modelování, parametrisace, identifikace, spektrální analýza, kódování, teorie informace, klasifikace vzorů, atd.
- **humanitní vědy**
  - *fonetika* – o tvoření a slyšení zvukové stránky řeči.
  - *fonologie* – o fonémech a jejich systému v daném jazyce. *Foném* je hláska, která je v jazyce *významotvorná*, její změna může tedy měnit význam slova.
  - *prosodie* – o zvukové stránce jazyka (melodie, trvání hlásek, přízvuk ve slovech a ve větách).
  - *lexikologie* – o slovní zásobě jazyka, jejím vývoji z vztazích mezi slovy (*synonyma:* 2 různá slova, stejný význam, *antonyma*: opačný význam, *homonyma:* stejné slovo, více významů). Napomáhá tvorbě *slovníků* a to i počítačových, důležité pro rozpoznávání řeči.
  - *gramatika (mluvnice)* – soubor pravidel o obměnách slov a o jejich spojování ve věty. Důležité pro syntézu.
  - *syntaxe* – část gramatiky zabývající se větou a souvětím.
  - *sémantika* – o významu jazykových jednotek. Vychází obvykle od *slova*, důležitá pro porozumění řeči.

Vědám odpovídají také různé úrovně zpracování řečového signálu — příklad na větě "Přišel jsem pozdě".

1. *akustická* - signál, spektrogram, parametry.
2. *fonetická* - p ř i š e l  j s...

3. *lexikální* - sloveso: tvar "přijít", sloveso: tvar "být", příslovce.
4. *syntaktická* - (nevyjádřený podmět) přísudek příslovečné určení času.
5. *sémantická* - sdělení, že jsem přišel pozdě.
6. *pragmatická* - co jsem svým sdělením sledoval - omluva, prosté oznámení ?

Pragmatická úroveň musí vzít v úvahu význam slov vzhledem k *záměru člověka*. V automatickém zpracování řeči je to zatím neřešená otázka.

## 3.3 Informační obsah řeči

snažíme se kvantifikovat *informační rychlost* (v bitech za sekundu, ve zkratce bit/s nebo bps), nutnou k popisu řeči v různých formách. Pro srovnání uvedeme formu fonetickou a akustickou s číslicovým vyjádřením signálu.

### 3.3.1 Fonetická forma

počet fonémů v češtině je 36. Pro vyčíslení *množství informace* budeme uvažovat zdroj generující na sobě navzájem nezávislé prvky $x_i$ z množiny $X = \{x_1, \ldots, x_S\}$, kde $S$ je konečný počet prvků. Každý z prvků má určitou pravděpodobnost $p(x_i)$ a prvky tvoří úplnou soustavu, takže:

$$\sum_{i=1}^{S} p(x_i) = 1. \tag{3.1}$$

Informační obsah $i$-tého prvku je dán počtem bitů, které potřebujeme k jeho vyjádření:

$$I(x_i) = -\log_2 p(x_i) \quad [\text{bit}]. \tag{3.2}$$

*Entropie zdroje* (střední hodnota informace) je pak dána:

$$H(X) = -\sum_{i=1}^{S} p(x_i) \log_2 p(x_i). \tag{3.3}$$

Pro české fonémy, předpokládáme-li zjednodušeně stejnou pravděpodobnost jejich výskytu, je tato entropie $H(X)$=5.2 bit. Vezmeme-li v úvahu jejich pravděpodobnost, je to $H(X)$=4.6 bit. Vezmeme-li navíc ještě v úvahu vzájemné *závislosti* mezi fonémy (bigramy: $p(x_j|x_i)$, trigramy: $p(x_k|x_ix_j)$, atd.), dosahuje entropie hodnoty $H(X)$=3–3.5 bit.

V běžném českém hovoru produkuje člověk cca 80–130 slov za minutu, tedy asi 10 fonémů za sekundu. Informační rychlost je tedy asi $C_{phn}$ = **30–40 bit/s**. Psychoakustické testy ukázaly, že člověk je schopen zpracovat příchozí informace do rychlosti asi 50 bit/s.

### 3.3.2 Akustická forma

Je-li řeč representována číslicovým signálem, musí být splněn Nyquistův–Shannonův–Kotelnikovův teorém:

$$F_s > 2F_m, \tag{3.4}$$

kde $F_s$ je vzorkovací kmitočet a $F_m$ je nejvyšší kmitočet obsažený ve spektru signálu. Každý vzorek je kvantován jednou $m$ možných kvantovacích hladin, k

jejich vyjádření potřebujeme $N = \log_2 m$ bitů. Odstup signálu od šumu v dB je úměrný hodnotě $6N$. Vyjádříme-li pomocí těchto veličin informační rychlost, dostaneme:

$$C_{ak} = \frac{I}{t} = \frac{\log_2 m}{T_s} = NF_s. \tag{3.5}$$


**Příklady:**

1. Signál s Hi-Fi kvalitou, $F_s$=44.1 kHz, $N$=16 bit. Výsledná informační rychlost je $C_{ak}$ = **705 kbit/s**.
2. Signál s telefonní kvalitou (pásmo omezeno od 300 do 3400 Hz): $F_s$=8 kHz, $N$=8 bit. Výsledná informační rychlost je $C_{ak}$ = **64 kbit/s**.

### 3.3.3 Závěr ?

Z příkladů je patrné, že akustická forma má oproti fonetické formě obrovskou *redundanci* (nadbytečnost). Mimo informace obsažené ve fonetické formě nese totiž informaci o osobnosti mluvčího, o barvě jeho hlasu, náladě, prostředí, atd. Mozek posluchače pak provádí dekódování a separaci různých typů informací — Bohu žel, nevíme přesně jak. Přesto je vhodné využít alespoň základních informací o *tvorbě řeči* k **omezení této informační rychlosti.**

## 3.4 Aplikační oblasti zpracování řeči

### 3.4.1 Kódování

se používá pro přenos a ukládání. Jeho **cílem** je vyjádření řeči co nejmenším počtem bitů. Na kódování klademe následující **požadavky:**

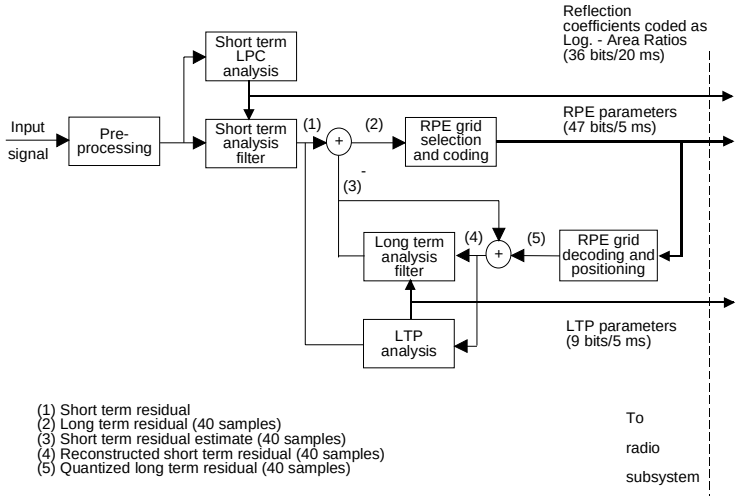
- co nejmenší náročnost.
- co nejmenší zpoždění.
- co největší srozumitelnost.
- co největší přirozenost.
- co největší odolnost proti chybám v kanálu.

A je zřejmé, že tyto požadavky jdou proti sobě (malých bitových rychlostí typicky dosahují složité kodéry, ty jsou ale značně náročné na paměť a CPU a mají velké zpoždění).

Pro představu náročnosti běžného kodéru (LTP-RPE – norma GSM full rate) uvádíme jeho schéma na Obr. 3.1. Stručný přehled kvalita a bitové rychlosti kodérů uvádí Obr. 3.2.

### 3.4.2 Rozpoznávání

- **řeči** (Speech Recognition)
  - izolovaná slova
  - spojená slova (např. číslovky při zadávání tlf. čísla nebo čísla kreditní karty).
  - plynulá řeč (continuous speech) – nejtěžší úkol, zatím uspokojivě nevyřešeno ve 100% případů, zvl. v případě velkých slovníků (LVCSR - Large Vocabulary Continuous Speech Recognition).




Obrázek 3.1: Schéma kodéru RPE-LTP – GSM full rate

Obrázek 3.3 presentuje pokrok ve vývoji rozpoznávačů vzhledem k velikosti slovníku a stylu mluvy (od izolovaných povelů až po spontánní řeč).

- **mluvčího** (Speaker Recognition)
  - identifikace – Který ze skupiny mluvčích mluví ?
  - verifikace – je člověk, který prohlašuje, že je pan Novák, skutečně pan Novák ?

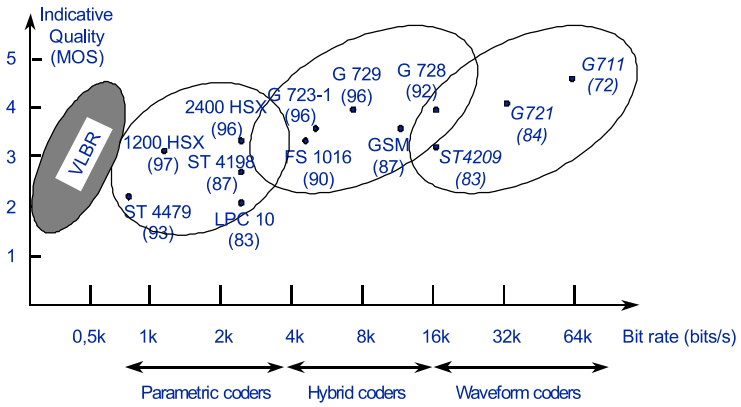
## 3.5 Syntéza

počítač má za úkol říci text, který nikdy předtím "neslyšel" (tj. který nebyl předem namluven člověkem). Spektrum syntezačních metod je široké a sahá od jednoduchého "spojování wavek" (viz vozidla MHD) až po syntézu z textu (TTS – text-to-speech). Stránka `http://www.cs.indiana.edu/rhythmsp/ASA/Contents.html` uvádí zajímavé příklady syntézy řeči.

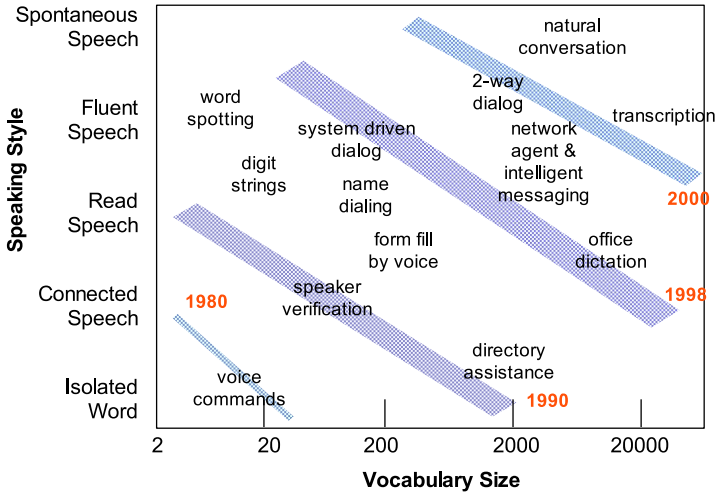
WWW

## 3.6 Další aplikace zpracování řeči

- medicína: zkoumání anomálií a chorob hlasového traktu.
- psychologie a kriminalistika: detektor lži, identifikace alkoholu v krvi, detekce stresu či únavy,...
- pomoc handicapovaným (učení hluchých dětí správné výslovnosti, atd.)
- identifikace jazyka
- detekce klíčových slov




Obrázek 3.2: Přehled bitových rychlostí a kvality současných kodérů řeči. Obrázek převzat od Andreas Spanias (Arizona University) `http://www.eas.asu.edu/ speech/table.html`




Obrázek 3.3: Vývoj rozpoznávačů řeči.

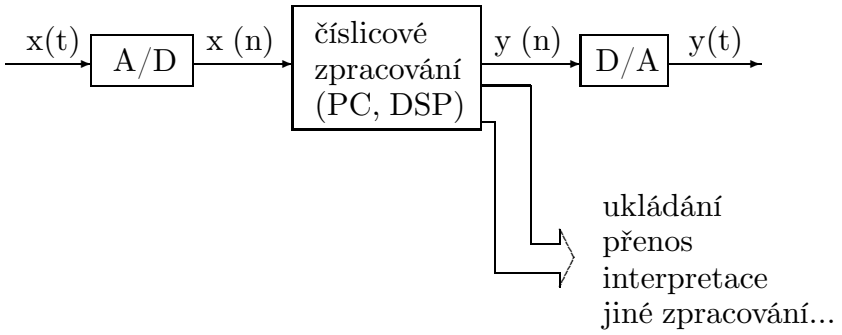
# Kapitola 4

# Zpracování signálu – shrnutí


Cílem této kapitoly je presentovat ve zhuštěné formě základní poučky o číslicovém zpracování signálů. Pro lepší porozumění doporučuji ale navrátit se pokorně k Vašim zápiskům z kursu Signály a systémy (ISS) a především si projet jeho počítačová cvičení na:
`http://www.fit.vutbr.cz/~cernocky/sig`


## 4.1 Vzorkování a spol.

Typické schéma zpracování signálu je na Obr. 4.1. Ne vždy se po ukončeném zpracování navracíme zpět do "analogového světa signálů", velmi často (např. při rozpoznávání nebo další interpretaci) vystačíme s číslicovou informací.




Obrázek 4.1: Schéma zpracování signálu.

Na začátku zpracování je signál se spojitým **časem**: je definován všude od $-\infty$ do $\infty$, a čas má $\infty$ hodnot (levá strana Obr. 4.2). Pro representaci signálu ve **frekvenční oblasti (spektrum)** použijeme Fourierovu transformaci:

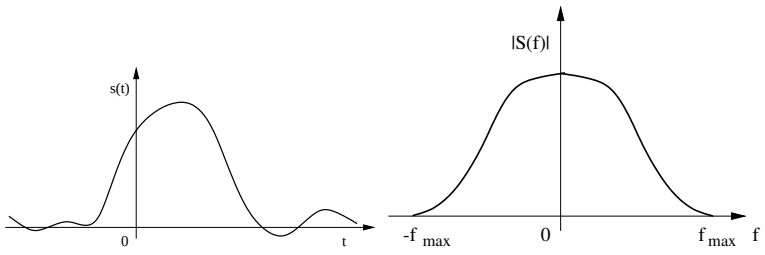
$$X(f) = \int_{-\infty}^{-\infty} x(t) e^{-j2\pi ft} dt, \tag{4.1}$$


kde funkci $X(f)$ říkáme **spektrální funkce**, nebo krátce **spektrum**. Funkce je definována pro $\forall f$ od $-\infty$ do $\infty$ a je komplexní. Má modul $|X(f)|$ a argument $\angle X(f)$. Hovoříme o modulovém (pravá část Obr. 4.2) a argumentovém spektru. Pro reálné signály nám stačí znát pouze pravou část spektrální funkce ($f > 0$), protože část levá je s ní komplexně sdružená:

$$X(f) = X^{\star}(-f), \tag{4.2}$$

neboli $|X(f)| = |X(-f)|$ a $\arg X(f) = -\arg X(-f)$.

- Inteligentní signály jsou frekvenčně omezené – mají energii pouze v pásmu $(0, f_{max})$.

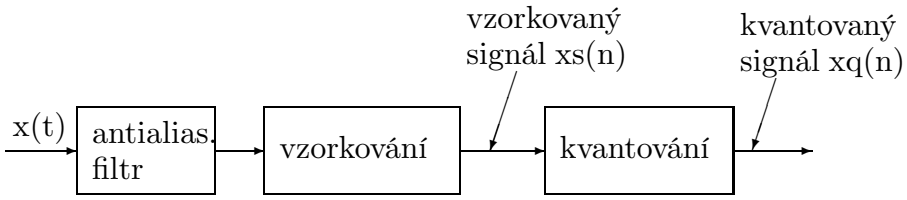



Obrázek 4.2: Signál a jeho spektrální funkce (modulová).

- spektrální funkce se **nedá spočítat** (nekonečna, integrál,...) a v praxi ji dokážeme pouze odhadnout pomocí diskrétní Fourierovy transformace.

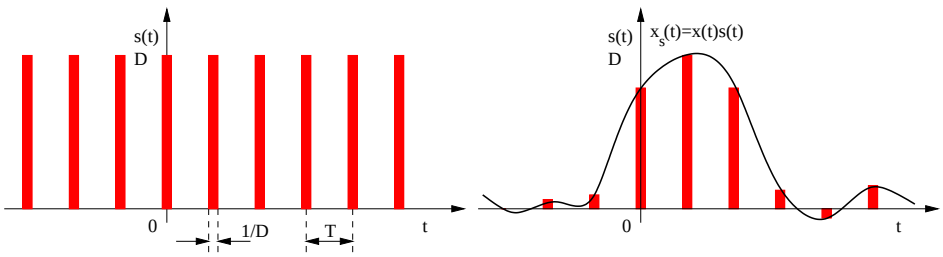
## 4.2 Analogově – číslicový (AD) převod

můžeme schematicky zachytit pomocí tandemu vzorkování a kvantování (Obr. 4.3):




Obrázek 4.3: Analogově-číslicový převod: vzorkování a kvantování.

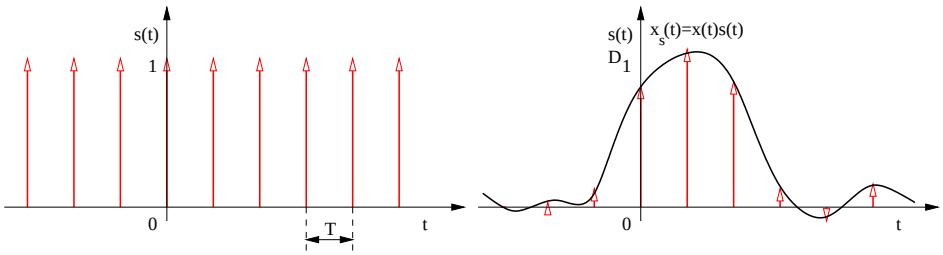
**Vzorkovaný signál** dostaneme tak, že původní signál vynásobíme něčím, co je periodické v čase – sledem obdélníkových impulsů (Obr. 4.4).




Obrázek 4.4: Násobení periodickým sledem obdélníkových impulsů.

Teoreticky vysvětlujeme vzorkování tak, že násobíme signál periodickým sledem Diracových impulsů (nekonečná výška, nulová šířka, plocha – "mocnost" 1). Po násobení dostaneme opět periodický sled Diracových impulsů, ale s mocnostmi danými hodnotami původního signálu v bodech $nT$ (Obr. 4.5).

DEF




Obrázek 4.5: Násobení periodickým sledem Diracových impulsů.

$T$ je vzorkovací perioda

$F_s = \frac{1}{T}$ je vzorkovací frekvence

Jak to vypadá se *spektrem* vzorkovaného signálu ? Periodizuje se !

$$X_s(f) = \frac{1}{T} \sum_{n=-\infty}^{+\infty} X\left(f - \frac{n}{T}\right) = \frac{1}{T} \sum_{n=-\infty}^{+\infty} X\left(f - nF_s\right) \tag{4.3}$$

Podle vztahu maximální frekvence obsažené ve spektru signálu $f_{max}$ a vzorkovací frekvence rozlišujeme dva případy:

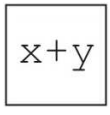
1. $F_s > 2f_{max}$: Jednotlivé kopie původního spektra se nepřekrývají a původní signál můžeme *ideálně rekonstruovat* tak, že vzorkovaný signál vyfiltrujeme dolní propustí s mezním kmitočtem $F_s/2$.
2. $F_s \leq 2f_{max}$: Jednotlivé kopie původního spektra se překrývají, výsledné spektrum má *jiný* tvar než původní spektrum. Původní signál *nemůžeme* žádným způsobem rekonstruovat, dochází k takzvanému **aliasingu**.

**Shannonův–Kotelnikovův–Nyquistův–vzorkovací teorém** udává podmínku pro ideální vzorkování, kdy k aliasingu nedochází:

$$F_s > 2f_{max}$$

jinými slovy: maximální frekvence obsažená v signálu musí být nižší, než polovina vzorkovací frekvence.

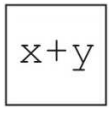
## Příklad 1.




Máme signál a vzorkování s následujícími parametry: $F_s = 8000\ Hz$, $f_{max} = 3000\ Hz$, a tedy $\Omega_s = 16000\pi$ rad/s, $\omega_{max} = 6000\pi$ rad/s. $T = \frac{1}{8000}$ s
Dojde k aliasingu ? Bude možné signál ideálně rekonstruovat ?

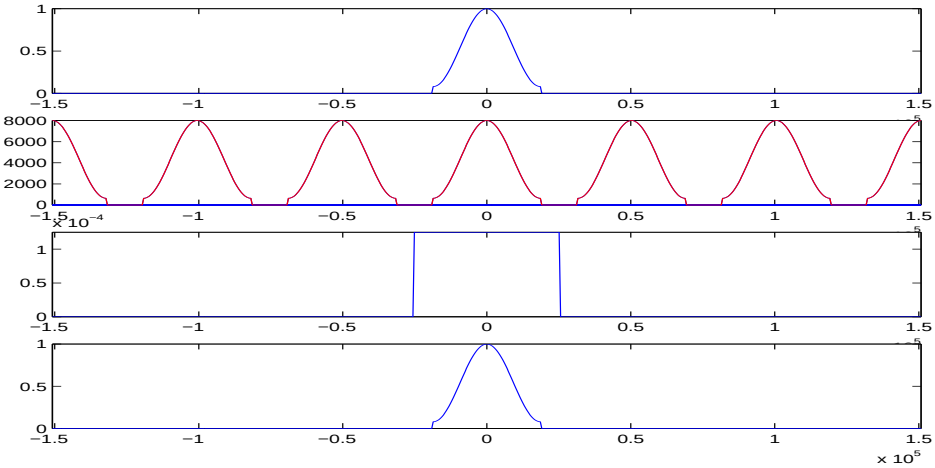
Řešení je graficky vyznačeno na Obr. 4.6

## Příklad 2.




Máme signál a vzorkování s následujícími parametry: $F_s = 8000\ Hz$, $f_{max} = 7000\ Hz$, a tedy $\Omega_s = 16000\pi$ rad/s, $\omega_{max} = 14000\pi$ rad/s. $T = \frac{1}{8000}$ s
Dojde k aliasingu ? Bude možné signál ideálně rekonstruovat ?

Řešení je graficky vyznačeno na Obr. 4.7




Obrázek 4.6: Spektrální funkce při vzorkování a rekonstrukci. Zvrchu dolů: původní spektrální funkce, spektrální funkce navzorkovaného signálu, rekonstrukční filtr a spektrální funkce rekonstruovaného signálu, která je přesně shodná s původní.

### 4.2.1 Antialiasingový filtr

který omezí spektrum původního signálu do intervalu $[-F_s/2,\ F_s/2]$ je alespoň částečným řešením situace: omezíme sice a-priori frekvenční rozsah signálu, ale alespoň nedojde k "překlopení" vyšších frekvencí do užitečného intervalu $[-F_s/2,\ F_s/2]$. Obrázek 4.8 demonstruje použití anti-aliasingového filtru před vzorkováním.

### 4.2.2 Zápis vzorkovaného signálu

Vzorkovaný signál zapisujeme $x_s(nT)$ nebo také jen $x_s[n]$ není to nic jiného než *posloupnost čísel*.

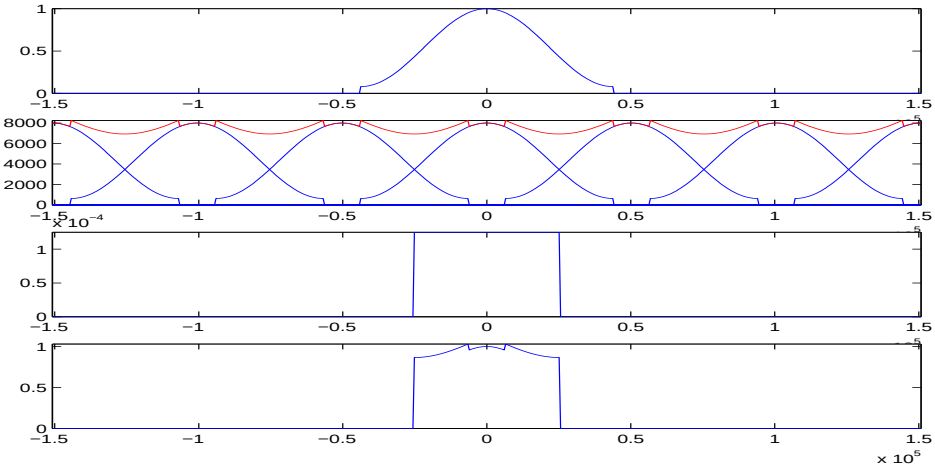
1. Máme-li vzorkovaný signál, musíme k němu dostat i informaci o vzorkovací frekvenci (implicitně: vzorky ze zvukové karty) přicházejí s periodou $T$, explicitně: např. hlavička souboru WAV).
2. počítáme-li se vzorkovanými signály, rádi se času zcela zbavíme. Budeme předpokládat periodu $T' = 1$, tedy $F_s' = 1$. Normovaný čas je pak dán:

$$t' = \frac{t}{T}, \quad \text{takže} \quad n = \frac{nT}{T} \tag{4.4}$$

a normovaná frekvence

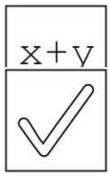
$$f' = \frac{f}{F_s} \tag{4.5}$$

Jelikož jsou ale zpracovatelé signálu *lenoši*, čas většinou nepoužívají vůbec a fakt, že se jedná o normovanou frekvenci nijak neoznačují. Ve vzorcích se normovaná frekvence pozná tak, že blízko ní nikde nestojí žádný "pořádný čas" $t$ ani vzorkovací perioda $T$.



Obrázek 4.7: Spektrální funkce při vzorkování a rekonstrukci, kdy dojde k aliasingu. Zvrchu dolů: původní spektrální funkce, spektrální funkce navzorkovaného signálu, rekonstrukční filtr a spektrální funkce rekonstruovaného signálu, která je od původní odlišná.

## Příklad




Napište funkci pro generování cosinusovky s frekvencí 200 Hz pro vzorkovací kmitočet $F_s = 8000$ Hz.

Pro spojitý čas je signál dán: $s(t) = \cos(2\pi f_0 t) = \cos(2\pi 200t)$. Při přechodu na vzorkovaný signál nahradím spojitý čas $t$ diskrétním časem $nT$, kde $T$ je vzorkovací perioda: $x(nT) = \cos(2\pi f_0 nT) = \cos\left(2\pi \frac{f_0}{F_s}\right) n$.

Frekvence $\frac{f_0}{F_s}$ je normovaná frekvence. Výsledný signál můžeme zapsat zapsat: $x(n) = \cos\left(2\pi \frac{1}{400} n\right)$.

Generování 1s takového signálu v Matlabu:

```
n = 0:7999;
x = cos (2 * pi * 1 / 400 * n);
wavwrite(x,8000,16,'sig.wav');
```

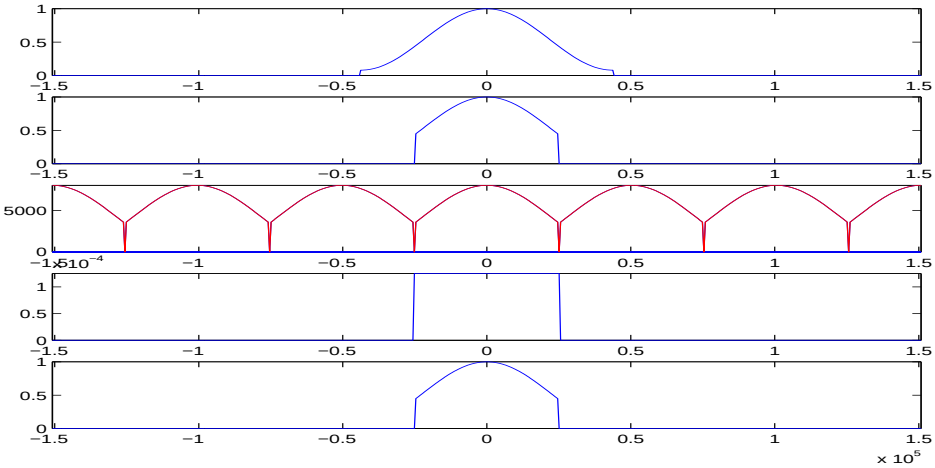
### 4.2.3 Chování vzorkovaného signálu ve frekvenční oblasti — spektrum


Spektrum vzorkovaného signálu je možné spočítat pomocí **Diskrétní Fourierovy transformace** – **DFT**:

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{nk}{N}} \quad \text{pro } k \in < 0, N-1 > \tag{4.6}$$

Jak však aplikovat DFT na diskrétní signál?:

- analyzujeme "okno" o délce $N$ vzorků.



Obrázek 4.8: Spektrální funkce při vzorkování a rekonstrukci, s použitím anti-aliasingového filtru. Zvrchu dolů: původní spektrální funkce, spektrální funkce po průchodu anti-aliasingovým filtrem, spektrální funkce navzorkovaného signálu, rekonstrukční filtr a spektrální funkce rekonstruovaného signálu.

- co bude vlastně výsledkem ? Vynásobím-li hodnoty $X(k)$ vzorkovací periodou $T$, dostanu aproximaci spektrální funkce ve frekvenčních bodech $k\Delta f$, kde $\Delta f = \frac{F_s}{N}$ (skutečná frekvence) nebo $\Delta f' = \frac{1}{N}$ (normovaná frekvence, i když, jak jsme si řekli, $f'$ nikdo nepoužívá ☹

$$\hat{X}(k\Delta f) = T \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{nk}{N}} \tag{4.7}$$

  $x[n]$ v této rovnici můžeme volně zaměnit za $x(nT)$.

### Co získáme pomocí DFT

Oproti spektrální funkci získané "analogovou" FT jsme ovšem v žádném případě nespočítali totéž !!!

1. počítáme spektrum *vzorkovaného* signálu, takže je toto spektrum nutně *periodické*, a to s periodou $N$ čísel, což odpovídá vzorkovací frekvenci $F_s$ (u frekvence se raději vyhneme označení "vzorek"). Pokud necháme $k \in (-\infty, +\infty)$, zjistíme, že $\hat{X}(k\Delta f)$ se po $N$ hodnotách opakují.
2. signál jsme "vykousli" oknem. Spočtené spektrum nese i vlastnosti tohoto okna: okno v čase *násobí* signál, spektrum okna se tedy ve frekvenci konvoluuje se spektrem signálu. Toto s sebou často nese rozmazání teoreticky ostrých spektrálních čar (např. při analýze harmonického signálu). Více v kapitole o předzpracování.
3. spektrum je *diskrétní* (máme k disposici pouze $N$ hodnot od 0 do $F_s$), takže jsme vlastně spočetli spektrum **periodického signálu** ! Můžeme si to představit tak, že okno signálu se $\infty$-krát opakuje.

**Shrnutí: co je čím způsobeno**

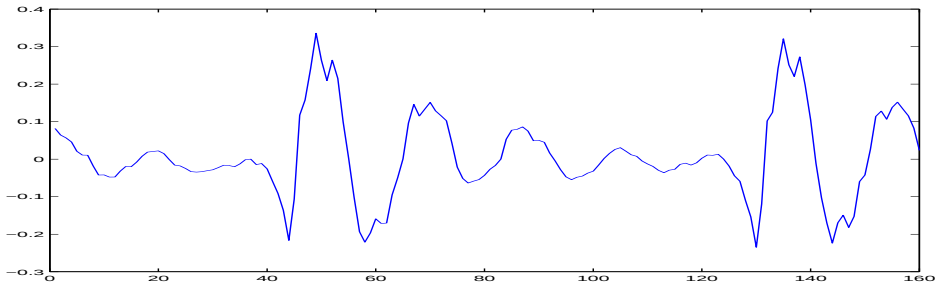
| čas | frekvence |
|---|---|
| vzorkování | periodisace |
| periodisace | diskretisace |

### 4.2.4 Jak na frekvenční analýzu prakticky ?

Chceme frekvenčně analyzovat jeden znělý řečový rámec (č. 13 z "létajícího prasete"):

```
s = wavread('test.wav')';
sfr = frame (s,160,80);
x = sfr(:,13);
plot (x);
```

Vybraný rámec je na Obr. 4.9




Obrázek 4.9: Načtený rámec pro frekvenční analýzu.

Spočítáme-li pouze DFT, musíme si dát pozor na správnou frekvenční osu:

```
Fs = 8000; f = (0:159) / 160 * Fs; X = fft(x);
subplot (211); plot(f,abs(X)); subplot (212); plot(f,angle(X));
```

výsledek je na Obr. 4.10.

A vzhledem k tomu, že horní polovina spektra je symetrická se spodní a moc nás nezajímá, zobrazíme pouze jednu polovinu:

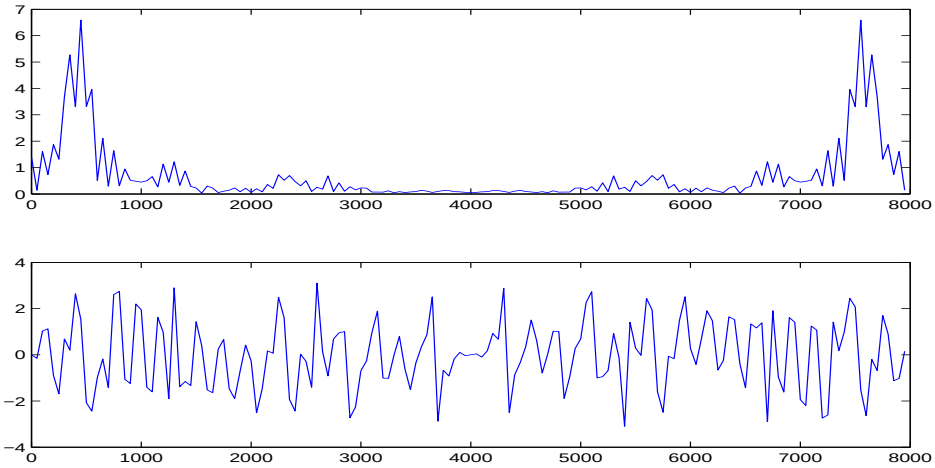
```
Fs = 8000; f = (0:79) / 160 * Fs; X = fft(x); X = X(1:80);
subplot (211); plot(f,abs(X)); subplot (212); plot(f,angle(X));
```

Výsledek viz Obr. 4.11

Při výpočtu spektra někdy požadujeme "hladší" výstup s větším množstvím bodů. Řešením by bylo prodloužení rámce, ale předpokládejme, že jeho délka je daná a že jej prodloužit nelze. Můžeme si pomoci technikou doplňování nul **"zero padding"**:


```
Fs = 8000; f = (0:511) / 1024 * Fs;
X = fft([x' zeros(1,1024-160)]); X = X(1:512);
subplot (211); plot(f,abs(X)); subplot (212); plot(f,angle(X));
```

Výsledek je patrný na Obr. 4.12. Uvědomme si, že přidáním nul nepřidáváme do výpočtu spektra žádnou informaci, pouze "zkrášlujeme" výsledný obrázek.




Obrázek 4.10: DFT rámce.

## 4.3 Frekvenční analýza náhodných signálů


Z hlediska teorie se řečové signály pokládají za **náhodné**. Měly by se tedy frekvenčně analyzovat pomocí **spektrální hustoty výkonu (power spectral density – PSD)**, která je reálná a udává rozdělení výkonu ve frekvenční oblasti. Jeden z odhadů PSD využívá DFT:

$$\hat{G}_{DFT}(k\Delta f) = \frac{1}{N}|X[k]|^2. \tag{4.8}$$

jedná se tedy pouze o absolutní hodnotu modulů na druhou.

V Matlabu si dáme na výpočet $|X[k]|^2$. pozor – druhá mocnina komplexního čísla není totéž co druhá mocnina modulu komplexního čísla:

- `X = fft(x); Gdft = X .^ 2;` špatně !
- `X = fft(x); Gdft = abs(X) .^ 2;` dobře !
- `X = fft(x); Gdft = X .* conj(X);` dobře a navíc rychle ☺

U dělení ve vztahu 4.8 si dáme pozor na to, že dělíme počtem vzorků na vstupu, nikoliv délkou "prodloužené" DFT.

### Příklad - výpočet spektrální hustoty výkonu rámce

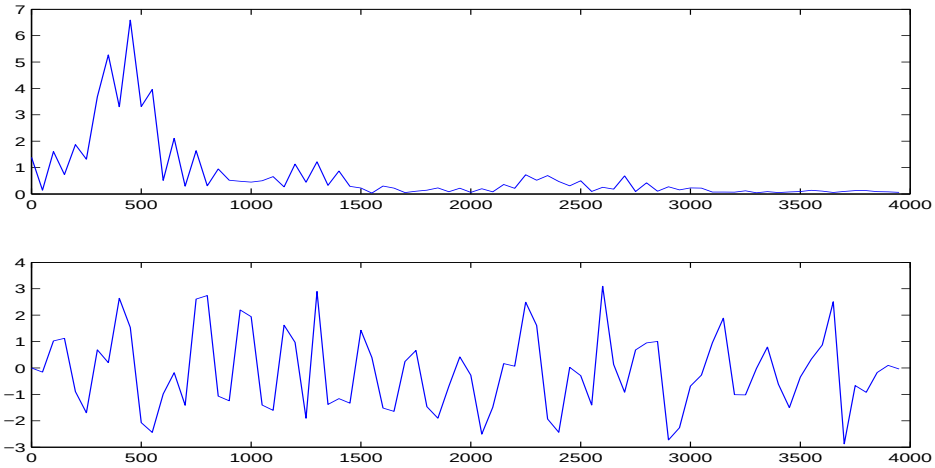
```
Fs = 8000; f = (0:511) / 1024 * Fs
X = fft([x' zeros(1,1024-160)]); X = X(1:512);
Gdft= 1/160 *abs(X) .^ 2; plot(f,Gdft);
```

Výsledek je zobrazen na Obr. 4.13

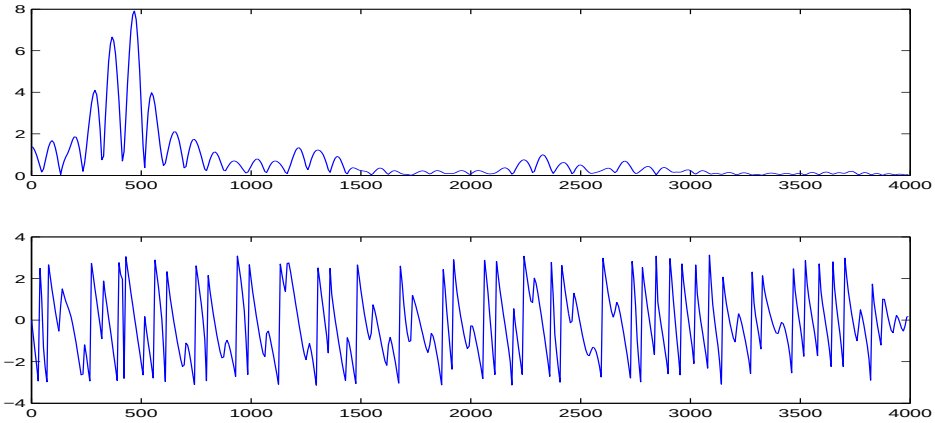
Dynamika spektrální hustoty výkonu je větší než u DFT (druhá mocnina...) a na obrázcích nejsou vidět "slabé" části, proto se často používá zobrazení v decibelech (Matlab: funkce `log10`):

$$\hat{G}_{DFT}(k\Delta f) = 10\log_{10}\frac{1}{N}|X[k]|^2.$$

Výsledek je vidět na spodním panelu Obr. 4.13.




Obrázek 4.11: DFT rámce – spodní polovina spektra.



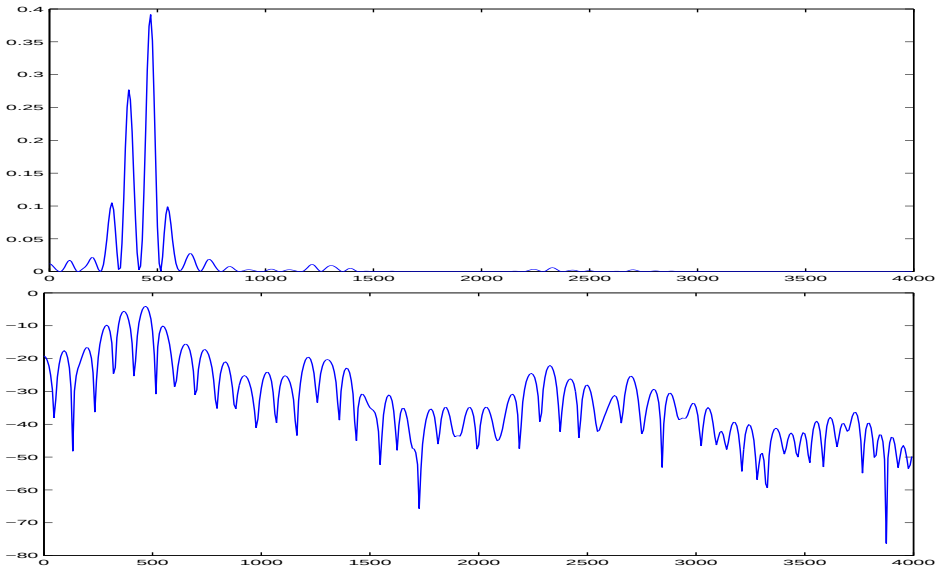

Obrázek 4.12: DFT rámce – zero padding.

## 4.4 Lineární filtrace

Lineární filtr použijeme, chceme-li nějak upravit obsah kmitočtových složek v signálu:


Běžné filtry jsou

- lineární — zachovávají lineární kombinaci: pokud $x_1(n) \rightarrow y_1(n)$ a $x_2(n) \rightarrow y_2(n)$, pak $ax_1(n) + bx_2(n) \rightarrow ay_1(n) + by_2(n)$, kde $a, b \in \Re$.
- časově invariantní — chovají se "stále stejně": pokud $x(n) \rightarrow y(n)$, pak také $x(n-n_0) \rightarrow y_(n-n_0)$, kde $n_0$ je libovolný posuv. Někdy však naopak *chceme*, aby se charakteristiky filtru v čase měnily — adaptivní systémy, řečové rámce (změna $\forall$10 ms).




Obrázek 4.13: Spektrální hustota výkonu v lineární stupnici a v decibelech.

- kauzální — filtr "nevidí do budoucnosti": $y(n)$ závisí pouze na $y(m < n)$ a $x(m \leq n)$.

### 4.4.1 Impulsní odezva

DEF

nebo také **impulsní charakteristika** je vrácena filtrem při buzení Kroneckerovým či jednotkovým impulsem (není to stejné, co Diracův v případě spojitých signálů, diskrétní jednotkový impuls je zcela běžným signálem a lze ho bez problémů vygenerovat):

$$\delta(n) = \begin{cases} 0 & \text{pro } n \neq 0 \\ 1 & \text{pro } n = 0 \end{cases} \tag{4.9}$$


Známe-li impulsní odezvu, můžeme spočítat, jak bude filtr reagovat na libovolný vstupní signál. Každý vstupní vzorek totiž "spustí" jednu impulsní odezvu (násobenou velikostí vstupního vzorku), a ty se na výstupu sečtou – nezapomeňme, že filtr je lineární. Můžeme zapsat **konvolucí**:

DEF

$$y(n) = x(n) \star h(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m) = \sum_{m=-\infty}^{\infty} h(m)x(n-m) \tag{4.10}$$

O impulsní charakteristice můžeme říci:

- pokud $h(k) = 0$ pro $\forall k < 0$, pak je filtr kauzální (vzorky po $n$-tém nebudou násobeny ničím nenulovým).

- impulsní odezva může být konečná — FIR (finite impulse response) nebo nekonečná — IIR (infinite impulse response).
- její Fourierův obraz ve frekvenci udává komplexní kmitočtovou charakteristiku filtru:

$$h(k) \rightarrow H(f)$$

Konvoluci v časové oblasti odpovídá *součin* v oblasti kmitočtové, takže spektrum výsledného signálu je:

$$Y(f) = X(f)H(f) \tag{4.11}$$

Mějme na paměti, že pracujeme s diskrétními signály (i impulsní odezva filtru je diskrétní), vše je tedy ve frekvenci *periodické* a to s periodou $F_s$ (nebo 1 pro normovanou frekvenci).

### 4.4.2 Jak vypadá filtr

Schéma obecného filtru je na Obr. 4.14 Blok $z^{-1}$ označuje zpoždění o 1 vzorek. Chování filtru lze zapsat **diferenční rovnicí**:

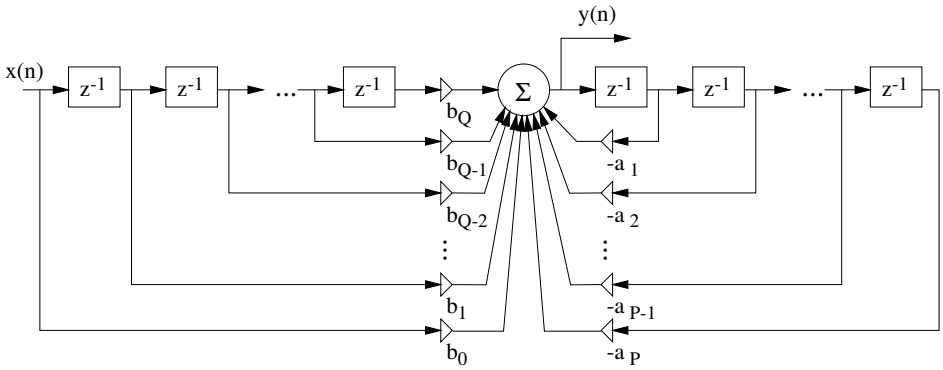
$$y(n) = \sum_{k=0}^{Q} b_k x(n-k) - \sum_{k=1}^{P} a_k y(n-k), \tag{4.12}$$

kde $x(n-k)$ jsou aktuální a zpožděné verze vstupu a $y(n-k)$ jsou zpožděné verze výstupu.

Základní typy filtrů:

- **FIR** – nerekurzivní: jen $b_0 \ldots b_Q$ nenulové. Je vždy stabilní.
- **IIR** – čistě rekurzivní: jen $b_0, a_1 \ldots a_P$ nenulové.
- **IIR** – obecně rekurzivní: $a_i$ i $b_i$ nenulové.

Z diferenční rovnice se ovšem těžko dá přímo poznat chování filtru ve frekvenční oblasti a těžko také vyšetříme jeho *stabilitu*.




Obrázek 4.14: Číslicový filtr.

### 4.4.3 $z$-transformace

nám pomůže právě při zjišťování chování filtru ve frekvenci (komplexní kmitočtová charakteristika) a stability filtru. Pro diskrétní signál $x[n]$ je $z$-transformace dána:

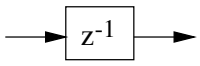
DEF

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}, \tag{4.13}$$

kde $z$ je komplexní proměnná. Existují slovníky $z$-transformace, které udávají $z$-obrazy pro různé typy signálů, ty však nebudeme vůbec potřebovat. Budeme předpokládat, že signál $x(n)$ má $z$-transformaci $X(z)$. Definujeme poučku o **zpoždění**: je-li $x(n) \rightarrow X(z)$, pak pro $y(n) = x(n - n_0)$ bude:

$$Y(z) = z^{-n_0} X(z) \tag{4.14}$$

pro zpoždění o jeden vzorek platí: $x(n-1) \rightarrow z^{-1}X(z)$. Proto značíme zpoždění o 1 vzorek




### 4.4.4 Přenosová funkce filtru

Diferenční rovnici je možné pomocí $z$-transformace přepsat na:

$$Y(z) = \sum_{k=0}^{Q} b_k X(z) z^{-k} - \sum_{k=1}^{P} a_k Y(z) z^{-k}, \tag{4.15}$$

**Přenosovou funkci** můžeme definovat jako podíl:

DEF

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{Q} b_k z^{-k}}{1 + \sum_{k=1}^{P} a_k z^{-k}} = \frac{B(z)}{A(z)}, \tag{4.16}$$

kde $B(z)$ a $A(z)$ jsou dva polynomy. Koeficient polynomu jmenovatele $a_0$ musí být "povinně" roven 1, ve filtru se fyzicky nevyskytujeme, je to vlastně matematické vyjádření toho, že filtr má výstupní vzorek.

### 4.4.5 Frekvenční charakteristika

DEF

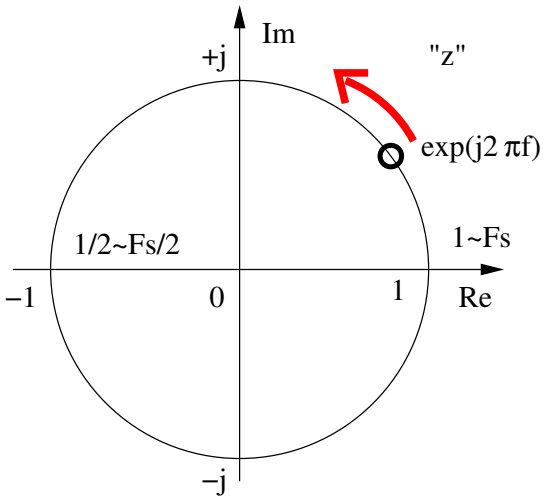
filtru od 0 do $F_s$ (nebo od 0 do 1 v normované frekvenci) se snadno získá z přenosové funkce tak, že "objedeme" jednotkovou kružnici a budeme zaznamenávat komplexní hodnoty funkce $H(z)$:

$$H(f) = H(z)|_{z=e^{j2\pi f}} \tag{4.17}$$

pro normovanou frekvenci nebo:

$$H(f) = H(z)|_{z=e^{j2\pi fT}} \tag{4.18}$$

pro "obyčejnou" frekvenci. Pro každou hodnotu $f$ vyčíslíme polohu bodu na jednotkové kružnici: $z = e^{j2\pi f}$ (komplexní číslo - viz Obr. 4.15), pak pro toto číslo vypočteme podíl polynomů $B(z)$ a $A(z)$ (také komplexní číslo). V Matlabu za nás tento výpočet pro celý interval zajímavých frekvencí (od 0 do $F_s/2$) provede funkce `freqz`.




Obrázek 4.15: Bod $z = e^{j2\pi f}$, pro který vyšetřujeme hodnoty přenosové funkce a jeho pohyb při zvyšování frekvence.

### 4.4.6 Nuly a póly přenosové funkce a co s nimi...

Přenosovou funkci $H(z)$ můžeme upravit a zapsat také pomocí součinů:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_Q z^{-Q}}{1 + a_1 z^{-1} + \ldots + a_P z^{-P}} = \frac{z^{-Q}(b_0 z^Q + b_1 z^{Q-1} + \ldots + b_Q)}{z^{-P}(z^P + a_1 z^{P-1} + \ldots + a_P)} =$$

$$= b_0 \frac{z^{-Q}}{z^{-P}} \frac{\prod_{k=1}^{Q}(z - n_k)}{\prod_{k=1}^{P}(z - p_k)} = b_0 z^{P-Q} \frac{\prod_{k=1}^{Q}(z - n_k)}{\prod_{k=1}^{P}(z - p_k)},$$

Hodnoty $n_k$ jsou kořeny čitatele a nazývají se **nulové body** nebo **nuly**. Hodnoty $p_k$ jsou kořeny jmenovatele a nazývají se **póly**. Kořeny libovolného polynomu dostaneme tak, že tento položíme rovný nule a vyřešíme.


Pokud $a_k, b_k \in \Re$, pak póly $p_k$ a nuly $n_k$ mohou být buď reálné, nebo ve dvojicích komplexně sdružené. Z poloh nul a pólů se dá graficky určit přibližný průběh frekvenční charakteristiky $H(f)$.

**Stabilita** filtru je zajištěna, pokud všechny póly leží *uvnitř jednotkové kružnice:*

$$\boxed{|p_k| < 1}$$

#### Příklad filtru

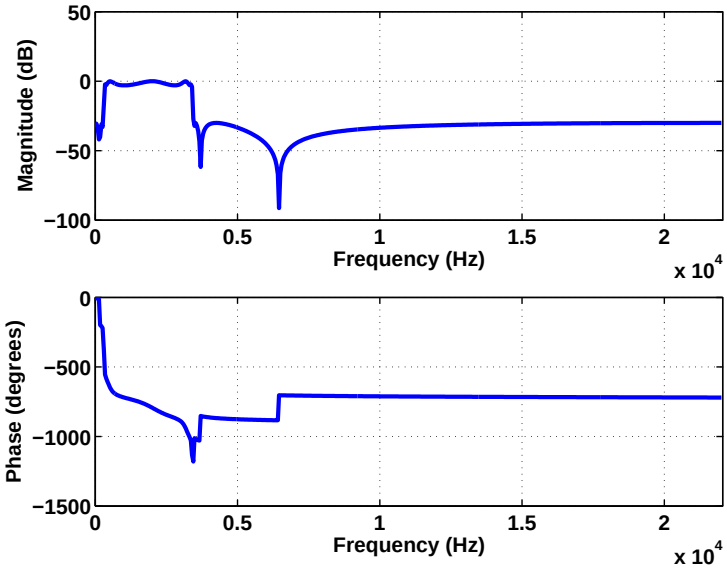
Chceme filtr, který bude simulovat telefonní kanál pro filtrování signálů s CD kvalitou. Bude to pásmová propusť od 300 do 3400 Hz. V Matlabu můžeme použít mnoho funkcí pro návrh filtrů, vybíráme tzv. eliptické filtry:

```
Fs = 44100; Fs2 = Fs/2;   % musi se normovat polovinou Fs
Wp = [300/Fs2 3400/Fs2];  % pass-band
```

```
Ws = [200/Fs2 3500/Fs2];  % stop-band - priblizne
Rp = 3;                   % zvlneni v pass-bandu dB
Rs = 30;                  % potlaceni stop-bandu dB (obe hodnoty od
  % oka, preseneji viz normy.
[N, Wn] = ellipord(Wp, Ws, Rp, Rs)  % vypocet radu filtru
[B,A] = ellip(N,Rp,Rs,Wn)           % vypocet polynomu B a A
```

. . . výsledkem jsou 2 polynomy 12-tého řádu.

Frekvenční charakteristika se vypočítá pomocí:
`freqz (B,A,512,Fs);`
a je vidět na Obr. 4.16. Póly a nuly: dostaneme pomocí:
`zplane (B,A);`
a výsledek je na Obr. 4.17.




Obrázek 4.16: Frekvenční charakteristika filtru.

### 4.4.7 Implementace filtru v C

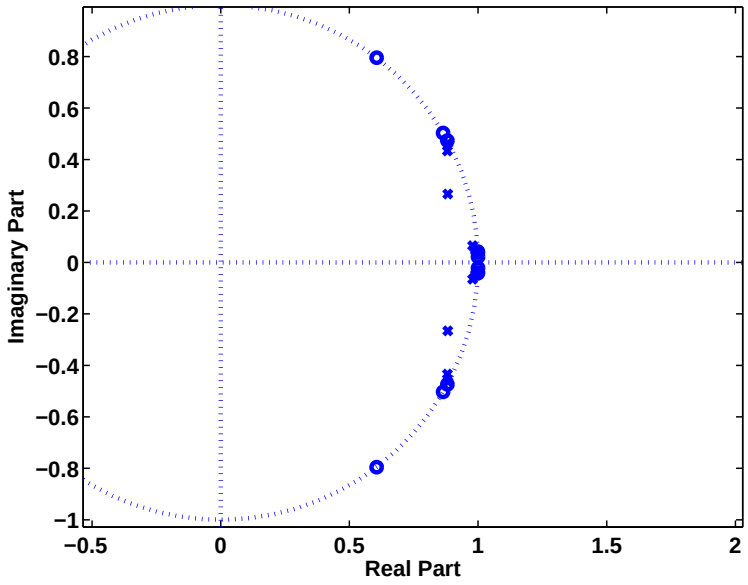
- základní implementace přímé struktury je velmi jednoduchá – prakticky se přepíše diferenční rovnice (4.12): viz soubor `filter.c` na webové stránce kursu.
- v praxi se používají optimálnější struktury, které mají pouze jednu zpožďovací linku a jsou méně náchylné k zaokrouhlovacím chybám.


více o teorii filtrů viz kurs ISS – přednáška "diskrétní systémy":
`http://www.fit.vutbr.cz/~cernocky/sig`

### 4.4.8 Průchod náhodného signálu filtrem

filtr má komplexní kmitočtovou charakteristiku $H(f)$. Pro vstupní signál se spektrální hustotou výkonu $G_x(f)$ je výstupní spektrální hustota výkonu dána:

$$G_y(f) = |H(f)|^2 G_x(f)$$

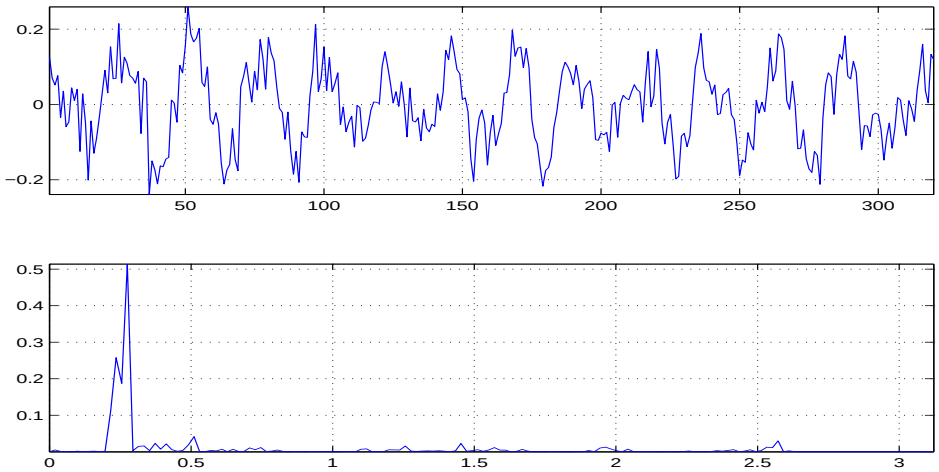



Obrázek 4.17: Póly a nuly filtru.

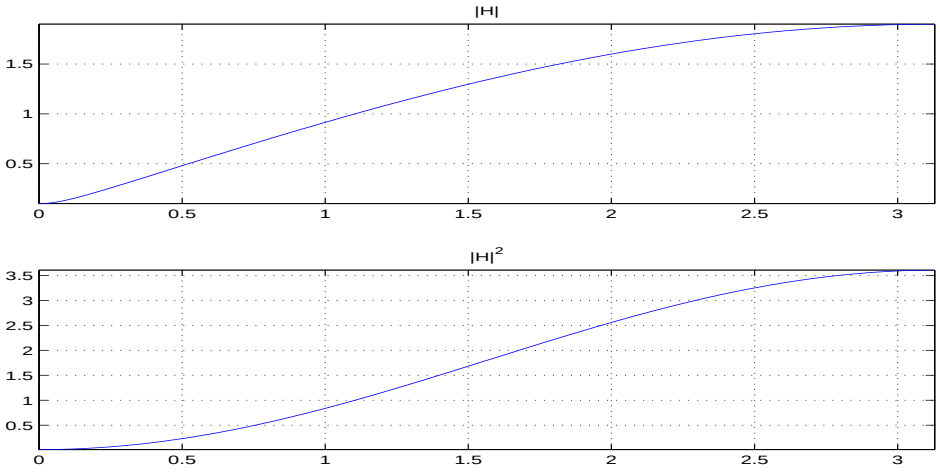
. . . vstupní PSD násobíme druhou mocninou **modulu** komplexní kmitočtové charakteristiky.

## Příklad:

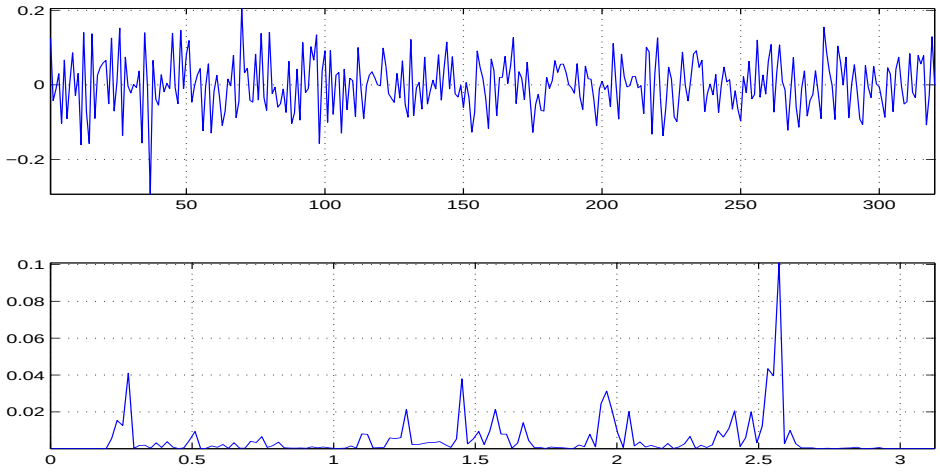
Byl nahrán zvuk protékání vody vodovodní trubkou. Filtrujeme jednu realizaci tohoto signálu filtrem $H(z) = 1 - 0.9z^{-1}$. Vstupní signál a jeho spektrální hustota výkonu je patrná na Obr. 4.18. Modul komplexní kmitočtové charakteristiky filtru a jeho druhá mocnina jsou na Obr. 4.19. Výstupní signál a jeho spektrální hustota výkonu pak na Obr. 4.20.




Obrázek 4.18: Vstupní signál a jeho PSD.




Obrázek 4.19: Modul komplexní kmitočtové charakteristiky filtru a jeho druhá mocnina.




Obrázek 4.20: Výstupní signál a jeho druhá mocnina.

# Kapitola 5

# Předzpracování řeči, tvorba řeči, cepstrum


Cílem této kapitoly je představit tvorbu řeči v nás — lidech a pokusit se toto schéma vtělit do automatického zpracování. Uvidíme, že základem zpracování řeči je *segmentace* na krátké časové úseky a výpočet příznaků, které tyto úseky popisují.

## 5.1 Úkol parametrizace


Úkolem "parametrisace", (anglicky "feature extraction") je vyjádřit řečový signál omezeným množstvím hodnot. V klasické literatuře jsou metody popisu děleny na:

- *neparametrický popis*, který je založen pouze na poznatcích o zpracování signálu (banky filtrů, Fourierova transformace, atd.)
- *parametrický popis*, který je založen na poznatcích o tvorbě řeči.

Druhá technika ovšem používá mnoho technik neparametrického popisu, takže tyto dvě skupiny není snadné (a někdy ani žádoucí) oddělit. Vypočtené hodnoty stejně vždy nazýváme *parametry*.

### 5.1.1 Typy parametry

Podle charakteru dělíme parametry na

- **skalární** – jedno číslo na řečový úsek (krátkodobá energie nebo počet průchodů nulou).
- **vektorové** – sada čísel (vektor) na řečový úsek. Pokud je více řečových úseků, řadíme vektorové hodnoty do *matic* tak, že čas běží vodorovně (Obr. 5.1).

## 5.2 Předzpracování (pre-processing)

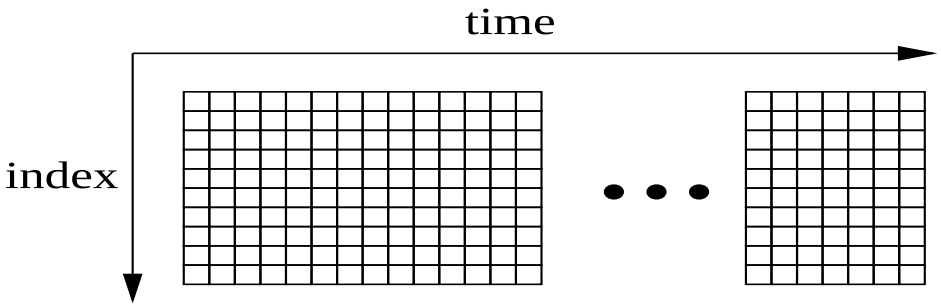
Před vlastním výpočtem parametrů je vhodné provést několik kroků:

### 5.2.1 Ustřednění

Stejnosměrná složka (dc-offset) nenese žádnou užitečnou informaci, naopak může být pro další zpracování rušivá (např. výpočet energie). Bude tedy dobré ji odstranit odečtením střední hodnoty:

$$s'[n] = s[n] - \mu_s, \qquad \text{kde } \mu_s \text{ musíme odhadnout.} \tag{5.1}$$

Střední hodnotu je ovšem možné počítat dvěma různými způsoby:




Obrázek 5.1: Řazení vektorových parametrů do matic.

**Střední hodnota off-line**

se spočítá prostým průměrováním po ukončení signálu:

$$\bar{s} = \frac{1}{N}\sum_{n=1}^{N} s[n] \tag{5.2}$$

Výsledek výpočtu s off-line odhadem střední hodnoty je patrný na Obr. 5.2.

**Střední hodnota on-line**

Nemáme k disposici celý signál: je příliš dlouhý, nebo neustále "přibývá". Střední hodnota se pak dá odhadnout rekurzivně:

$$\bar{s}[n] = \gamma\bar{s}[n-1] + (1-\gamma)s[n], \tag{5.3}$$

kde $\gamma \longrightarrow 1$. To je ekvivalentní filtraci signálu filtrem s impulsní odezvou:

$$h = [(1-\gamma)\ \ (1-\gamma)\gamma\ \ (1-\gamma)\gamma^2\ \ \ldots]. \tag{5.4}$$

Pro kontrolu si pojďme udělat součet těchto vah. Jedná se o geometrickou posloupnost: počáteční člen $a_0 = 1-\gamma$ a kvocient $q = \gamma$. Její součet je tedy:

$$\sum_{n=0}^{\infty} h[n] = \frac{a_0}{1-q} = \frac{1-\gamma}{1-\gamma} = 1, \tag{5.5}$$

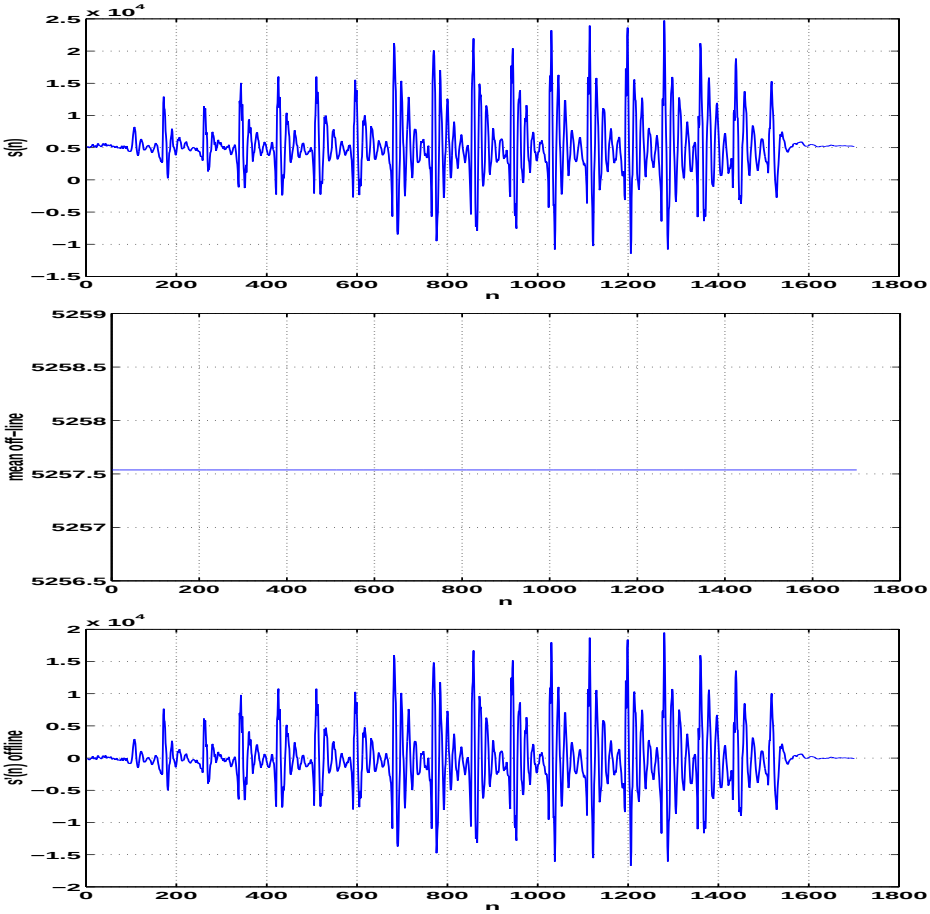
což jsme u výrazu počítajícího střední hodnotu očekávali ☺. Příklad pro $\gamma = 0.99$ je na Obr. 5.3.

### 5.2.2 Preemfáze

se stará o vyrovnání kmitočtové charakteristiky řeči (energie řeči klesá směrem k vyšším frekvencím). Jedná se spíše o historickou operaci, pokud si uvědomíme, co se děje při výpočtu Mel-frekvenčních cepstrálních koeficientů (viz dále v sekci 5.8), nemá preemfáze žádný vliv.

Zvýraznění vyšších frekvencí je možné realizovat jednoduchým filtrem 1. řádu:

$$H(z) = 1 - \kappa z^{-1}, \tag{5.6}$$




Obrázek 5.2: Horní panel: původní signál. Prostřední: off-line odhad střední hodnoty (pro všechny vzorky stejný). Spodní panel: ustředněný signál.

kde $\kappa \in [0.9,\ 1]$. Filtr tedy prakticky počítá rozdíl dvou sousedních vzorků. Modulová frekvenční charakteristika filtru pro $\kappa$=0.95 je na Obr. 5.4. Filtrace pak probíhá zcela standardně:

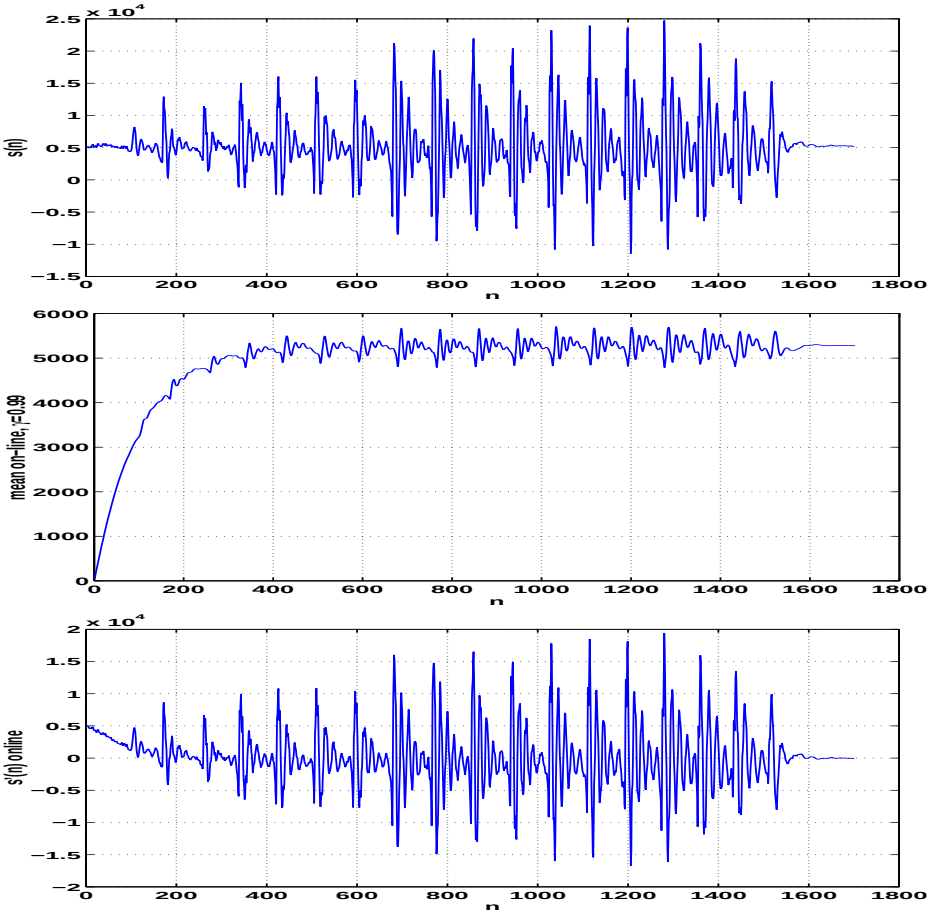
$$s'[n] = s[n] - \kappa s[n-1] \tag{5.7}$$

a její výsledek pro běžný signál je vidět na Obr. 5.5. Je zřejmé, že preemfázovaný průběh je "kostrbatější" a má více "ostrých hran", to ukazuje na větší podíl vyšších frekvencí.

## 5.3 Segmentace na rámce

Rámce jsou úseky signálu, na které je nutné řečový signál před dalším zpracováním rozdělit. Proč je to nutné ? Řečový signál považujeme za *náhodný*, pro metody odhadu parametrů by měl být *stacionární*. Bohu žel není, proto jej dělíme na kratší úseky (rámce, segmenty, mikrosegmenty, frames). Tam stacionární bude nebo budeme doufat, že bude.... Parametry rámců: délka (length) $l_{ram}$, překrytí (overlap) $p_{ram}$, posun rámce (frame shift) $s_{ram} = l_{ram} - p_{ram}$.

DEF



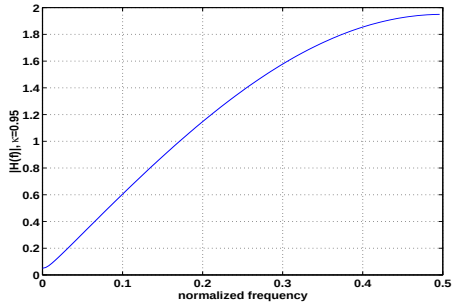

Obrázek 5.3: Horní panel: původní signál. Prostřední: on-line odhad střední hodnoty. Spodní panel: ustředněný signál.

**Délka rámců** by měla být dostatečně *malá*, aby bylo možné pokládat signál na daném úseku za stacionární, ale na druhé straně dostatečně *velká*, aby bylo možné dostatečně přesně odhadnout požadované parametry. Kompromisem je délka respektující setrvačnost hlasového ústrojí, typicky 20–25 ms (160–200 vzorků pro $F_s$ =8000 Hz).

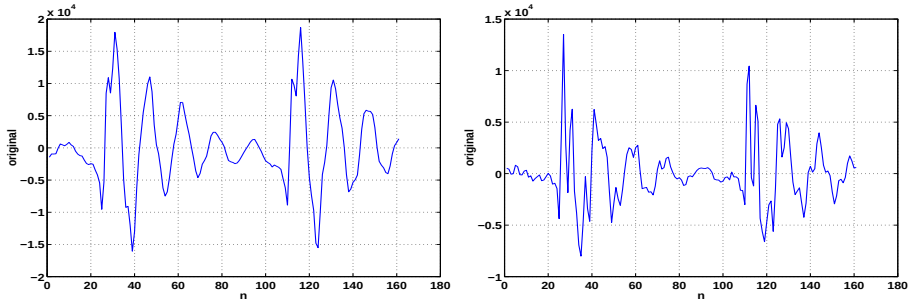
**Překrytí rámců** by bylo vhodné:

- **malé nebo žádné:** zajišťuje rychlý časový posun v signálu, malé nároky na paměť/procesor, hodnoty parametrů se však od jednoho rámce ke druhému mohou hodně měnit.
- **velké:** zajišťuje pomalý časový posuv, "vyhlazené" průběhy parametrů, avšak má velké nároky na paměť/procesor. Výsledné parametry mohou být navíc rámec od rámce příliš podobné, což jde proti požadavku statistické nezávislosti při rozpoznávání pomocí skrytých Markovových modelů (viz kapitola 11).

Kompromisem je typická délka 10 ms, tedy 100 rámců za vteřinu, mluvíme o centisecond vectors.



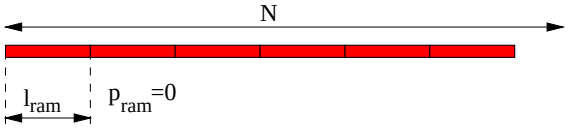

Obrázek 5.4: Frekvenční charakteristika preemfázovacího filtru.




Obrázek 5.5: Signál před a po preemfázi.

### 5.3.1 Kolik rámců pro řečový signál o délce $N$ ?

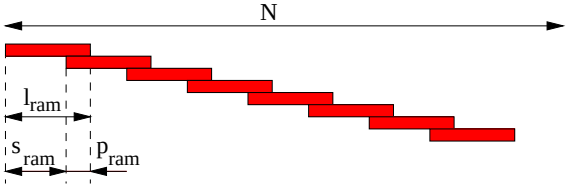
U rámců bez překrývání, kde $p_{ram} = 0$ (běžné pro kódování)




dostaneme jejich počet prostým dělením a zaokrouhlením dolů (floor). Předpokládáme, že poslední rámec, na který již nemáme dost vzorků, zahazujeme.

$$N_{ram} = \left\lfloor \frac{N}{l_{ram}} \right\rfloor , \tag{5.8}$$

Pro rámce s překrýváním, kde $p_{ram} \neq 0$ (běžné pro rozpoznávání):




je počet rámců (pokud je signál alespoň jeden rámec dlouhý) dán:

$$N_{ram} = 1 + \left\lfloor \frac{N - l_{ram}}{s_{ram}} \right\rfloor \tag{5.9}$$

### 5.3.2 Výběr signálu do rámců - okénkové funkce

Při "vykrojení" rámce je použito "okno" - window(ing) function. Z různých typů vybíráme pouze dvě nejpoužívanější:

**Pravoúhlé (rectangular) okno** – se signálem neudělá nic:

$$w[n] = \begin{cases} 1 & \text{pro} \quad 0 \leq n \leq l_{ram} - 1 \\ 0 & \text{jinde} \end{cases} \tag{5.10}$$

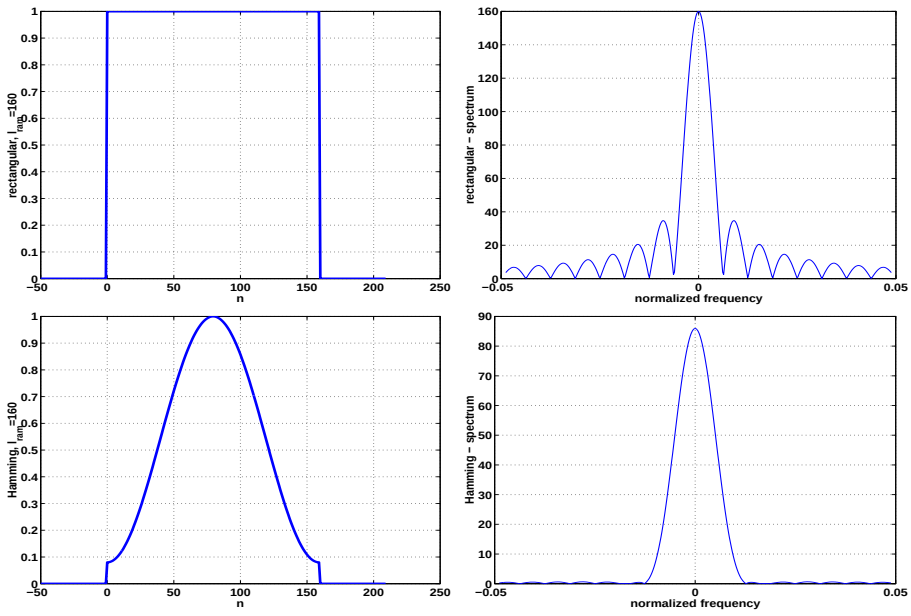
**Hammingovo okno** – utlumí signál na okrajích:

$$w[n] = \begin{cases} 0.54 - 0.46 \cos \dfrac{2\pi n}{l_{ram} - 1} & \text{pro} \quad 0 \leq n \leq l_{ram} - 1 \\ 0 & \text{jinde} \end{cases} \tag{5.11}$$

Oknem se ovšem změní spektrum vykusovaného signálu. Násobení signálu oknem v časové oblasti totiž odpovídá *konvoluce* spektra řeči se spektrem okna:

$$X(f) = S(f) \star W(f) \tag{5.12}$$

Srovnáme-li pravoúhlé a Hammingovo okno (Obr. 5.6), vidíme, že pravoúhlé je sice selektivnější (má užší centrální lalok), ale "začuní" spektrum výsledného signálu většími vysokofrekvenčními komponentami. Hammingovo okno je oproti němu "čistší", avšak méně selektivní.




Obrázek 5.6: Srovnání pravoúhlého a Hammingova okna v časové a frekvenční oblasti.

## 5.4 Základní parametry řečového signálu

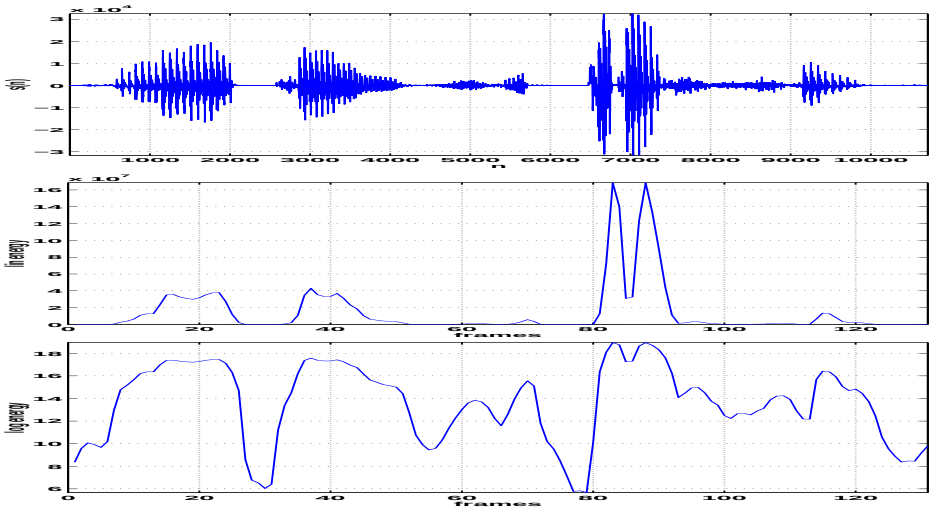
všechny budeme odvozovat na jednotlivých rámcích. Pokud pracujeme se všemi rámci, předpokládáme, že pro všechny rámce tvoří skalární parametry jeden řádkový vektor, pro vektorové parametry pak matici, kde svisle je index parametrů, vodorovně čas.

### 5.4.1 Střední krátkodobá energie

DEF

$$E = \frac{1}{l_{ram}} \sum_{n=0}^{l_{ram}-1} x^2[n] \tag{5.13}$$

nám poslouží jako detektor řečové aktivity a pro rozlišení hlásek na znělé (vysoká energie) a neznělé (nízká). Často pracujeme s log-energií, která má příznivější (menší) dynamický rozsah. Je nutné si uvědomit, že energie nebude příliš spolehlivá pro detekci řeči a ticha v šumu, především bude selhávat u nízkoenergetických hlásek, které jsou šumem "zamaskovány". Obr. 5.7 uvádí příklad lineární a logaritmické energie.




Obrázek 5.7: Signál a průběh jeho lineární a logaritmické krátkodobé energie.

### 5.4.2 Počet průchodů nulou (zero-crossing rate)

DEF

určuje, kolikrát za rámec projde signál nulou:

$$Z = \frac{1}{2} \sum_{n=1}^{l_{ram}-1} |\operatorname{sign} x[n] - \operatorname{sign} x[n-1]|, \tag{5.14}$$

kde $\operatorname{sign}(x)$ je zjednodušená znaménková funkce:

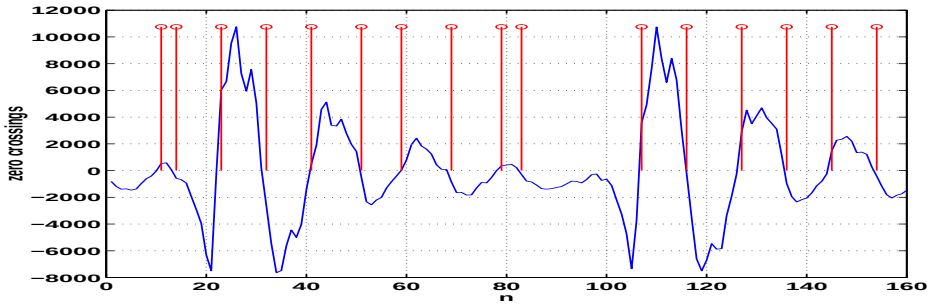
$$\operatorname{sign} x[n] = \begin{cases} +1 & \text{pro} \quad x[n] \geq 0 \\ -1 & \text{pro} \quad x[n] < 0 \end{cases} \tag{5.15}$$

Jak to funguje ? Funkce $|\operatorname{sign} x[n] - \operatorname{sign} x[n-1]|$ dává hodnotu 2 pro každý případ, že se od vzorku $x[n-1]$ ke vzorku $x[n]$ změní znaménko (Obr. 5.8).

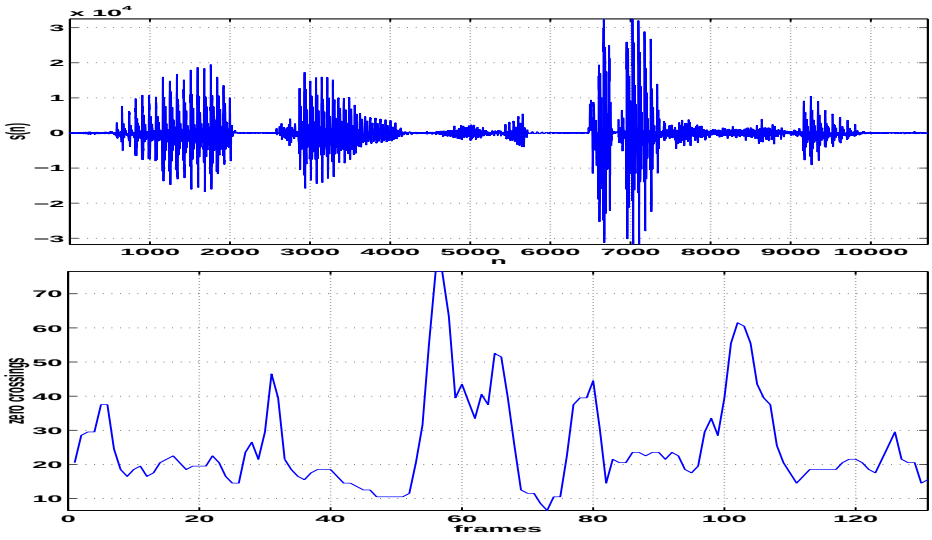
Detekce průchodů nulou je dobrá pro rozlišení hlásek na znělé (málo průchodů) a neznělé (více jako šum, tedy více průchodů). Je ale extrémně citlivá na šum a na posuny stejnosměrné složky. Příklad je na Obr. 5.9.

## 5.5 Hlasové ústrojí člověka a jeho model

Hlasové ústrojí je na Obr. 5.10.



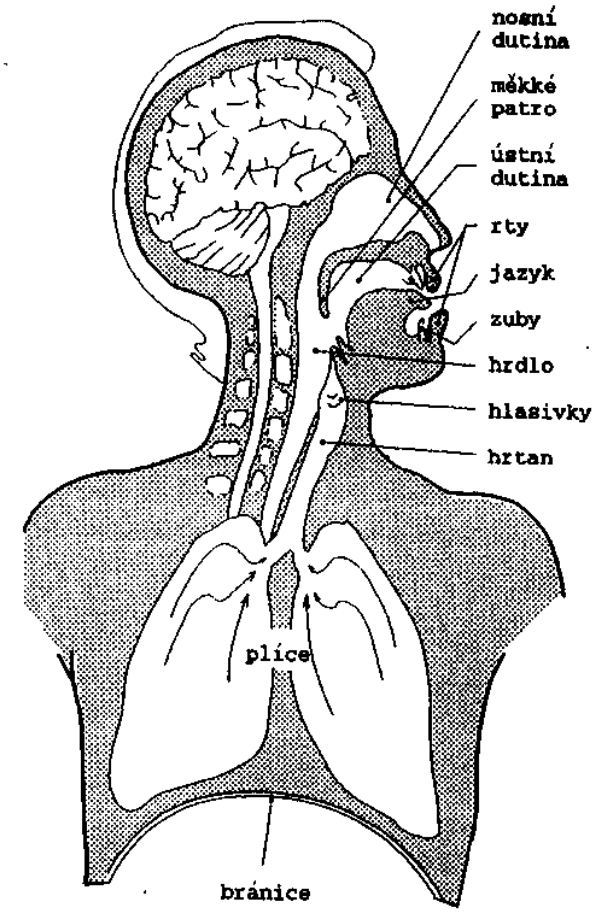

Obrázek 5.8: Průchody nulou – jeden rámec. Pro průchody nulou má funkce $|\operatorname{sign} x[n] - \operatorname{sign} x[n-1]|$ hodnotu 2.




Obrázek 5.9: Počet průchodů nulou – celý signál.

Funkce jednotlivých částí a jejich signálové modelovy jsou následující:

- **plíce** — zdroj energie — **signály: NIC**
- **hrtan** (larynx) — modulace energie – **signály: buzení (excitation)**.
  - otevřené hlasivky — šum.
  - kmitající hlasivky — periodický signál, typické hodnoty **základního tónu řeči** jsou uvedeny v Tabulce 5.1.
  - Buzení je většinou *smíšené* (moderní kodéry, GSM . . . )
- **hlasový (artikulační) trakt** — modifikující ústrojí — **signály: filtrace**. hlasový trakt se skládá ze hltanu (pharynx), měkkého patra (vélum), jazyka, ústní dutiny, nosní dutiny, zubů z rtů.




Obrázek 5.10: Hlasové ústrojí člověka – s laskavým svolením autora převzato z J. Psutka: Komunikace s počítačem mluvenou řečí, Academia Praha 1995.

### 5.5.1 Model

Model hlasového ústrojí je znázorněn na Obr. 5.11. Jako přenosový systém se používá obyčejný lineární filtr, nejčastěji typu IIR. Chceme-li prostudovat chování jednotlivých komponentů modelu v časové a frekvenční oblasti (Obr. 5.12), je nutné si uvědomit, že výsledný signál je dán v časové oblasti *konvolucí*:

$$s(t) = g(t) \star h(t) = \int_{-\infty}^{+\infty} g(\tau)h(t-\tau)d\tau. \tag{5.16}$$

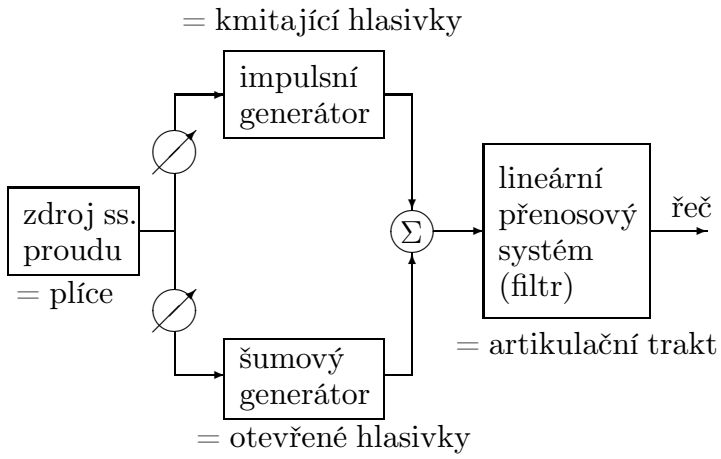
Do kmitočtové oblasti se tato konvoluce promítá jako *součin*:

$$S(f) = G(f)H(f). \tag{5.17}$$

Důležitý (a nelehký) úkol ve zpr. řeči je **dekonvoluce**, kdy se snažíme oddělit vliv buzení a artikulačního traktu.

| | |
|---|---|
| muži | 90–120 Hz |
| ženy | 150–300 Hz |
| děti | 350–400 Hz |

Tabulka 5.1: Typické hodnoty základního tónu řeči $F_0$.




Obrázek 5.11: Model hlasového ústrojí.

## 5.6 Spektrogram

Jedno spektrum nestačí (řeč je nestacionární), proto usilujeme o znázornění průběhu spektra řeči (přesněji spektrální hustoty výkonu – PSD) v čase:

- řeč dělíme na rámce.
- v každém rámci odhadneme PSD, nejčastěji pomocí DFT, a zobrazíme:
  - vodorovná osa čas ("hrubý" čas v rámcích).
  - svislá osa frekvence.
  - stupně šedi nebo barvičky udávají energii.

Podle délky rámce dělíme na:

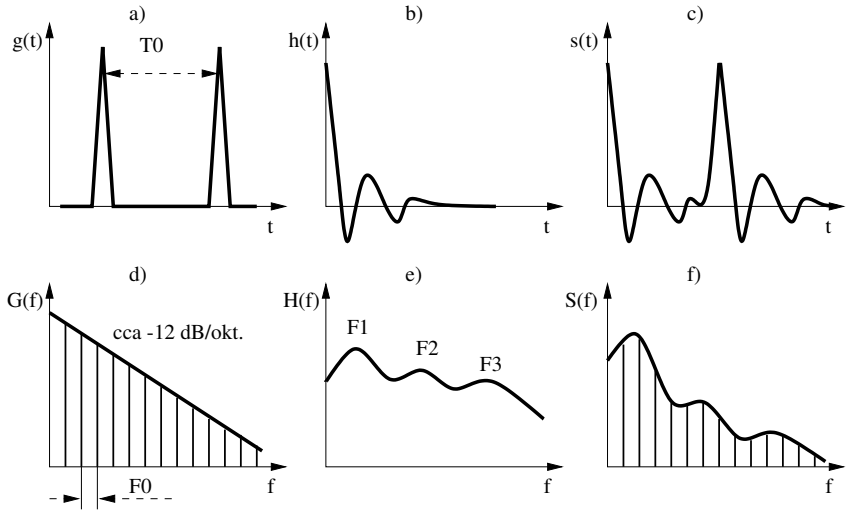
- dlouhodobý (long-term) spektrogram. Příklad výpočtu:
  `specgram(s,256,8000,hamming(256),200);`
- krátkodobý (short-term) spektrogram. Příklad výpočtu:
  `specgram(s,256,8000,hamming(50));`

Nevýhoda DFT tkví v tom, že přesnost ve frekvenci a v čase se nedá získat zároveň, řešením je případně vlnková (wavelet) analysis. [1] Srovnání obou spektrogramů je na Obr. 5.13.

## 5.7 Cepstrum

slouží pro oddělení buzení a modifikace – v kódování se nám s nimi pracuje lépe samostatně, v rozpoznávání buzení zahazujeme úplně (příliš závislé na řečníkovi,

[1] Vlnková analýza je sice zajímavým tématem pro výzkum, ale v žádném state-of-the-art systému nikomu nedala nic lepšího než normální FT...




Obrázek 5.12: Model hlasového ústrojí v časové a frekvenční oblasti: Horní polovina – časové průběhy, spodní polovina – spektra. a) a d) buzení: $T_0$ je perioda, $F_0$ je frekvence základního tónu. b) a e): artikulační trakt: $F_1$ až $F_3$ jsou formanty (rezonanční frekvence hlasového traktu), dány jeho fyzickou konfigurací. c) a f) výsledný signál a jeho spektrum.

náladě, ...). Jak je možné vliv buzení od modifikačního ústrojí oddělit ?

**Návrh č. 1:** odfiltrujeme část pod 400 Hz a zbavíme se základního tónu Není to moc dobrý nápad (viz Obr. 5.14), protože

- násobky základního tónu jsou "rozesety" po celém spektru.
- mohli bychom přijít o první formant.
- telefonní pásmo začíná na 300 Hz a také perfektně slyšíme, s jakou výškou hlasu člověk mluví.

Budeme tedy potřebovat něco lepšího, a řešení nalezneme v **cepstru**. Formulujme problém matematicky: Buzení $e(t)$ je konvoluováno s impulsní odezvou filtru:

$$s(t) = g(t) \star h(t) = \int_{-\infty}^{+\infty} g(\tau)h(t-\tau)d\tau, \tag{5.18}$$

v kmitočtové oblasti *součin*:

$$S(f) = G(f)H(f). \tag{5.19}$$

ani v jedné z oblastí nelze dvě složky dobře oddělit. Řešením bude **nelinearita**, která dokáže převést součin na součet.

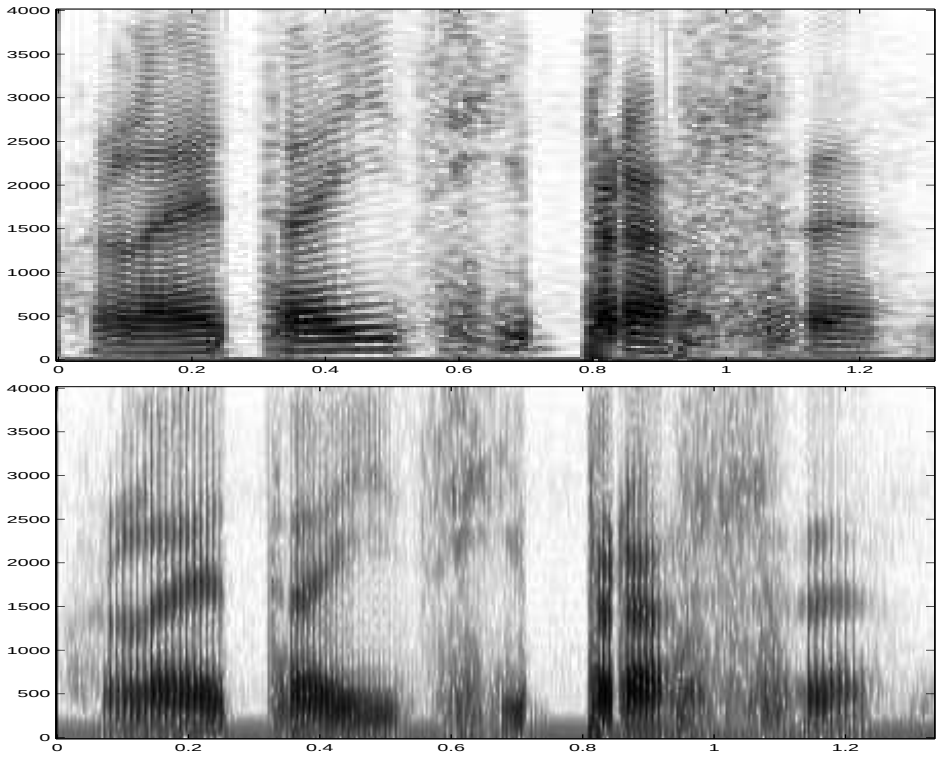
### 5.7.1 Definice cepstra

$$\ln G(f) = \sum_{n=-\infty}^{+\infty} c(n)e^{-j2\pi fn} \tag{5.20}$$

DEF

Hodnoty $c(n)$ jsou **cepstrální koeficienty**. Jelikož $G(f)$ je sudá funkce, jsou $c(n)$ reálné a platí:

$$c(n) = c(-n) \tag{5.21}$$




Obrázek 5.13: Srovnání long- a short-term spektrogramu.

Suma v rovnici je definicí DFT, proto můžeme $c(n)$ vypočítat jako:

$$c(n) = \mathcal{F}^{-1}\left[\ln G(f)\right] \tag{5.22}$$

Vzhledem k tomu, že inversní Fourierova transformace nás vrací zpět do časové oblasti (tedy "anti-spektra"), můžeme si dovolit následující slovní hříčky:

- spectrum $\longrightarrow$ cepstrum.
- frekvence $\longrightarrow$ kvefrence.
- filtrování $\longrightarrow$ liftrování.
- atd.

### 5.7.2 DFT-cepstrum

V tomto případě odhadneme spektrální hustotu výkonu pomocí DFT (4.8):

$$c(n) = \mathcal{F}^{-1}\left\{\ln\left|\mathcal{F}[s(n)]\right|^2\right\}, \tag{5.23}$$

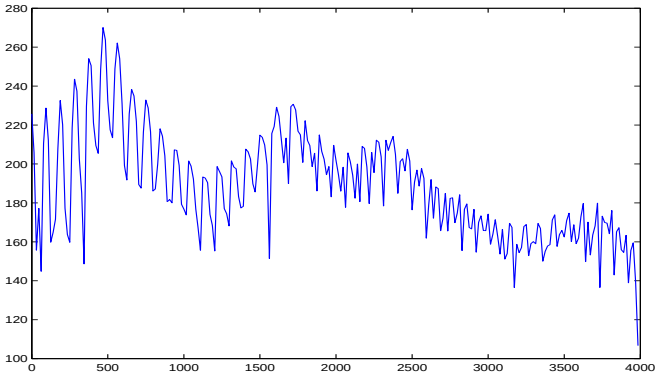
Opravdu však dokáže cepstrum "rozetnout" konvoluci ? Pojďme si to ukázat:

$$s(n) = e(n) \star h(n), \tag{5.24}$$

$$S(f) = E(f)H(f) \;\text{ a tedy }\; |S(f)|^2 = |E(f)|^2\,|H(f)|^2. \tag{5.25}$$

Při výpočtu cepstra využíváme linearity zpětné Fourierovy transformace:

$$\mathcal{F}^{-1}(a+b) = \mathcal{F}^{-1}(a) + \mathcal{F}^{-1}(b).$$



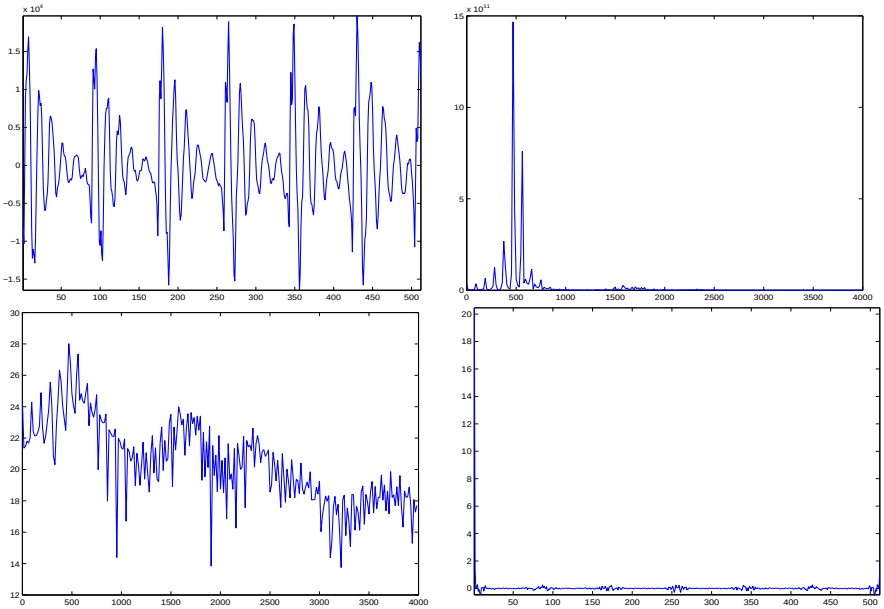

Obrázek 5.14: Spektrum znělého úseku řeči.

Dostáváme:

$$c(n) = \mathcal{F}^{-1}\left\{\ln[|E(f)|^2\,|H(f)|^2]\right\} = \mathcal{F}^{-1}\left\{\ln|E(f)|^2 + \ln|H(f)|^2\right\} \quad (5.26)$$

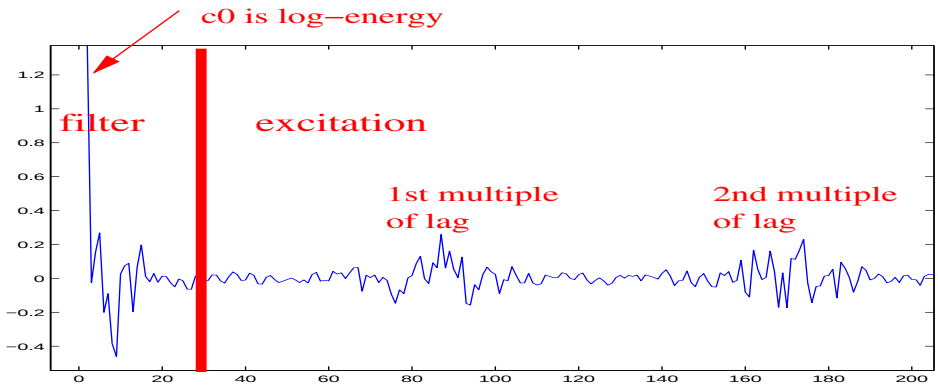
$$= \mathcal{F}^{-1}\left\{\ln|E(f)|^2\right\} + \mathcal{F}^{-1}\left\{\ln|H(f)|^2\right\} = c_e(n) + c_h(n) \quad (5.27)$$

Z konvoluce se stal **součet**. Postup je demonstrován na Obr. 5.15. Pokud jsou koeficienty $c_e(n)$ a $c_h(n)$ odděleny na kvefrenční ose, je možné je separovat jednoduchým oknem — naštěstí **jsou** (Obr. 5.16).



Obrázek 5.15: Ilustrace výpočtu cepstra: signál, $|\mathcal{F}[s(n)]|^2$, $\ln|\mathcal{F}[s(n)]|^2$, $\mathcal{F}^{-1}\left\{\ln|\mathcal{F}[s(n)]|^2\right\}$.

Zajímavé je sledovat, co se stane při zpětné transformaci na signál, pokud vynulujeme část koeficientů patřících buzení nebo filtru (Obrázky 5.17 a 5.18).




Obrázek 5.16: Separace koeficientů: Pro vzorkovací frekvenci $F_s$ =8000 Hz můžeme na kvefrenční ose oddělit vliv buzení a filtru separací s hranicí 30.

## 5.8 Mel-frekvenční cepstrální koeficienty – MFCC

Výše uvedený postup výpočtu cepstra příliš neodpovídá lidskému slyšení:

- DFT má všude stejné frekvenční rozlišení.
- Lidské ucho má na nízkých frekvencích větší rozlišení než na vysokých.
- Pro rozpoznávače řeči chceme přiblížit cepstrum slyšení.

U MFCC postupujeme tak, že na frekvenční osu rozmístíme **nelineárně** filtry. Měříme energii na jejich výstupu, a tuto použijeme místo DFT při výpočtu cepstra.

Při konstrukci filtrů můžeme nelineárně upravit frekvenční osu a na upravené ose pak filtry rozmístit rovnoměrně. Používaná nelineární úprava využívá převodu Hertzů na Mely:

$$F_{Mel} = 2959 \log_{10}(1 + \frac{F_{Hz}}{700}) \tag{5.28}$$

Lineární rozmístění filtrů na Mel-ové ose má za následek nelineární rozmístění na standardní kmitočtové ose v Hz (Obr. 5.19).

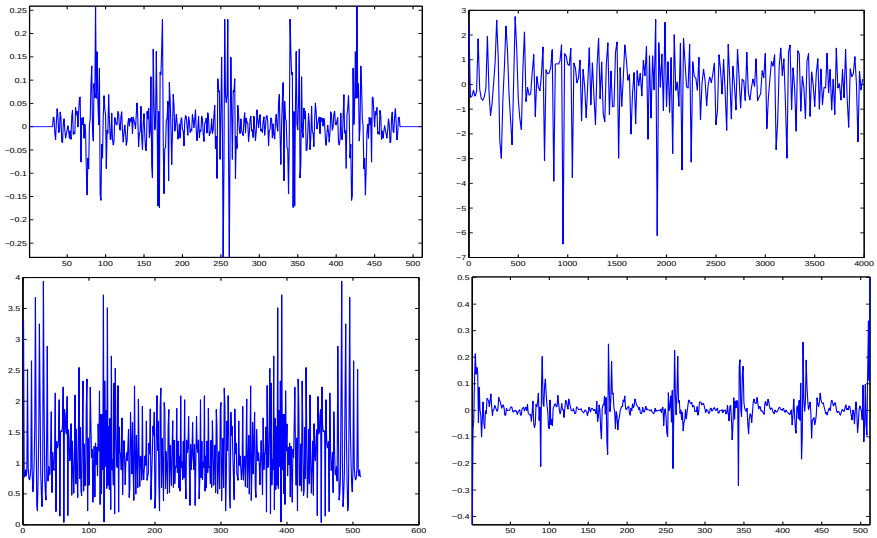
Při výpočet energií z jednotlivých filtrů (frekvenčních pásem) bychom mohli skutečně zkonstruovat banku filtrů, vstupní signál filtrovat v časové oblasti a počítat energii pomocí $\sum_n s_i^2(n)$. Toto by však bylo příliš složité, proto využijeme DFT, umocníme, vynásobíme trojúhelníkovým oknem a sečteme. Tento postup je mj. použit ve standardním toolkitu pro rozpoznávání řeči HTK Hidden Markov Model ToolKit z University of Cambridge.

Zbývá provést zpětnou FT logaritmu těchto energií. Zpětnou FT můžeme realizovat pomocí diskrétní cosinové transformace (DCT), která nahrazuje inverzní FT (bez odvození: využíváme symetrie spektra a toho, že výsledek musí vyjít reálný):

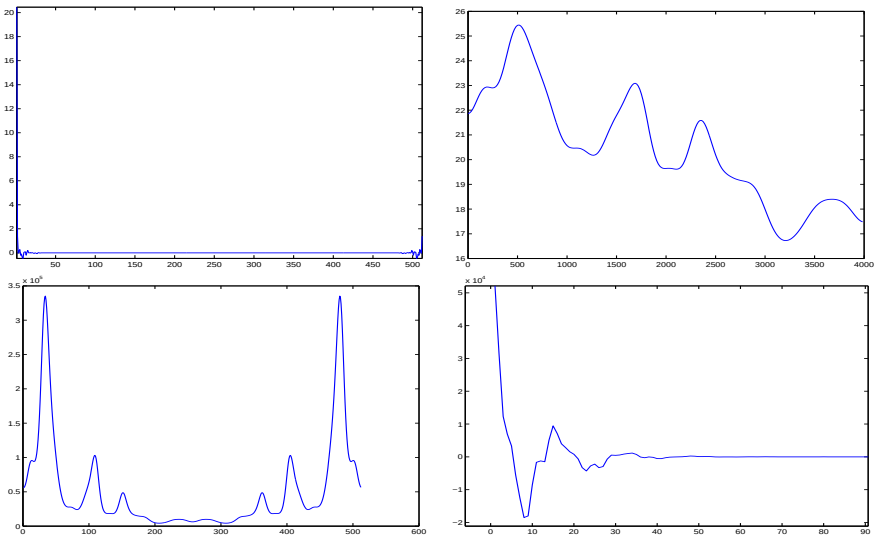
$$c_{mf}(n) = \sum_{i=1}^{K} \log m_k \cos\left[n(k-0.5)\frac{\pi}{K}\right] \tag{5.29}$$

Výsledkem jsou **Mel-frekvenční cepstrální koeficienty (MFCC)**. Obrázek 5.20 shrnuje postup jejich výpočtu do blokového schématu, Obr. 5.21 pak presentuje výsledky jednotlivých mezikroků.

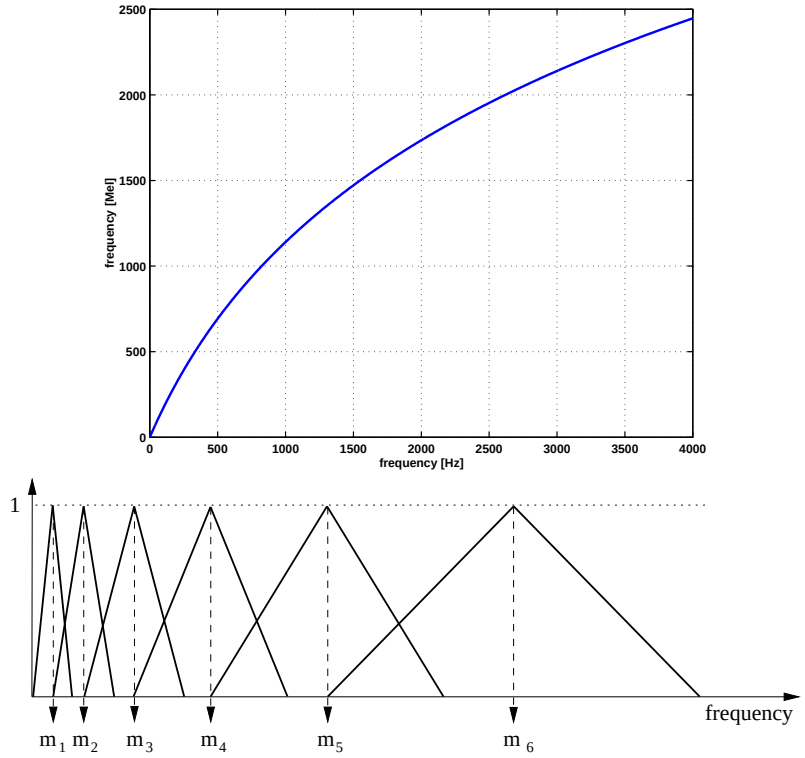
DEF



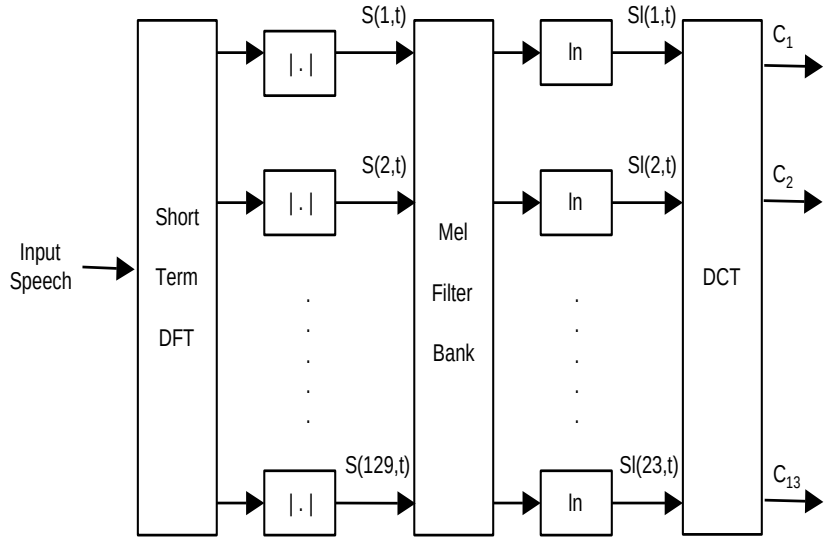
Obrázek 5.17: Vynulované koeficienty patřící filtru u cepstra: cepstrum, $\ln|\mathcal{F}[s(n)]|^2$, $|\mathcal{F}[s(n)]|$, signál (při IDFT byly použity fáze původního signálu).



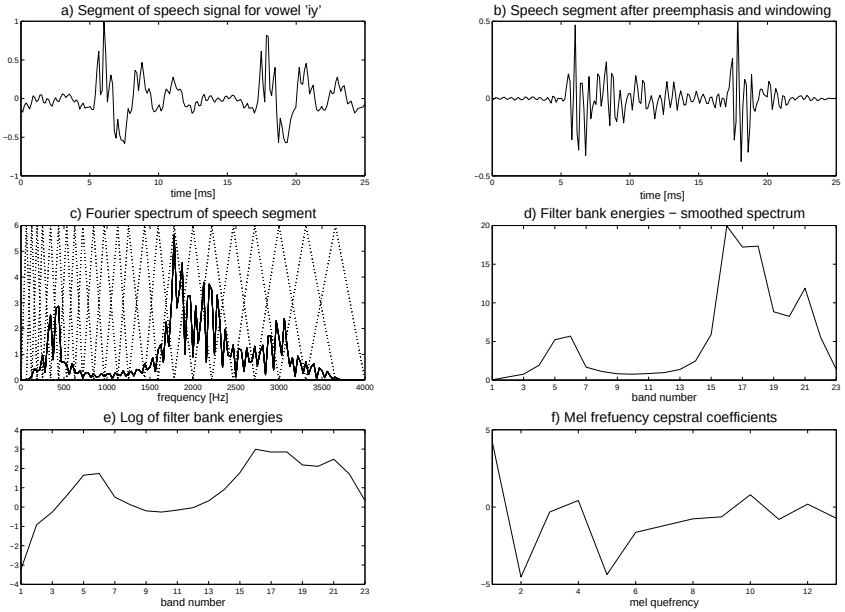
Obrázek 5.18: Vynulované koeficienty patřící buzení u cepstra: cepstrum, $\ln|\mathcal{F}[s(n)]|^2$, $|\mathcal{F}[s(n)]|$, signál (při IDFT byly použity fáze nula).




Obrázek 5.19: Nelineární úprava kmitočtové osy a výsledné rozmístěná filtrů.




Obrázek 5.20: Výpočet MFCC - blokové schema.




Obrázek 5.21: Výpočet MFCC: a) původní signál, b) preemfázovaný signál, c) DFT spektrum a jeho váhování filtry, d) energie na výstupech jednotlivých filtrů, e) log této energie, f) MFCC.

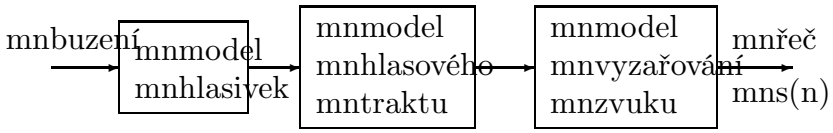
# Kapitola 6

# Lineární predikce – LPC


V minulé kapitole (Obr. 5.11) jsme definovali, jak budeme modelovat hlasové ústrojí. Tato kapitola se zabývá odhadem parametrů **filtru**, který modeluje artikulační trakt.

## 6.1 Model hlasového traktu

Nejprve je nutné určit, jak by měl takový filtr vypadat. K tomu nám pomůže analýza jednotlivých částí hlasového traktu na Obr. 6.1. Opakuji, že v této kapitole se nevěnujeme buzení; na všechny komponenty budeme nahlížet jako na filtry.



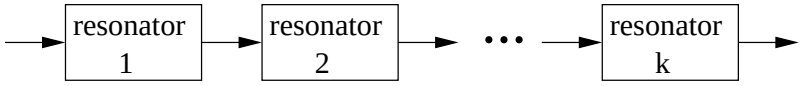

Obrázek 6.1: Komponenty hlasového traktu.

- **Hlasivky** jsou dolní propusť 2. řádu, lomová frekvence okolo 100 Hz, jejich přenosová funkce se dá aproximovat jako:

$$G(z) = \frac{1}{[1 - e^{-cT_s} z^{-1}]^2}, \tag{6.1}$$

  kde $T_s$ je časová konstanta.

- **Hlasový trakt** je kaskáda malých dvojpólových *rezonátorů* odpovídajících *formantům*[1], viz. Obr. 6.2.




Obrázek 6.2: Kaskáda rezonátorů modelujících hlasový trakt.

  Pro $k$ formantů $F_i$ s šířkami pásem $B_i$ je možné takovou kaskádu zapsat přenosovou funkcí:

$$V(z) = \frac{1}{\prod_{i=1}^{K} [1 - 2e^{-\alpha_i T_s} \cos \beta_i T_s z^{-1} + e^{-2\alpha_i T_s} z^{-2}]}, \tag{6.2}$$

  kde parametry $\alpha_i$ a $\beta_i$ jsou určeny polohou a šířkou pásma formantů.

[1] Formanty jsou rezonanční frekvence našeho hlasového traktu.

- **Model vyzařování zvuku** z úst má formu jednoduché horní propusti:

$$L(z) = 1 - z^{-1} \tag{6.3}$$

Sepíšeme-li tyto tři komponenty dohromady (kaskádní zapojení filtrů odpovídá prostému násobení přenosových funkcí), dostaneme:

$$\begin{aligned} H(z) &= G(z)V(z)L(z) = \\ &= \frac{1 - z^{-1}}{(1 - e^{-cT_s}z^{-1})^2 \prod_{i=1}^{K}[1 - 2e^{-\alpha_i T_s}\cos\beta_i T_s z^{-1} + e^{-2\alpha_i T_s}z^{-2}]} \end{aligned} \tag{6.4}$$

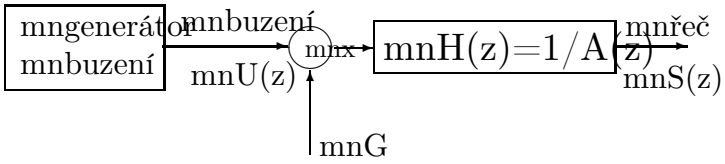
Člen $cT_s \to 0$, proto můžeme krátit čitatele i jmenovatele o jeden člen $1 - z^{-1}$. Celkový model je tedy **celopólový** (obsahuje jen jmenovatele – čistý IIR filtr). Běžný zápis je:

$$H(z) = \frac{1}{1 + \sum_{i=1}^{P} a_i z^{-i}} = \frac{1}{A(z)}, \tag{6.5}$$

kde polynom $A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_P z^{-P}$ má řád $P = 2k + 1$ ($k$ je počet formantů). Za užitečný počet pokládáme $k$=4 či 5, proto volíme často $P$=10 (pro $F_s$=8 kHz). Pro vyšší vzorkovací frekvence volíme $P$ vyšší (např. 16), abychom postihli i vysokofrekvenční část spektra.

## 6.2 Určení parametrů modelu pomocí lineární predikce (LP)

Namalujeme-li podrobněji schéma modelu tvorby řeči, dostáváme Obr. 6.3.



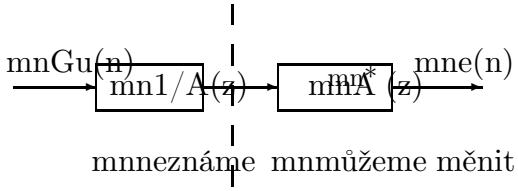

Obrázek 6.3: Model tvorby řeči.

$n$-tý vzorek řeči je tedy dán:

$$s(n) = Gu(n) - \sum_{i=1}^{P} a_i s(n-i) \tag{6.6}$$

Parametry (koeficienty) filtru $a_i$ jsou ovšem **neznámé** a musíme je **odhadnout**, odborně **identifikovat** (system identification):

Odhad začneme tak, že zkonstruujeme *inverzní filtr* $A^\star(z)$ s koeficienty $\alpha_i$ (Obr. 6.4). Ukazuje se, že v případě stacionárního signálu $s(n)$ jsou koeficienty $a_i$ *identifikovány* pomocí koeficientů $\alpha_i$, je-li *minimalizována energie signálu na výstupu* $e(n)$: $\mathcal{E}\{e^2(n)\}$. Zjednodušeně řečeno: "kroutíme parametry filtru tak dlouho, dokud není energie signálu na výstupu minimální. . . "




Obrázek 6.4: Inverzní filtr.

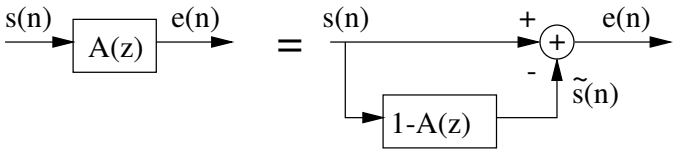
### 6.2.1 Proč "lineární predikce" ?

Předpokládáme, že $\mathcal{E}\{e^2(n)\}$ je již minimalizována, tedy že $A^\star(z) = A(z)$ a budeme tedy používat pouze označení koeficientů $a_i$.

$A(z)$ můžeme (trochu podivně – jako když se levou rukou drbete za pravým uchem...) zapsat jako:

$$A(z) = 1 - [1 - A(z)] \tag{6.7}$$

viz Obr. 6.5




Obrázek 6.5: Ilustrace $A(z) = 1 - [1 - A(z)]$.

Signál $\tilde{s}(n)$ je dán lineární kombinací několika předchozích vzorků, považujeme jej za *předpověď* skutečného vzorku $s(n)$:

$$\tilde{s}(n) = -\sum_{i=1}^{P} a_i s(n-i) \tag{6.8}$$

**Chyba predikce** je dána jako rozdíl skutečné a předpovězené hodnoty:

DEF

$$e(n) = s(n) - \tilde{s}(n) = s(n) - [-\sum_{i=1}^{P} a_i s(n-i)] = s(n) + \sum_{i=1}^{P} a_i s(n-i). \tag{6.9}$$

a platí samozřejmě, že čím lepší je predikce, tím menší je chyba. Pokud (6.9) přepíšeme v rovině $z$, dostáváme jednoduchý vztah:

$$E(z) = S(z)A(z) \tag{6.10}$$

Výhody získání parametrů touto metodou jsou následující:

- je-li $\alpha_i = a_i$, je chyba predikce rovna *buzení* (můžeme se tedy dostat ke vstupu do hlasového traktu bez skalpelu).
- určení koeficientů pomocí LP vede k soustavě snadno řešitelných lineárních rovnic.

### 6.2.2 Minimalisace energie chyby – řešení

V této etapě řešení zatím záměrně nezmiňujeme, kolik vzorků vstupního signálu máme k disposici, sumy jsou tedy zatím bez mezí. Nenormalizovaná energie chyby predikce je dána:

$$E = \sum_n e^2(n) \tag{6.11}$$

Tento výraz je třeba minimalizovat. Vyjádříme jej pomocí signálu $s(n)$ (známá veličina) a neznámých koeficientů $a_i$. Pro nalezení minima budeme parciálně derivovat podle každého $a_i$, derivace položíme rovny nule:

$$\frac{\delta}{\delta a_j}\left\{\sum_n [s(n) + \sum_{i=1}^{P} a_i s(n-i)]^2\right\} = 0 \tag{6.12}$$

$$\sum_n 2[s(n) + \sum_{i=1}^{P} a_i s(n-i)]s(n-j) = 0 \tag{6.13}$$

$$\sum_n s(n)s(n-j) + \sum_{i=1}^{P} a_i \sum_n s(n-i)s(n-j) = 0. \tag{6.14}$$

Označíme-li

$$\sum_n s(n-i)s(n-j) = \phi(i,j), \tag{6.15}$$

pak

$$\sum_{i=1}^{P} a_i \phi(i,j) = -\phi(0,j) \quad \text{pro} \quad 1 \le j \le P, \tag{6.16}$$

což je soustava lineárních rovnic:

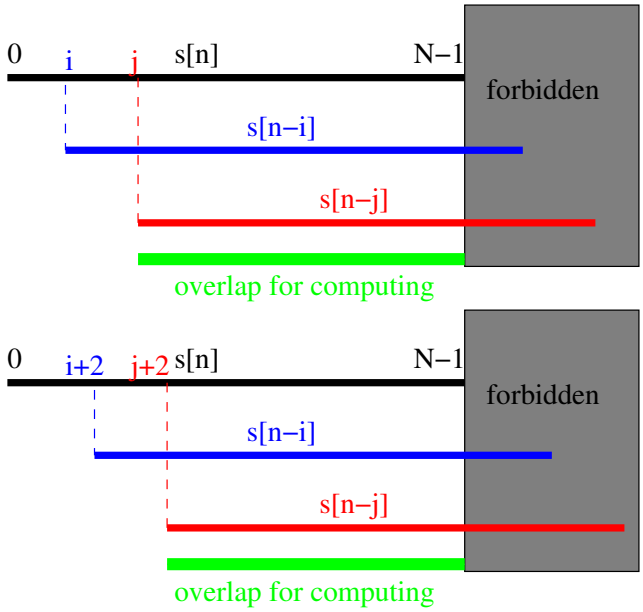
$$\begin{array}{lll} \phi(1,1)a_1 + \phi(2,1)a_2 + & \cdots & +\phi(P,1)a_P = -\phi(0,1) \\ \phi(1,2)a_1 + \phi(2,2)a_2 + & \cdots & +\phi(P,2)a_P = -\phi(0,2) \\ & \vdots & \\ \phi(1,P)a_1 + \phi(2,P)a_2 + & \cdots & +\phi(P,P)a_P = -\phi(0,P), \end{array} \tag{6.17}$$

### 6.2.3 Výpočet $\phi(\cdot,\cdot)$

Koeficienty odhadujeme na rámci o délce $N$ vzorků. Existují dvě metody lišící se v tom, jak nahlížíme na signál vně rámce (tedy pro vzorky $n < 0$ a $n > N-1$):

1. **Kovarianční metoda:** signál vně rámce je **neznámý**: se vzorky mimo $[0, N-1]$ nemohu počítat, ani když je tam signál zpožděn (viz Obr. 6.6). Důsledkem je to, že hodnoty $\phi(i,j)$ a $\phi(i+const, j+const)$ **nejsou** stejné, protože máme pokaždé jiný počet vzorků. V tomto případě se musí řešit plná soustava lineárních rovnic, což je složité. Kovarianční metoda navíc vede k **nestabilnímu** filtru $1/A(z)$.


2. **Korelační metoda:** signál vně rámce je považován za **známý, ale nulový** – mohu s ním počítat (Obr. 6.7). Tato metoda vede ke **stejným** koeficientům $\phi(i,j)$ a $\phi(i+const, j+const)$ (počítáme pokaždé ze stejných vzorků), se kterými se nám bude soustava rovnic dobře počítat, protože na diagonálách budou stejné hodnoty: např. $\phi(2,1) = \phi(3,2) = \phi(4,3) = \cdots$




Obrázek 6.6: Kovarianční metoda – signál vně rámce je neznámý a nesmíme jej použít.

### 6.2.4 Proč jsou $\phi$ autokorelační koeficienty


Odhad autokorelačních koeficientů (bez normalizace) pro signál o délce $N$ pro kladné $k$, viz (Signály a systémy, Náhodné procesy II.: `http://www.fit.vutbr.cz/~cernocky/sig`), je:

$$R(k) = \sum_{n=0}^{N-1-k} s(n)s(n+k)$$

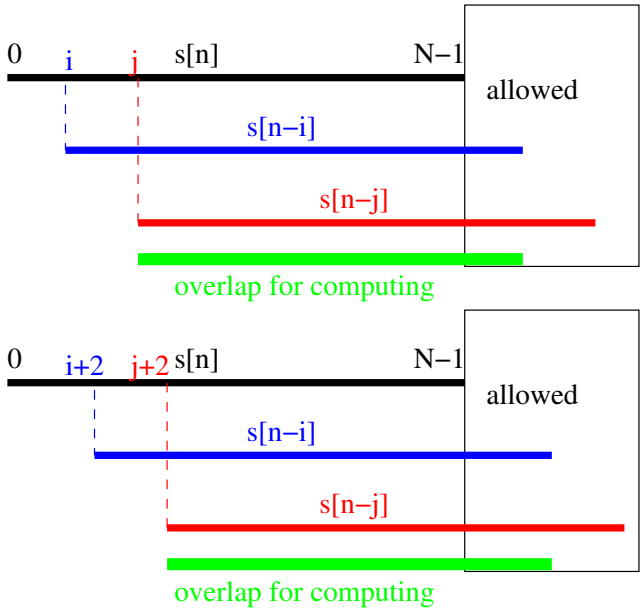
Korelační koeficienty "udávají podobnost signálu samotného se sebou, když ho posuneme o $k$ vzorků" (viz Obr. 6.8).

Situace pro $\phi(i,j)$ a $\phi(j,i)$ je znázorněna na Obr. 6.9. Jelikož pokaždé počítáme se stejnými vzorku, jsou oba dva jsou rovny autokorelačnímu koeficientu $R(|i-j|)$. To je fajn, protože matice bude navíc ještě symetrická. Matici, která je symetrická a má na diagonálách stejné prvky, se říká **Töplitzova**.

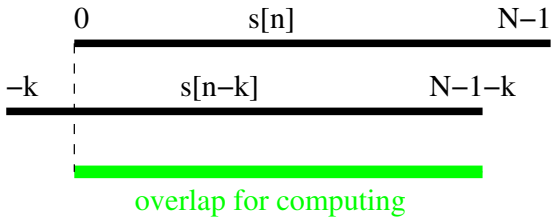
### 6.2.5 Výsledná soustava rovnic

pro koeficienty $a_1 \ldots a_P$ vypadá následovně:

$$\begin{array}{lcl} R(0)a_1 + R(1)a_2 + & \cdots & +R(P-1)a_P = -R(1) \\ R(1)a_1 + R(0)a_2 + & \cdots & +R(P-2)a_P = -R(2) \\ & \vdots & \\ R(P-1)a_1 + R(P-2)a_2 + & \cdots & +R(0)a_P = -R(P), \end{array} \tag{6.18}$$




Obrázek 6.7: Korelační metoda – signál vně rámce je známý, ale nulový, mohu jej použít.




Obrázek 6.8: Definice autokorelačního koeficientu.

kterou můžeme zapsat maticově:

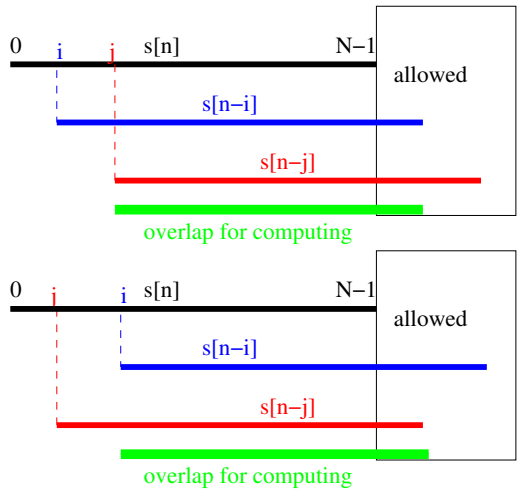
$$\begin{bmatrix} R(0) & R(1) & \cdots & R(P-1) \\ R(1) & R(0) & \cdots & R(P-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(P-1) & R(P-2) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} -R(1) \\ -R(2) \\ \vdots \\ -R(P) \end{bmatrix}. \quad (6.19)$$

### 6.2.6 Energie chyby predikce

DEF

Pomocí LPC můžeme dostat i následující vzoreček pro výpočet **nenormalizované energie chyby predikce**:

$$E = \sum_{n=0}^{N+P-1} e^2(n) = R(0) + \sum_{i=1}^{P} a_i R(i) \quad (6.20)$$




Obrázek 6.9: Ilustrace, proč je v korelační metodě $\phi(i,j)$ a $\phi(j,i)$ ten samý autokorelační koeficient $R(|i-j|)$.

Pokud má budící signál *normovanou energii* rovnu 1 — např. bílý šum s rozptylem 1 nebo pulsy, kde $\frac{1}{N}\sum_{n=0}^{N-1} u^2(n) = 1$, pak, abychom dostali tutéž energii jako $s(n)$, musíme nastavit gain (zesílení) filtru na:

$$G^2 = \frac{E}{N} = \frac{1}{N}\left[R(0) + \sum_{i=1}^{P} a_i R(i)\right]. \tag{6.21}$$

Toto se nám bude hodit při kódování.

### 6.2.7 Levinson–Durbin

Jelikož je matice $\mathbf{R}$ symetrická a Töplitzova (všechny prvky na diagonálách jsou stejné), dá se k řešení soustavy 6.18 použít rychlý algoritmus Levinsona a Durbina:

$$E^{(0)} = R(0) \tag{6.22}$$

$$k_i = -\left[R(i) + \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j)\right] / E^{(i-1)} \tag{6.23}$$

$$a_i^{(i)} = k_i \tag{6.24}$$

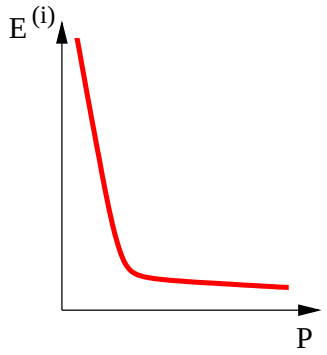
$$a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)} \quad \text{pro} \quad 1 \le j \le i-1 \tag{6.25}$$

$$E^{(i)} = (1-k_i^2)E^{(i-1)} \tag{6.26}$$

Algoritmus probíhá prakticky tak, že postupně zvyšujeme řád prediktoru (sloupce následující tabulky). $a_j^{(i)}$ je $j$-tý koeficient prediktoru řádu $i$:

$$\begin{array}{ccccc} a_1^{(1)} & a_1^{(2)} & a_1^{(3)} & \cdots & a_1^{(P)} \\ & a_2^{(2)} & a_2^{(3)} & \cdots & a_2^{(P)} \\ & & a_3^{(3)} & \cdots & a_3^{(P)} \\ & & & \ddots & \vdots \\ & & & & a_P^{(P)} \end{array} \tag{6.27}$$

Pokud se nespokojíme s klasickými hodnotami $P = 10$ nebo $P = 16$, můžeme se pokusit určit optimální řád prediktoru. Budeme vynášet průběhu energie chyby predikce $E(i)$ (viz 6.20) v závislosti na řádu prediktoru (Obr. 6.10) a zastavíme se, až se energie přestane výrazně snižovat. Zvyšování řádu prediktoru nad "lom" funkce již nepřináší téměř žádné zlepšení energie chyby.




Obrázek 6.10: Určení řádu prediktoru pomocí energie chyby predikce.

## 6.3 Odhad spektrální hustoty výkonu (PSD) pomocí modelu LPC

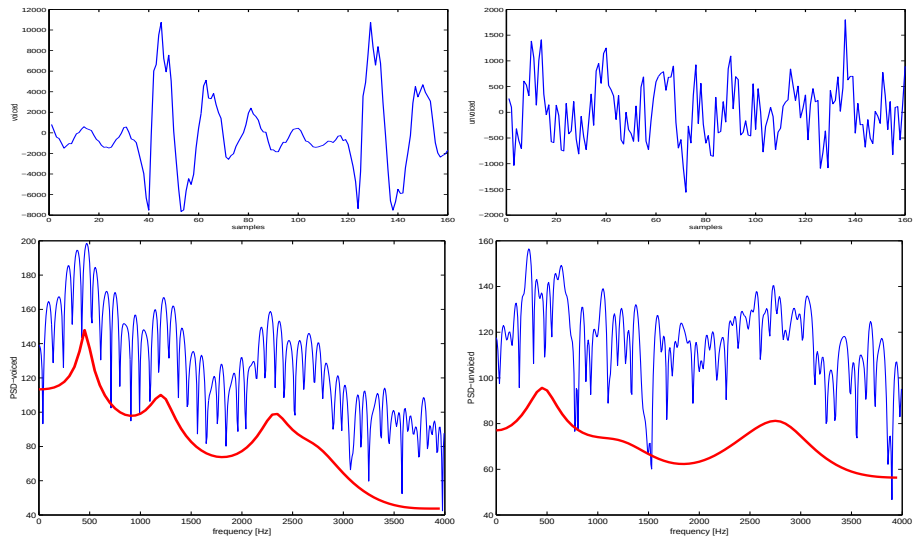
Prozatím jsme PSD odhadovali pomocí DFT (4.8), ta ale obsahovala i "jemnou" složku způsobenou násobky frekvence základního tónu. PSD se však dá odhadnout i pomocí frekvenční charakteristiky filtru $1/A(z)$ pomocí standardního přechodu z přenosové funkce na kmitočtovou charakteristiku:

$$\hat{G}_{LPC} = \left| \frac{G}{A(z)} \right|^2_{z=e^{j2\pi f}}, \tag{6.28}$$

kde $f$ je normovaná frekvence $f = \dfrac{F}{F_s}$. Po dosazení:

$$\hat{G}_{LPC} = \frac{G^2}{\left| 1 + \sum_{i=1}^{P} a_i e^{-j2\pi fi} \right|^2} \tag{6.29}$$

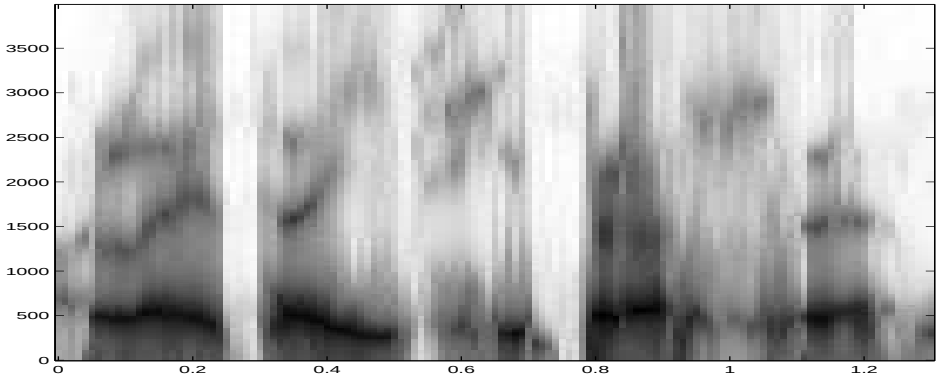
Na této spektrální hustotě výkonu se dají dobře rozeznat formanty, protože je eliminován vliv základního tónu (a tedy jemná struktura spektra, která se opakuje




Obrázek 6.11: Spektrální hustota energie získaná pomocí DFT (modrá) a LPC (červená). Vlevo je znělý rámec, vpravo neznělý.

s $F_0$). Obr. 6.11 uvádí příklad odhadu spektrální hustoty získaný pomocí DFT a LPC na znělém a neznělém rámci.

Pomocí LPC můžeme samozřejmě vygenerovat i spektrogram. Srovnejte dlouhodobé a krátkodobé spektrogramy z minulé kapitoly (Obr. 5.12) s LPC-spektrogramem na Obr. 6.12.



Obrázek 6.12: LPC spektrogram.

## 6.4 Parametry odvozené z LPC koeficientů

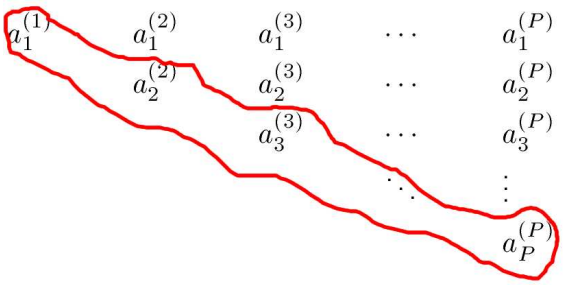
Proč nám nestačí koeficienty filtru $a_i$ ? Jsou dobré na filtrování, ale to je tak všechno, mají následující nevýhody:

- Špatně se kvantují (velká citlivost stability filtru na přesnost kvantizace, nejsou omezeny: $a_i \in < -\infty, +\infty >$.)

- Jsou tvrdě korelovány – špatné pro rozpoznávání založené na HMM.
- Vzdálenost dvou vektorů koeficientů $a_i$ nemá nic společného s podobností nebo nepodobností řečových rámců – špatné i pro rozpoznávání založené na přímém srovnávání parametrů (DTW).

### 6.4.1 PARCOR

*koeficienty odrazu* nebo také *koeficienty PARCOR* (partial correlation). jsou mezivýsledky v algoritmu Levinsona-Durbina: koeficienty $k_i = a_i^{(i)}$ – viz Obr. 6.13. Platí pro ně: $k_i \in < -1, 1 >$, jsou tedy vhodnější pro kódování než $a_i$. Sady koeficientů $a_i$ a $k_i$ se jedna na druhou dají jednoduše převést. Dokázali byste určit, jak ?




Obrázek 6.13: PARCOR koeficienty jako produkt algoritmu Levinsona-Durbina.

### 6.4.2 Válcový model hlasového traktu a jeho parametry

hlasový trakt můžeme fyzikálně modelovat pomocí válcových sekcí o stejné délce, avšak o různých průměrech (a tím i průřezech), viz Obr. 6.14. Poměr sousedních sekcí se dá zapsat pomocí PARCOR koeficientů:

$$\frac{A_{m-1}}{A_m} = \frac{1 + k_m}{1 - k_m} \tag{6.30}$$

pro $m = P, P-1, \ldots, 1$. Plocha $A_P$ je fiktivní – skutečnou velikost neznáme, položíme tedy $A_P = 1$. Hodnoty $\frac{A_{m-1}}{A_m}$ jsou pak poměry ploch – area ratios (AR). Používanější jsou logaritmické poměry ploch – log area ratios (LAR):

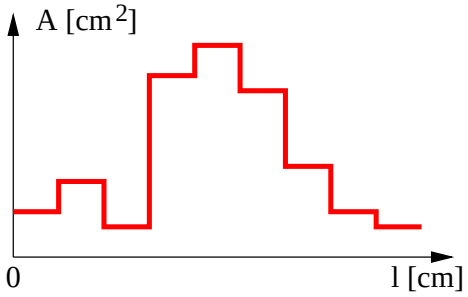
$$g_m = \log \frac{A_{m-1}}{A_m} = \log \frac{1 + k_m}{1 - k_m} \tag{6.31}$$

Výhoda $g_m$ oproti $k_i$ je v lineární citlivosti spektra na hodnoty. Je možné použít lineární kvantifikátor hodnot $g_m$. U $k_i$ je spektrum silně citlivé na hodnoty $k_i \to 0$.

### 6.4.3 Line spectral frequencies (LSF) a Line spectral pairs (LSP)

jsou jsou odvozeny z kořenů dvou polynomů:

$$\begin{aligned} M(z) &= A(z) - z^{-(P+1)} A(z^{-1}) \\ Q(z) &= A(z) + z^{-(P+1)} A(z^{-1}). \end{aligned} \tag{6.32}$$




Obrázek 6.14: Válcový model hlasového traktu.

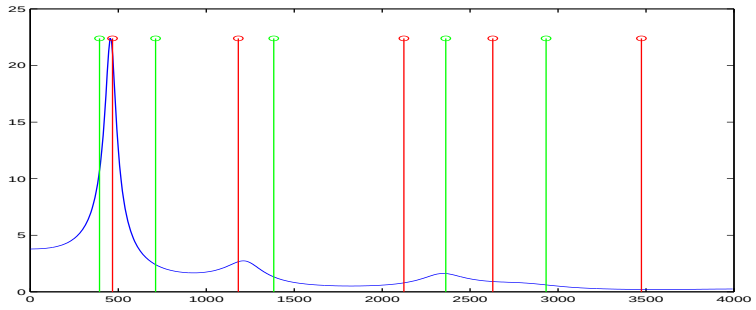
Pomocí kořenů se dají zapsat:

$$\begin{aligned} M(z) &= (1-z^{-1}) \prod_{i=2,4,\ldots,P} (1-2z^{-1}\cos\omega_i + z^{-2}) \\ Q(z) &= (1+z^{-1}) \prod_{i=1,3,\ldots,P-1} (1-2z^{-1}\cos\omega_i + z^{-2}). \end{aligned} \tag{6.33}$$

kde $\omega$ je normovaná kruhová frekvence $\omega = 2\pi f$ ($f$ je normovaná "obyčejná" frekvence). Line spectral frequencies $f_i$ jsou v intervalu (0,0.5) a jsou seřazeny vzestupně:

$$0 < f_1 < f_2 < \ldots < f_{P-1} < f_P < \frac{1}{2}. \tag{6.34}$$

Použijeme-li LSFs pro přenos (jsou kvantovány), můžeme v dekodéru otestovat správné dekódování tak, že zkontrolujeme jejich řazení. Příklad LSF je na Obr. 6.15




Obrázek 6.15: Příklad line spectral frequencies.

## 6.5 LPC-cepstrum

V minulé kapitole jsme viděli, jak se dá spektrální hustota výkonu ve výpočtu cepstra (5.22) odhadnout pomocí DFT (5.23). Nic nám ale nebrání odhadnout spektrální hustotu výkonu pomocí LPC-odhadu:

$$\hat{G}_{LPC}(f) = \left| \frac{G}{A(z)} \right|^2_{z=e^{j2\pi f}}, \tag{6.35}$$

kde $G$ je gain "syntezačního" filtru a $A(z)$ je polynom řádu $P$. V tomto případě hovoříme o LPC-cepstru (LPCC):

$$c(n) = \mathcal{F}^{-1}[\ln \hat{G}_{LPC}(f)] \tag{6.36}$$

Dají se odvodit následující vlastnosti LPC-cepstrálních koeficientů:

$$c(0) = \ln G^2. \tag{6.37}$$

Nultý LPC-cepstrální koeficient tedy nese informaci o *energii* daného řečového rámce. Další koeficienty lze vypočítat z LPC koeficientů pomocí rekurentních vztahů:

$$\begin{aligned} c(n) &= -a_n - \frac{1}{n}\sum_{k=1}^{n-1} k c_k a_{n-k} \quad \text{pro} \quad 1 \leq n \leq P \\ c(n) &= -\frac{1}{n}\sum_{k=1}^{n-1} k c_k a_{n-k} \qquad\quad \text{pro} \qquad n > P \end{aligned} \tag{6.38}$$

převod je tedy velmi jednoduchý.

### 6.5.1 Použití LPCC koeficientů

1. LPCC koeficienty jsou jednou z parametrisací používaných v *rozpoznávačích řeči*. Jejich výhodou je, že jednotlivé koeficienty jsou méně korelovány než například LPC koeficienty $a_i$, v rozpoznávačích postavených na skrytých Markovových modelech (hidden Markov models – HMM) se obejdeme bez plných kovariančních matic $\mathbf{\Sigma}$ a vystačíme s vektory rozptylů. Více v kapitole o rozpoznávání pomocí HMM.
2. pomocí dvou sad LPCC koeficientů můžeme jednoduše spočítat logaritmickou spektrální vzdálenost (logarithmic spectral distance) mezi dvěma řečovými rámci – nepříjemná definice s integrálem přejde na prostou Euklidovu vzdálenost dvou vektorů LPCC. Více v sekci o vyhodnocování kvality kódování 8.1.5.

# Kapitola 7

# Určování základního tónu řeči


V minulé kapitole jsme probrali odhad koeficientů filtru, který modeluje artikulační trakt. Tato kapitola se zabývá **odhadem frekvence základního tónu**, na které kmitají hlasivky. Druhým úkolem je určení znělosti, tedy rozhodnutí, **zda** hlasivky vůbec kmitají.

## 7.1 Úvod


Základní terminologie:

- Frekvence základního tónu je základním kmitočtem, na kterém kmitají hlasivky: $F_0$, anglický název *pitch*.
- Periodu základního tónu *(pitch period)* spočítáme jako převrácenou hodnotu frekvence: $T_0 = \frac{1}{F_0}$.
- Jako *lag* označujeme periodu základního tónu vyjádřenou ve vzorcích: $L = T_0 F_s$, kde $F_s$ je vzorkovací frekvence.

### 7.1.1 Využití základního tónu

je následující:

- **syntezátory řeči** – generování melodie.
- **kódování**
  - v jednoduchém kódování označovaném jako LPC se zmenšení bitového toku dosáhne tak, že se samostatně přenáší parametry artikulačního ústrojí (např. koeficienty predikčního filtru $a_i$ nebo odvozené), energie, příznak znělý/neznělý a $F_0$.
  - v modernějších kodérech (např. v RPE-LTP nebo ACELP pro mobilní telefony GSM) se využívá **dlouhodobého prediktoru LTP** (long time predictor). Jedná se o filtr s "dlouhou" impulsní odezvou, která však obsahuje jen jeden nebo několik nenulových prvků. Dosáhneme tak většího "vybělení" signálu.

### 7.1.2 Charakteristiky základního tónu

- $F_0$ může nabývat hodnot od 50 Hz (muži) až do 400 Hz (děti), při $F_s$=8000 Hz tyto frekvence odpovídají lagům $L$=160 až 20 vzorků. Je patrné, že při malých hodnotách $F_0$ se blížíme délkám běžně používaných oken (20 ms, což odpovídá 160 vzorkům).
- kolísání u jednoho mluvčího může být až v poměru 2:1.

- pro různé hlásky mívá základní tón typické průběhy, malé změny po prvním kmitu ($\Delta F_0 < 10$ Hz) charakterizují mluvčího, ale obtížně se zjišťují. V radiotechnice se těmto malým posuvům říká "jitter".
- $F_0$ je ovlivněn *vším* – větnou melodií, náladou, únavou, atd. Velikosti změn $F_0$ jsou větší (větší "modulování" hlasu) u profesionálních mluvčích, obyčejní lidé mluví monotónněji.

### 7.1.3 Problémy určování základního tónu

- ani znělé hlásky nejsou zcela periodické. Čistě periodický může být pouze velmi čistý zpěv. Při generování řeči s $F_0$=konst. je výsledná řeč monotónní.
- nevyskytuje se čistě znělé nebo neznělé buzení. Většinou je buzení smíšené (šum na vyšších frekvencích).
- při nízké energii signálu je určení znělosti a základního tónu obtížné.
- vysoký $F_0$ může být ovlivněn nízkým formantem $F_1$ (ženy, děti).
- při přenosu řeči v telefonním pásmu (300–3400 Hz) nemáme k dispozici základní kmitočet $F_0$, pouze násobky (vyšší harmonické). Filtrace za účelem získání $F_0$ by tedy k ničemu nevedla...

## 7.2 Metody pro určování základního tónu

se dělí do skupin které budou probrány v následujících sekcích:

- autokorelační metody, které pracují buď s prostou autokorelační funkcí nebo s normalizovanou cross-korelační funkcí (NCCF). Korelace se dá aplikovat na původní signál, na tzv. klipovaný signál a na chybu lineární predikce.
- využití prediktoru chyby lineární predikce.
- cepstrální metoda.

Na výstupech těchto metod lze aplikovat některé techniky pro zlepšení či opravy (viz závěr této kapitoly).

### 7.2.1 Autokorelační funkce – ACF (Autocorrelation function)

Je definována (jak jsme již viděli základech zpracování signálů a v minulé kapitole o LPC):

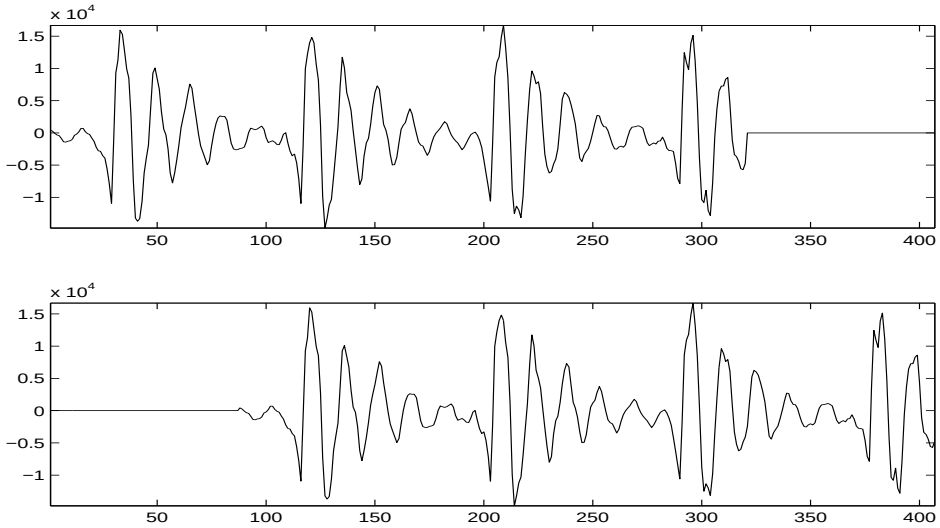
DEF

$$R(m) = \sum_{n=0}^{N-1-m} s(n)s(n+m) \tag{7.1}$$

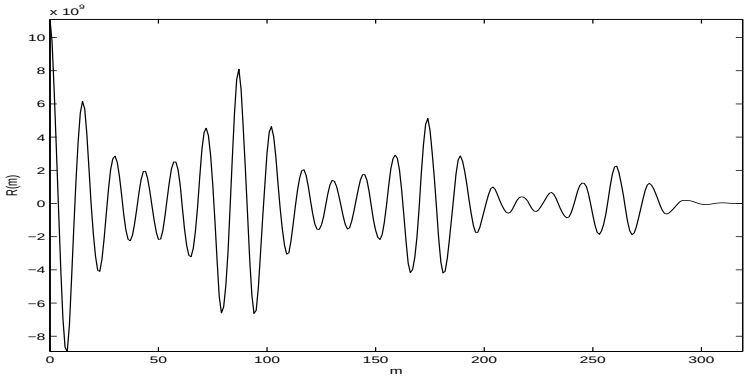
S využitím symetrie autokorelačních koeficientů:

$$R(m) = \sum_{n=m}^{N-1} s(n)s(n-m) \tag{7.2}$$

Na Obr. 7.1 je ilustrován jeden rámec řeči se jeho posunem. Na Obr. 7.2 je pak výsledná autokorelační funkce.




Obrázek 7.1: Posun rámce při výpočtu autokorelace.




Obrázek 7.2: Autokorelační funkce.

### 7.2.2 Výpočet lagu a určení znělosti

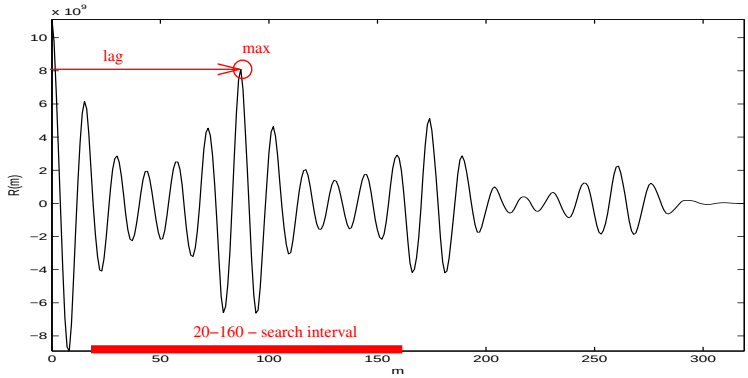
Lag se z ACF určí hledáním indexu jejího maxima:

$$lag = \arg\max_{m} R(m). \tag{7.3}$$

Znělost rámce můžeme odhadnout porovnáním nalezeného maxima s nultým (maximálním) autokorelačním koeficientem. Konstanta $\alpha$ se musí zvolit experimentálně.

$$\begin{array}{rcl} R_{max} < \alpha R(0) & \Rightarrow & \text{neznělý} \\ R_{max} \geq \alpha R(0) & \Rightarrow & \text{znělý} \end{array} \tag{7.4}$$

Pro uvedený příklad jsou nalezený lag a velikosti koeficientů $R_{max}$ a $R(0)$ zobrazeny na Obr. 7.3.



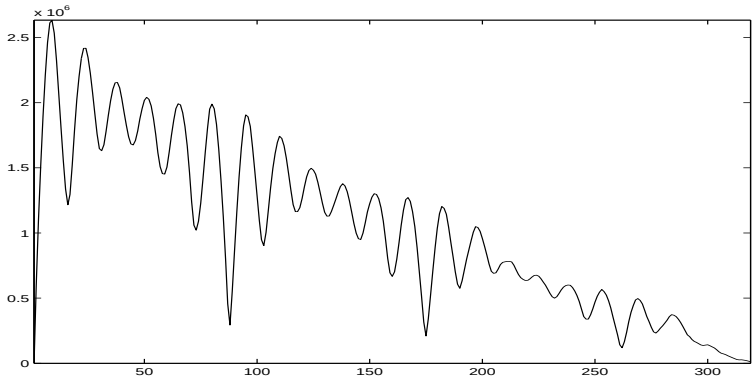

Obrázek 7.3: Nalezené maximum ACF (pro uvedený obrázek $L$=87).

### 7.2.3 AMDF

V dávných dobách, kdy bylo násobení náročnější na čas procesoru, se autokorelační funkce nahrazovala funkcí AMDF (Average Magnitude Difference Function):

$$R_D(m) = \sum_{n=0}^{N-1-m} |s(n) - s(n+m)|, \tag{7.5}$$

kde bylo naopak nutné hledat pro určení lagu *minimum*. Příklad AMDF je na Obr. 7.4.




Obrázek 7.4: Average Magnitude Difference Function

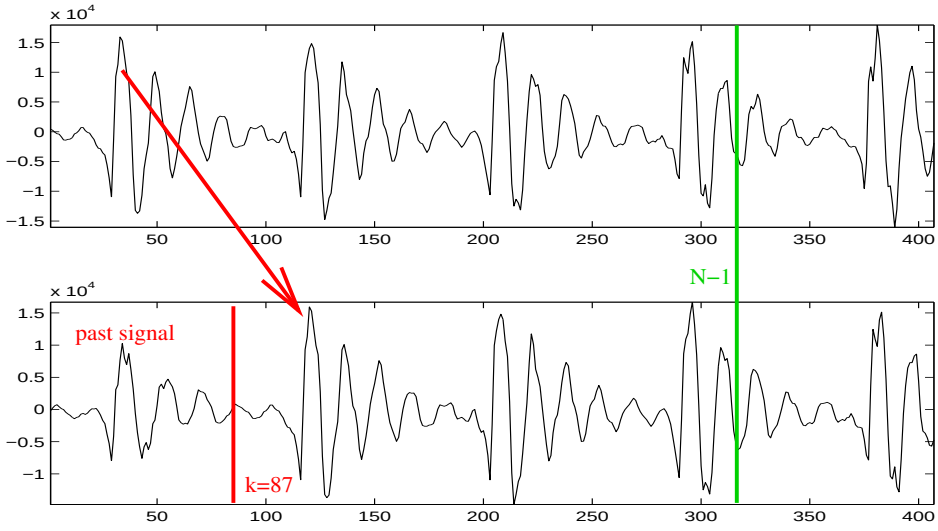
### 7.2.4 Cross-correlation function

Nevýhoda standardní ACF je postupné "zkracování" oblasti, ze které autokorelační koeficienty počítáme. Můžeme-li si dovolit použít celý signál (při reálném zpracování si ho musíme zapamatovat), lze přejít na cross-korelační funkci (cross-correlation function - CCF). Začátek rámce označíme $zr$, CCF je definována:

$$CCF(m) = \sum_{n=zr}^{zr+N-1} s(n)s(n-m) \tag{7.6}$$

DEF

Posunutí při výpočtu CCF je znázorněno na Obr. 7.5.




Obrázek 7.5: Posun rámce při výpočtu CCF.

Při výpočtu CCF můžeme ale narazit na problém příliš velké energie jednoho ze signálů, která "přebije" podobnost dvou rámců. Problém je ilustrován na Obr. 7.6.

### 7.2.5 Normalized cross-correlation function

Rozdílnost energií originálního a posunutého rámce můžeme řešit pomocí normalizace: **NCCF**

$$NCCF(m) = \frac{\sum_{n=zr}^{zr+N-1} s(n)s(n-m)}{\sqrt{E_1 E_2}}, \tag{7.7}$$

DEF

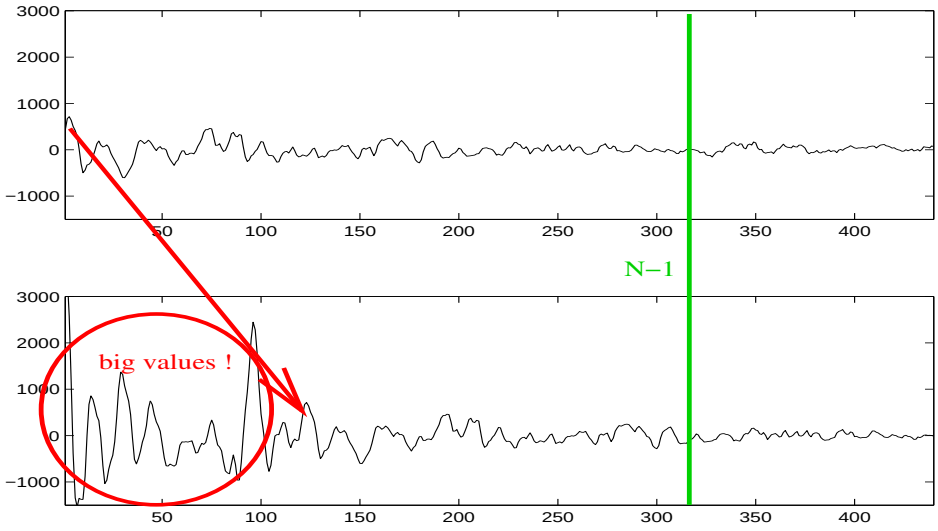
kde $E_1$ a $E_2$ jsou energie originálního a posunutého rámce:

$$E_1 = \sum_{n=zr}^{zr+N-1} s^2(n) \qquad E_2 = \sum_{n=zr}^{zr+N-1} s^2(n-m) \tag{7.8}$$

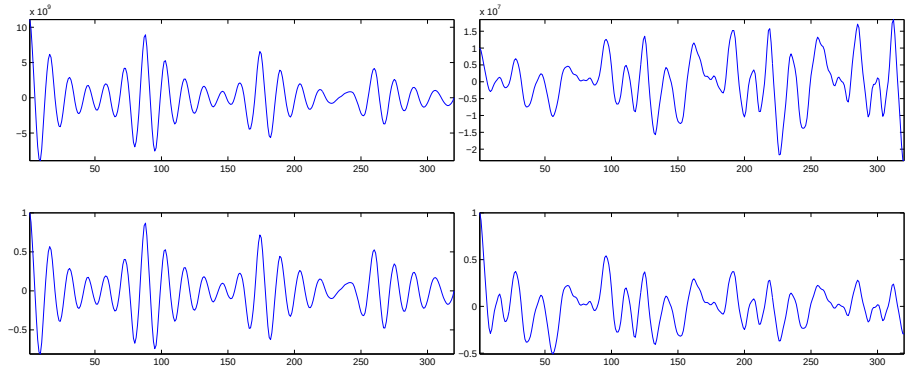
Srovnání CCF a NCCF pro příklad, kdy CCF "nevadí" a pro případ, kdy má CCF problém, je na Obr. 7.7.

## 7.3 Odstraněni vlivu formantů u autokorelačních metod

Postranní maxima v autokorelační funkci jsou způsobena formanty. Připomeňme si, že formanty (rezonanční frekvence) jsou zodpovědné za zákmity v rámci jedné periody řeči. Tato postranní maxima autokorelační metody nedostatečně potlačují, maximální lag pak může být mylně detekován v jednom z těchto postranních maxim.




Obrázek 7.6: Problém CCF při rozdílné energii rámců.



Obrázek 7.7: Ilustrace rozdílu CCF (nahoře) a NCCF (dole) pro případ, kde s energiemi není problém (vlevo) a kde tento problém je (vpravo).

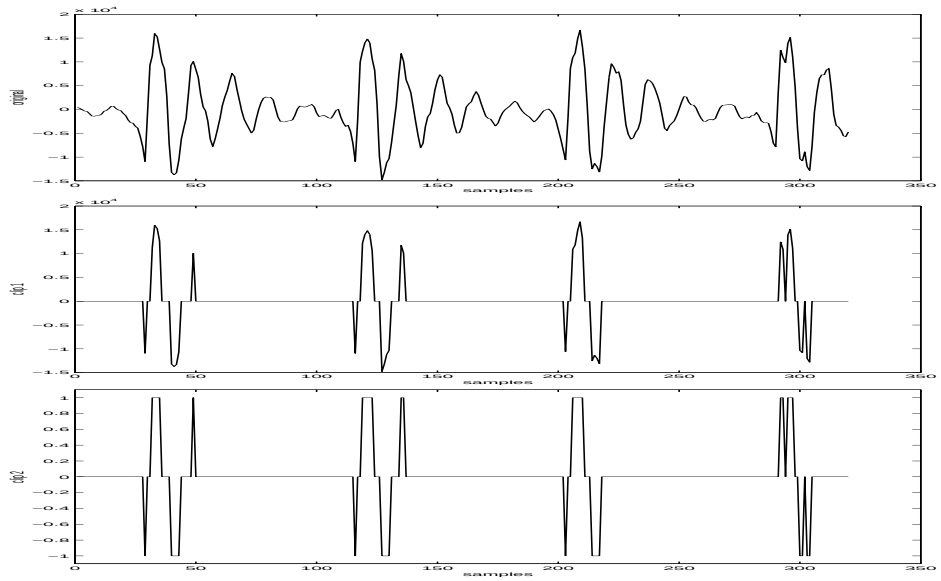
### 7.3.1 Centrální klipování – Center Clipping

je jedna z metod pro omezení postranních maxim. Center clipping předzpracovává signál pro ACF, zajímáme se pouze o *špičky signálu*. Definujeme tzv. klipovací úroveň $c_L$. V první variantě této metody ze signálu "vynecháváme interval" $< -c_L, +c_L >$. Ve druhé variantě nahrazujeme hodnotou 1 signál tam, kde je překročena úroveň $c_L$ a hodnotou -1 tam, kde signál nedosáhne úrovně $-c_L$:

$$c_1[s(n)] = \begin{cases} s(n) - c_L & \text{pro} \quad s(n) > c_L \\ 0 & \text{pro} \quad -c_L \leq s(n) \leq c_L \\ s(n) + c_L & \text{pro} \quad s(n) < -c_L \end{cases} \tag{7.9}$$

$$c_2[s(n)] = \begin{cases} +1 & \text{pro} \quad s(n) > c_L \\ 0 & \text{pro} \quad -c_L \leq s(n) \leq c_L \\ -1 & \text{pro} \quad s(n) < -c_L \end{cases} \tag{7.10}$$

Obr. 7.8 ilustruje klipování na rámci řečového signálu pro klipovací úroveň 9562.




Obrázek 7.8: Původní signál a jeho klipované varianty $c_1(s[n])$ a $c_2(s[n])$.

**Určení klipovací úrovně**

Vzhledem ke kolísání signálu $s(n)$ nemůže být konstantní a je nutné ji určovat pro každý rámec, na kterém odhadujeme základní tón. Jednoduchou metodou je určení klipovací úrovně z maximální absolutní hodnoty vzorků v rámci:

$$c_L = k \max_{n=0\ldots N-1} |x(n)|, \tag{7.11}$$

kde konstanta $k$ se volí od 0.6 do 0.8. Sofistikovanější metoda využívá rozdělení rámce na několik mikro-rámců, např. $x_1(n)$, $x_2(n)$, $x_3(n)$ o třetinové délce. Klipovací úroveň je pak určena pomocí "nejslabšího" maxima z těchto mikro-rámců jako:

$$c_L = k \min \{\max |x_1(n)|, \max |x_2(n)|, \max |x_3(n)|\} \tag{7.12}$$

Problémem může být klipování šumu v pauzách, kde může být následně detekován základní tón a znělost. Metodu je vhodné doplnit určením úrovně ticha $s_L$ (silence level). Pokud maximum signálu $< s_L$, pak se znělost a lag neurčují.

### 7.3.2 Využití chyby lineární predikce

Jedná se o předzpracování nejen pro metodu ACF, ale i pro jiné algoritmy určení základního tónu. Pro opakování: chybu lineární predikce získáme jako rozdíl skutečného a předpovězeného signálu:

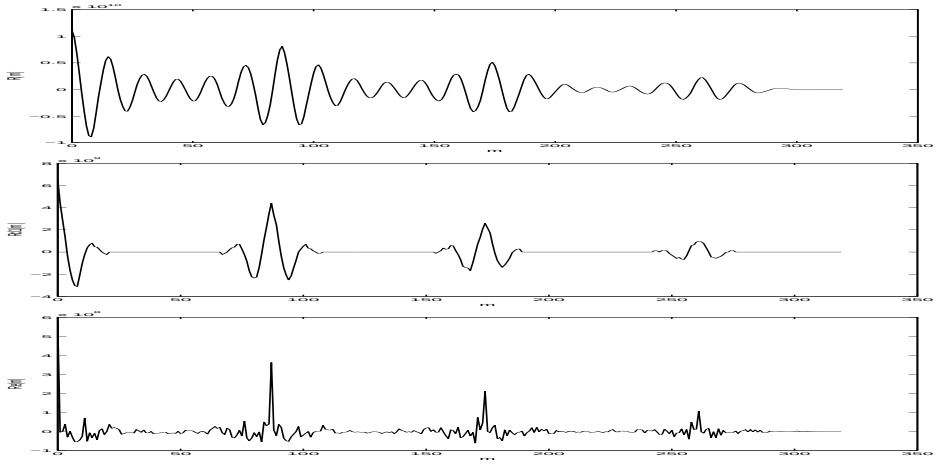
$$e(n) = s(n) - \hat{s}(n) \tag{7.13}$$

$$E(Z) = S(z)[1 - (1 - A(z))] = S(z)A(z) \tag{7.14}$$

$$e(n) = s(n) + \sum_{i=1}^{P} a_i s(n-i) \tag{7.15}$$

Signál $e(n)$ již neobsahuje informaci o formantech, proto je k určování základního tónu vhodnější než základní signál. Určení lagu z chybového signálu můžeme

provést pomocí ACF. Obrázek 7.9 presentuje autokorelační funkce vypočítané ze základního signálu, z klipovaného signálu a z chyby lineární predikce.



Obrázek 7.9: Autokorelační funkce původního signálu, klipovaného signálu a signálu chyby lineární predikce.

## 7.4 Dlouhodobý prediktor chyby predikce pro určení základního tónu

Podobně jako u LPC se snažíme předpovědět $n$-tý vzorek signálu. Ne ale z $P$ předcházejících vzorků (jako u LPC), ale ze jednoho nebo dvou vzorků vzdálených o předpokládaný lag. Pokud určíme posun s minimální energií chyby predikce, lag jsme nalezli. Predikovanou chybu predikce (pro dva vzdálené vzorky) zapíšeme:

$$\hat{e}(n) = -\beta_1 e(n-m+1) - \beta_2 e(n-m) \tag{7.16}$$

Chyba prediktoru chyby predikce je pak dána:

$$ee(n) = e(n) - \hat{e}(n) = e(n) + \beta_1 e(n-m+1) + \beta_2 e(n-m) \tag{7.17}$$

Když chceme minimalisovat energii tohoto signálu:

$$\min E = \min \sum_{n=0}^{N-1} ee^2(n), \tag{7.18}$$

postupujeme podobně jako při výpočtu LPC koeficientů, jako řešení dostáváme pro koeficienty $\beta_1$ a $\beta_2$:

$$\begin{aligned} \beta_1 &= [r_e(1)r_e(m) - r_e(m-1)]/[1 - r_e^2(1)] \\ \beta_2 &= [r_e(1)r_e(m-1) - r_e(m)]/[1 - r_e^2(1)], \end{aligned} \tag{7.19}$$

kde $r_e(m)$ jsou normované autokorelační koeficienty chybového signálu $e(n)$. Po dosazení těchto koeficientů do vzorce pro energii 7.18 můžeme tuto energii zapsat v závislosti na posunutí $m$ jako:

$$E(m) = 1 - K(m)/[1 - r_e^2(1)] \tag{7.20}$$

$$\text{kde } K(m) = r_e^2(m-1) + r_e^2(m) - 2r_e(1)r_e(m-1)r_e(m) \tag{7.21}$$

Lag nyní můžeme najít buď tak, že vyhledáme minimální energii nebo tak, že najdeme maximální hodnotu funkce $K(m)$ (uvědomme si, že jmenovatel $1 - r_e^2(1)$ na $m$ nezávisí).

$$L = \arg \min_{m \in [L_{min}, L_{max}]} E(m) = \arg \max_{m \in [L_{min}, L_{max}]} K(m) \tag{7.22}$$

## 7.5 Cepstrální analýza pro určení základního tónu

V kapitole 5 jsme viděli, že cepstrální koeficienty můžeme získat pomocí vztahu:

$$c(m) = \mathcal{F}^{-1} \left[ ln |\mathcal{F} s(n)|^2 \right] \tag{7.23}$$

Na Obr. 5.16 jsme si ukázali, že cepstrálních koeficientech se daří oddělit část koeficientů zodpovědnou za hlasový trakt (nízké indexy) od části zodpovědné za buzení, a tedy i za základní tón (vyšší indexy). Lag je nutné opět nalézt hledáním maxima $c(m)$ v rozsahu povolených lagů.

## 7.6 Zlepšení spolehlivosti určení základního tónu

Místo skutečného lagu je často detekován poloviční či několikanásobný lag. Předpokládejme např., že v pěti po sobě jdoucích rámcích byly detekovány tyto lagy: 50, 50, 100, 50, 50. V prostředním rámci se evidentně jedná o chybu: detekci dvojnásobného lagu. Chyby tohoto typu se můžeme snažit opravit několika způsoby.

### 7.6.1 Nelineární filtrace mediánovým filtrem

se provádí podle vztahu:

$$L(i) = \text{med} \left[ L(i-k), L(i-k+1), \ldots, L(i), \ldots, L(i+k) \right] \tag{7.24}$$

Medián seřadí hodnoty podle velikosti a vybere hodnotu, která se nachází uprostřed. Lagy z našeho příkladu tedy budou opraveny na 50, 50, 50, 50, 50.

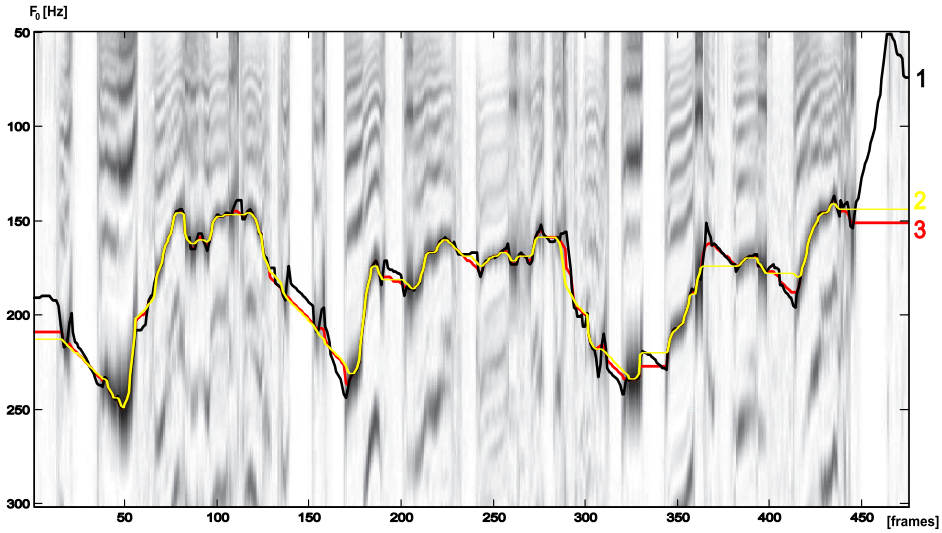
### 7.6.2 Metoda optimálních cest

V předcházejících metodách jsme lag určovali tak, že jsme určili pouze *jedno* maximum, případně minimum na jeden rámec. Hledání maxima či minima můžeme ovšem rozšířit na několik rámců vedle sebe: nebudeme hledat hodnotu, ale "cestičku", která minimalizuje (či maximalizuje) dané kritérium. Příspěvkem ke kritériu může být např. hodnota $\frac{R(m)}{R(0)}$ nebo energie chyby predikce pro daný lag. Dále je potřeba definovat hypotézy o tvaru cesty (cesta se nemůže z jednoho rámce na druhý výrazně změnit...).

Algoritmus má pak tyto kroky:

1. určení možných cest — např tak, že rozdíl v hodnotě lagu mezi sousedními rámci nesmí být větší než konstanta $\Delta L$.
2. určení celkového kriteria pro danou cestu.
3. výběr optimální cesty.

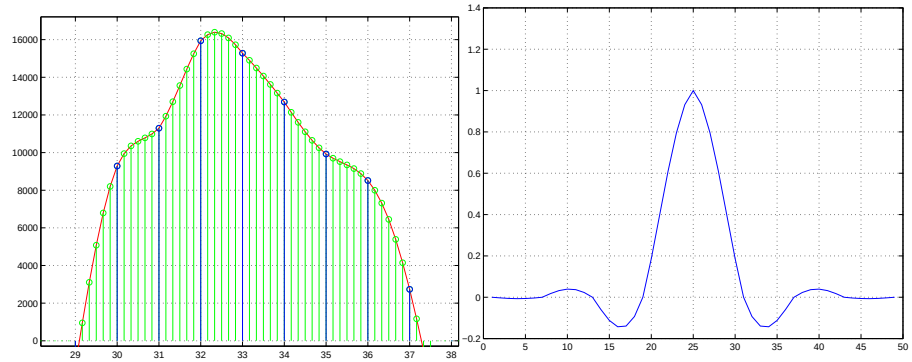
Na výslednou cestu optimálního základního tónu se můžete podívat na Obr. 7.10 (převzato z Diplomové práce Igora Szökeho).




Obrázek 7.10: Určení základního tónu metodou optimálních cest.

### 7.6.3 Desetinné vzorkování

Pro zvýšen přesnosti určení $F_0$ je vhodné signál nadvzorkovat a následně filtrovat. Dosáhneme tak zvýšení vzorkovací frekvence. Tuto operaci není nutné provádět "fyzicky"; dá se promítnout přímo do výpočtu autokorelačních koeficientů. Nadvzorkování často zamezíme falešné detekci dvojnásobku skutečného lagu. Příklad interpolovaného signálu a interpolačního filtru je uveden na Obr. 7.11.



Obrázek 7.11: Desetinné vzorkování pro přesný výpočet lagu a tvar interpolačního filtru.

# Kapitola 8

# Kódování řeči


Tato kapitola shrnuje metody používané pro kódování řeči. Požadavky na kódování jsou

- co nejmenší počet bitů.
- co největší kvalitu.
- co nejmenší zpoždění.
- co největší odolnost proti chybám.
- co nejmenší výpočetní náročnost.

... a je zřejmé (jako u mnoha jiných činností lidských), že jednotlivá kriteria jsou samozřejmě v rozporu. U kódování je také dobré si uvědomit, že zatímco rozpoznávání řeči je více "sexy" a zaplňuje více sekcí na odborných konferencích, kódování je jistě **komerčně nejdůležitější součástí automatického zpracování řeči**. Stačí si prohlédnout obsah svých kapes a kabelek.


Pro sekvenci kodéru a dekodéru (COder-DECoder) se často slovíčko zkratka **"codec"** nebo "kodek".

## 8.1 Úvod do kódování

### 8.1.1 Standardizace

Kódování řeči je nejvíce standardizovanou oblastí zpracování řeči. O normalizaci se starají následující instituce:


- CCITT (Centre Consultatif International Téléphonique et Télégraphique), ze kterého vznikla ITU-TSS (International Telecommunication Union — Telecom. Standardization Sector). Doporučení Gxxx. `http://www.itu.int`
- US DoD (Department of Defense). Federální standardy FSxxxx.
- ETSI (European Telecommunications Standards Institute), aktivní především v mobilní telefonii. `http://www.etsi.org`
- a další instituce, např. INMARSAT.

### 8.1.2 Dělení kodérů – podle principu

- **"kódování tvaru vlny"** (waveform coders) - vzorek po vzorku. Vysoká kvalita, ale za cenu velkého bitového toku. I pro neřečové signály.
- **vokodéry** (vocoders) založeny na poznatcích o tvorbě a slyšení řeči člověkem (buzení + modifikace). Rámce. LP model. Složitější než waveform coders. Střední a nízké rychlosti. Jen řeč.
- Jako **hybridní** jsou někdy nazývány algoritmy CELP (GSM). Obsahují model pro modifikační ústrojí, ale částečné kódování waveform pro buzení.

| označení | bitový tok |
|---|---|
| high rate | $> 16$ kbit/s |
| medium rate | 8 – 16 kbit/s |
| low rate | 2.4 – 8 kbit/s |
| very low rate | $< 2.4$ kbit/s |

Tabulka 8.1: Dělení kodérů podle bitového toku.

- **fonetické vokodéry** (phonetic vocoders) pracují s delšími řečovými úseky, než rámce (fonémy, automaticky natrénované jednotky). Jsou založené na rozpoznávání v kodéru a syntéze v dekodéru. Dosud jsou pouze v laboratorní stadiu, zatím není žádná normalizace. Kódují uspokojivě jen řeč, jsou závislé na jazyku nebo i na mluvčím.

### 8.1.3 Dělení podle bitového toku

DEF

**Bitový tok (bit rate)** je počet bitů za sekundu pro kódování řeči (**source coding**). Při dalším kódování bity přibývají – zabezpečení proti chybám v kanálu, kryptování, atd. má na starosti **channel coding**, kterým se zde nebudeme zabývat. Podle typu bitového toku dělíme kodéry na

- fixed-rate (klasické telefonní a mobilní sítě)
- variable-rate (paketové sítě, IP telefonie).

Tabulka 8.1 uvádí dělení kodérů podle průměrného bitového toku.

### 8.1.4 Dělení podle kvality

- **broadcast** (rozhlasová) – širší pásmo než telefonní, tok $> 64$ kbit/s.
- **network** nebo **toll** (síťová) – klasická kvalita analogového telefonního signálu, pásmo 300–3400 Hz.
- **communications** (komunikační) – poněkud horší, avšak srozumitelná, zachovává charakter mluvčího.
- **synthetic** (syntetická) – nepřirozená, nezachovává charakter mluvčího, snížená srozumitelnost.

### 8.1.5 Vyhodnocování kvality kódování

Metody dělíme na objektivní (kvalitu dokážeme vyhodnotit algoritmem, výsledkem je číslo) a subjektivní (kvalita je hodnocena lidskými posluchači).

#### Poměr signálu k šumu

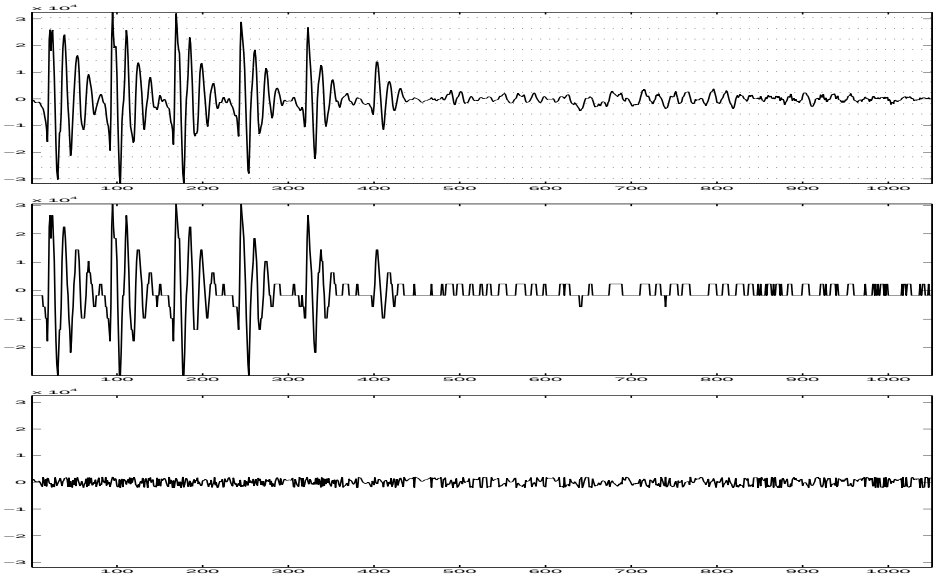
DEF

nebo také **odstup signálu od šumu**, (signal-to-noise ratio SNR) je nejjednodušší technika objektivního vyhodnocení:

$$SNR = 10 \log_{10} \left\{ \frac{\sum_{n=0}^{N-1} s^2(n)}{\sum_{n=0}^{N-1} [s(n) - \hat{s}(n)]^2} \right\}$$

Nevýhoda je ta, že pohlíží na signály "globálně", a i když může být signál v některé části totálně neposlouchatelný, můžeme stále "nahnat průměrnou hodnotu" na dlouhých úsecích signálu, kde je kódování slušné.

Na Obr. 8.1 je ilustrace výpočtu SNR, na kódování slabiky "as" pomocí 4 bitů (16 kvantizačních úrovní). Vypočtený SNR je 14.89 dB.



Obrázek 8.1: Výpočet poměru signálu k šumu: původní signál, kvantovaný signál, chyba kvantování.

### Segmentální poměr signálu k šumu (SEGSNR)

Je reprezentativnější než SNR, protože signál nejprve rozdělíme na rámce, spočítáme SNR pro jednotlivé rámce a pak teprve výsledné hodnoty SNR průměrujeme:

$$SEGSNR = \frac{10}{N_{ram}} \sum_{i=0}^{N_{ram}-1} \log_{10} \left\{ \frac{\sum_{n=0}^{l_{ram}-1} s^2(il_{ram}+n)}{\sum_{n=0}^{l_{ram}-1} [s(il_{ram}+n) - \hat{s}(il_{ram}+n)]^2} \right\} \quad (8.1)$$

Pro příklad z předcházející sekce v tomto případě vycházejí SNR v jednotlivých rámcích (při "klasické" délce rámce 160 vzorků):

20.04 19.63 14.35 0.21 4.26 -0.54(!)

a SEGSNR (jejich průměrná hodnota) je **9.66 dB**, což je podstatně horší, avšak o realitě více vypovídající nežli SNR.

### Logaritmická spektrální vzdálenost – logarithmic spectral distance

počítá zkreslení mezi originálním a kódovaným signálem ve *spektrální oblasti*:

$$d_2 = \sqrt{\int_{-1/2}^{+1/2} |V(f)|^2 \, df}, \quad \text{kde} \quad V(f) = 10 \log G(f) - 10 \log \hat{G}(f), \quad (8.2)$$

kde $G(f)$ a $\log \hat{G}(f)$ jsou spektrální hustoty výkonu originálního a kódovaného rámce. Dá se velmi snadno počítat pomocí LPC-cepstrálních koeficientů, vystačíme si zde se sumou, nemusíme integrovat:

$$d_{2\,LPC} = \mu \sqrt{\sum_{i=1}^{\infty} [c_t^{(i)} - c_r^{(i)}]^2} \tag{8.3}$$

kde konstanta $\mu = 2\frac{10}{\ln 10}$ napomáhá převodu z přirozených logaritmů (viz definice cepstra) na dB, $c_t^{(i)}$ jsou cepstrální koeficienty testovacího a $c_r^{(i)}$ referenčního rámce. Odmocninu většinou vynecháváme a sumu limitujeme na pouhých $P$ členů, čímž se dopouštíme chyby, protože jsme si v kapitole o cepstru řekli, že na plné vyjádření spektra je třeba nekonečně mnoho LPCC koeficientů. Tato "zkrácená vzdálenost" se nazývá *truncated cepstral distance* nebo také *kvadratická cepstrální míra*:

$$d_{CEP} = \sum_{i=1}^{P} [c_t^{(i)} - c_r^{(i)}]^2, \tag{8.4}$$

### 8.1.6 Subjektivní měření kvality

normalizované postupy vyžadují vždy skupinu posluchačů, kteří kvalitu posuzují:

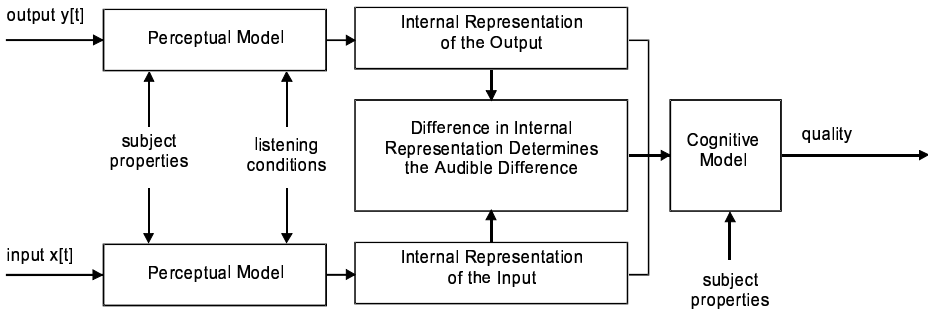
- DRT (Diagnostic Rhyme Test) – měření srozumitelnosti pomocí párů podobných slov (např. "meat"×"heat").
- DAM (Diagnostic Acceptability Measure) – soubor několika metod hodnotících kvalitu komunikačního systému.
- MOS (Mean Opinion Score) – skupina 12–64 posluchačů hodnotí kvalitu podle pětibodové stupnice. Posluchači jsou nejprve "kalibrování" signály se známými hodnotami MOS:

| MOS | kvalita | poznámka |
|---|---|---|
| 1 | bad (unacceptable) | velmi rušivý šum a artefakty v signálu |
| 2 | poor | ... |
| 3 | fair | něco mezi |
| 4 | good | ... |
| 5 | excellent | nerozeznatelné od originálu, bez slyšitelného šumu |

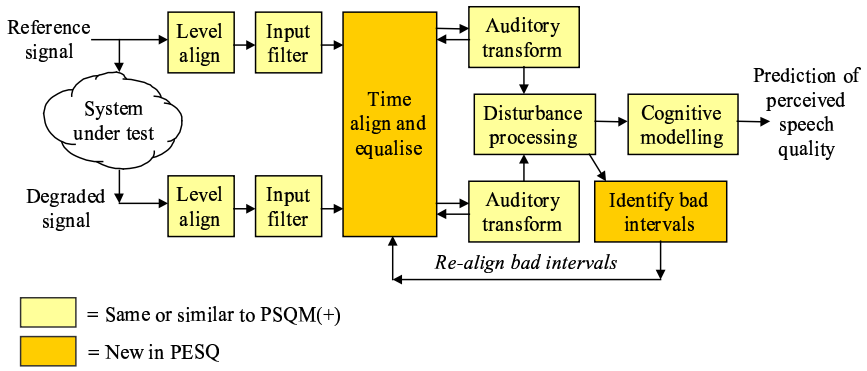
Poslechové testy jsou časově, organisačně i finančně velmi náročné, proto byly vyvinuty dvě techniky pro aproximaci výsledků lidských poslechových testů pomocí technik zpracování signálů – Perceptual Speech Quality Measure (PSQM – Obr. 8.2) a Perceptual Evaluation of Speech Quality (PESQ – Obr. 8.3). Obě metody jsou normalizovány jako ITU standardy: P.861 a P.862. PESQ je více adaptovaná na odhad kvality nikoliv jen kodéru, ale celého komunikačního řetězce, především v paketových sítích.

Pro zájemce o tato vyhodnocení kvality doporučuji: A.W.Rix et al.: Perceptual evaluation of speech quality (PESQ) a new method for speech quality assessment of telephone networks and codecs, in Proc. ICASSP 2001.[1]

[1] U všech článků Vám přednášející či členové skupiny Speech@FIT rádi poskytnou PDFka.




Obrázek 8.2: PSQM, převzato z OPTICOM Whitepaper on "State-of-the-Art Voice Quality Testing", 2000.

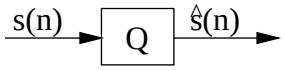



Obrázek 8.3: PESQ, převzato z OPTICOM Whitepaper on "State-of-the-Art Voice Quality Testing", 2000.

## 8.2 Kódování tvaru vlny (waveform coding)

### 8.2.1 Pulsní kódová modulace (PCM)

název je poněkud historický, jedná se o kvantování jednotlivých vzorků nezávisle na sobě pomocí pevného počtu bitů (viz Obr. 8.4).




Obrázek 8.4: Pulsní kódová modulace PCM.

Při kódování PCM je důležité, jakým způsobem jsou rozloženy *kvantovací hladiny*. Nejjednodušší je *rovnoměrné* (lineární, uniformní) rozložení, které se používá např. při kódování zvuku pro CD nebo ve zvukových souborech WAV (16 bitů). Odstup signálu od šumu je v tomto případě dán vztahem:

$$SNR = 6B + K \tag{8.5}$$

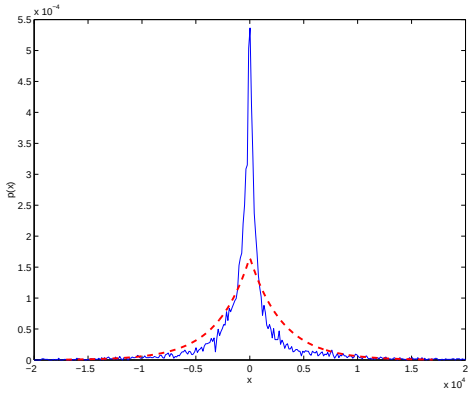
v dB, kde $B$ je počet bitů a $K$ je konstanta závislá na charakteru signálu. Pro CD

je tedy teoretický odstup s/š 16×6=96 dB. Rovnici 8.5 můžeme také interpretovat tak, že přidáme-li 1 bit, zlepší se poměr s/š o 6 dB.

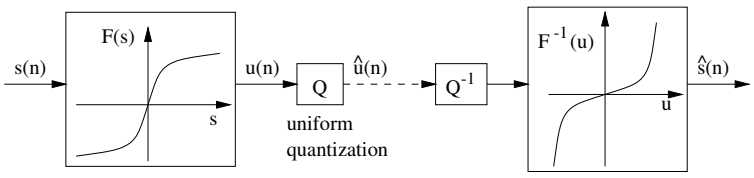
Rovnoměrné rozložení kvantovacích hladin ovšem není vhodné pro malé signály, jak jsme viděli i v příkladu výpočtu SNR. Řečový signál navíc obsahuje mnoho takových "malých vzorků"; funkce hustoty rozložení pravděpodobnosti (probability distribution function – pdf) velikostí vzorků řeči se dá aproximovat pomocí *Laplaceova* rozdělení:

$$p(x) = \frac{1}{\sqrt{2}\sigma_x} e^{-\frac{\sqrt{2}|x|}{\sigma_x}}, \tag{8.6}$$

kde $\sigma_x$ je směrodatná odchylka. Obrázek 8.5 udává tuto funkci pro známý signál "létající prase", plnou čarou je zobrazen odhad pdf pomocí histogramu, tlustou čárkovanou čarou Laplaceovo rozdělení. Ze signálu byly před odhadem histogramu a směrodatné odchylky odstraněny úseky ticha.




Obrázek 8.5: Aproximace rozložení velikostí vzorků řeči Laplaceovým rozdělením.




Obrázek 8.6: Logaritmická PCM.

Bylo by tedy vhodné rozložit kvantovací hladiny nerovnoměrně: hustěji pro malé amplitudy a řidčeji pro amplitudy velké. Lidské ucho navíc není na změnu amplitudy citlivé lineárně, ale logaritmicky, což jen potvrzuje tento požadavek. Variantou PCM je **logaritmická PCM**, kde v kodéru dochází ke kompresi signálu, rovnoměrnému kvantování, a v dekodéru pak k expansi (Obr. 8.6). Nelineární funkce nemůže být přímo log, protože $\log(0)=-\infty$, log také není definován pro záporná čísla. Používají se dvě aproximace:

- v Evropě **A-law**:

$$u(n) = S_{max} \frac{1 + \ln A \frac{|s(n)|}{S_{max}}}{1 + lnA} \operatorname{sign}[s(n)], \tag{8.7}$$

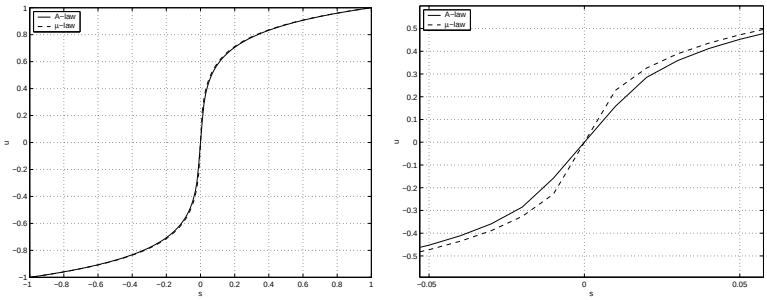
kde $A$ =87.56.

- v USA $\mu$-law:

$$u(n) = S_{max} \frac{\ln\left(1 + \mu \frac{|s(n)|}{S_{max}}\right)}{\ln(1+\mu)} \operatorname{sign}[s(n)], \tag{8.8}$$

kde $\mu$=255.

Ze srovnání A-law a $\mu$-law (Obr. 8.7) je patrné, že obě funkce jsou prakticky identické. Detail dokládá, že A-law má poněkud lepší rozlišení pro velmi malé signály. Obě komprese zlepšují SNR pro malé signály až od 12 dB. Pro telefonní síť se používá log-PCM s 8 bity, kvalita odpovídá 13 lineárním bitům. Doporučení CCITT G.711.




Obrázek 8.7: Srovnání A- a $\mu$-law. Celkový pohled a zoom okolo nuly.

### 8.2.2 Adaptivní pulsní kódová modulace (APCM)

Přiřazení kvantovacích hladin (rovnoměrné nebo nerovnoměrné) nemusí být fixní, ale může se *adaptovat* v závislosti na signálu. Rozložení hladin se počítá z bloku několika vzorků. Informace o kvantovacích hladinách:

- se může vysílat do dekodéru jako přídavná informace, tzv. *feed-forward*.
- se může počítat také zpětně z několika minulých vzorků, které má k disposici i dekodér. V tomto případě není nutné informaci přenášet, tzv. *feed-back*.

APCM se nepoužívá samostatně, ale je součástí některých kodérů, např. full rate GSM RPE-LTP (viz. sekce 8.4.6).

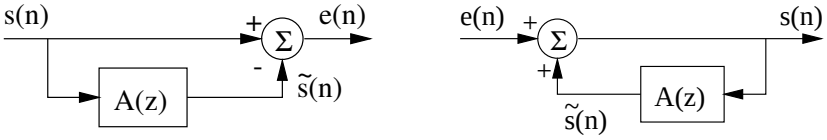
### 8.2.3 Diferenční pulsní kódová modulace (DPCM)

Jediný signál, kde neexistují závislosti mezi jednotlivými vzorky, je bílý šum. Ten příliš často nekódujeme... Pokud mezi vzorky závislosti existují, můžeme se pokusit odhadnout hodnotu současného vzorku z několika předcházejících, a přenést pouze chybový signál. Pokud je odhad dobrý, jsou energie chybového signálu i jeho amplitudy malé a můžeme pro jeho kódování použít méně bitů, než pro "plný" signál. Předpověď vzorků pomocí filtru jsme viděli již v kapitole o LPC.

Zde budeme současný vzorek předpovídat pomocí FIR-filtru definovaného jako[2]:

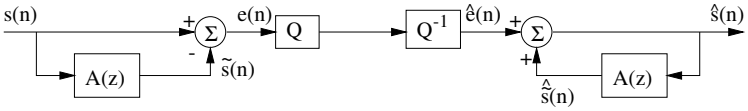
$$A(z) = \sum_{i=1}^{P} a_i z^{-i} \tag{8.9}$$

Chybový signál je dán rozdílem skutečného signálu $s(n)$ a předpovězeného $\tilde{s}(n)$ (Obr. 8.8 vlevo). **Dekodér** k takovému kodéru je jednoduchý: z minulých vzorků výstupu předpovídá signál, k předpovězenému vzorku se přičítá chyba kvantování (Obr. 8.8 vpravo).



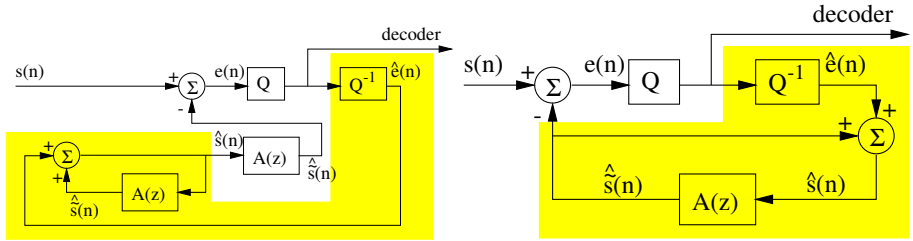

Obrázek 8.8: Princip kodéru a dekodéru DPCM.

Problém je v tom, že chybový signál je nutné kvantovat (Obr. 8.9) a v tomto případě by dekodér odhadoval následující vzorek z **jiných vzorků** než kodér — kodér má k disposici originální vzorky $s(n)$, dekodér pouze $\hat{s}(n)$, které díky kvantování $e(n)$ nejsou stejné jako $s(n)$ !




Obrázek 8.9: Kvantování chybového signálu v DPCM.

Budeme tedy muset dekodér "vestavět do kodéru" a použít k odhadu jeho výstupní vzorky – Obr. 8.10 vlevo. Schéma je nyní značně složité a navíc vidíme, že se v něm filtr $A(z)$ objevuje zbytečně dvakrát se stejným vstupem. Můžeme zjednodušit na strukturu, která sice již není tak přehledná, ale ekonomičtější (Obr. 8.10 vpravo). Opět vidíme, že v kodéru se "skrývá" celý dekodér (označen žlutě).



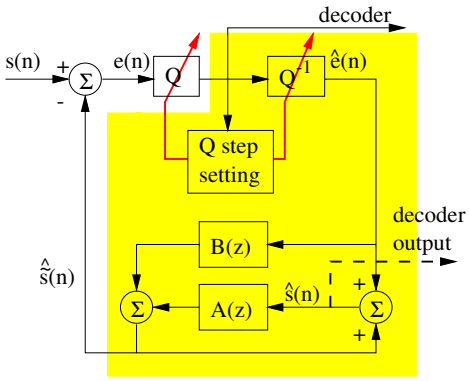

Obrázek 8.10: Reálné fungování DPCM: vlevo je princip, vpravo skutečné schema.

[2]Pozor! Tato definice se liší od kapitoly o LPC, kde byl polynom $A(z)$ definován jako $A(z) = 1 + \sum_{i=1}^{P} a_i z^{-i}$. Definice 8.9 se v literatuře a v normách často u waveform-kodérů objevuje, je proto použita i zde. V sekci 8.3 o vokodérech se k původní definici vrátíme

### 8.2.4 Adaptivní diferenční pulsní kódová modulace (ADPCM)

jediným rozdílem oproti DPCM je adaptace kroku pro kvantování chybového signálu. ADCPM je použita v normě G.721 (Obr. 8.11), která log-PCM (64 kbit/s) v pevné telefonní síti komprimuje na 32 kbit/s. Na rozdíl od jednoduchého schématu z předcházející sekce obsahuje G.721 dva filtry, které se podílejí na tvorbě chybového signálu $e(n)$: $A(z)$ funguje podobně jako v předcházejícím případě. $B(z)$ odhaduje vzorek na základě několika vzorků *chybového signálu*.

Blok "Q-step setting" provádí nastavení kvantizačního kroku. Žlutě (šedě) je opět vyznačen dekodér, "skrytý" v kodéru. Výstup dekodéru je označen čárkovanou šipkou "decoder output". Informace o adaptaci kvantovacího kroku i o hodnotách koeficientů filtrů $A(z)$ a $B(z)$ jsou počítány zpětně (back-ward), není tedy nutný jejich přenos do dekodéru. Dekodér si je spočítá sám z minulých vzorků.




Obrázek 8.11: ADPCM.

## 8.3 Vokodéry

- využívají poznatků o lidském řečovém ústrojí pro redukci bitového toku.
- uspokojivě zpracovávají pouze řeč, pokud jsou jim ke kódování předloženy jiné signály (např. hudba), výsledkem je většinou "cosi podobného řeči", samozřejmě zcela nesrozumitelného.
- využívají modelu buzení—filtr.

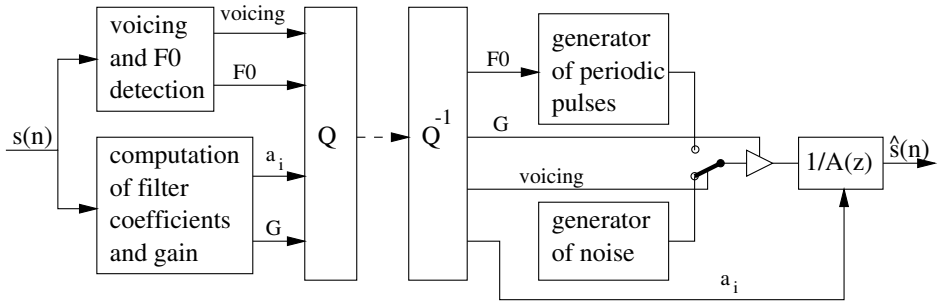
### 8.3.1 Kodér založený na lineárně prediktivním modelu - LPC

využívá principů probraných v kapitole o LPC. Vstupní signál je rozdělen na rámce, z každého je odhadnuta sada koeficientů polynomu[3]: $A(z) = 1 + \sum_{i=1}^{P} a_i z^{-i}$. Dále je spočten gain tohoto filtru $G$ a je detekována znělost a v případě znělého rámce perioda základního tónu (lag).

Vše je přeneseno do dekodéru, kde je podle znělosti generován bílý šum nebo periodický sled impulsů. Vzorky jsou násobeny gainem $G$ a filtrovány "syntezačním" filtrem $H(z) = \frac{1}{A(z)}$.

[3] Zde se pokorně vracíme k definici zavedené v kapitole o LPC...

Schéma na Obr. 8.12 využívá např. kodér FS1015 standardizovaný US ministerstvem obrany, pracující s bitovým tokem 2.4 kbit/s. Kvalita dekódované řeči s jednoduchým LPC kodérem je ovšem dosti špatná. Hlavní vinu nese příliš zjednodušené buzení. Proto se v moderních kodérech věnuje kódování buzení velká pozornost.




Obrázek 8.12: LPC kodér.

### 8.3.2 Residual Excited Linear Prediction – RELP

v každém rámci jsou odvozeny parametry filtru $A(z)$ a je vypočten chybový signál $e(n)$: z kapitoly o LPC si vzpomeneme, že se jedná o filtraci "inverzním filtrem", v $z$-doméně $E(z) = A(z)S(z)$. Tento signál je přenesen do dekodéru, kde je filtrován filtrem $H(z) = \frac{1}{A(z)}$. Pokud nejsou koeficienty filtru $a_i$ ani chybový signál $e(n)$ kvantovány, výsledkem je naprosto přesně vstupní signál $s(n)$. Tento postup je ovšem značně nepraktický: nejen, že jsme bitový tok nesnížili oproti běžné PCM, ale ještě jsme jej *zvýšili* ! Vzorků $e(n)$ je totiž stejný počet jako $s(n)$ a navíc musíme ještě přenášet koeficienty $a_i$ (v nejhorším případě 10 reálných čísel na 1 rámec, tady např. 4×10×100=4 kbit/s). Buzení i koeficienty bude tedy nutné zakódovat efektivněji.

## 8.4 Redukce bitového toku a zlepšení kvality u vokodérů


Tato sekce nepopisuje konkrétní kodér, ale spíše sady technik, které se v kodérech používají pro

- zmenšení bitového toku.
- zlepšení kvality.

Uvidíme, že zlepšování kvality se bude především týkat **buzení** $e(n)$, které je v základním LPC kodéru velmi hrubě modelováno pouze bílým šumem nebo sekvencí pulsů.

### 8.4.1 Vektorové kvantování

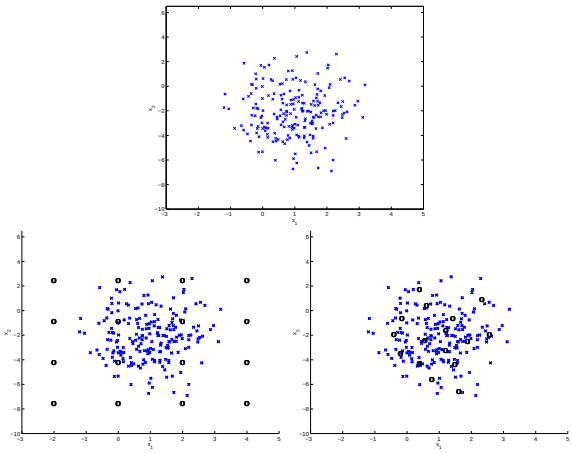
(Vector Quantization – VQ) je tak zásadní technika, používaná ne jen v kódování řeči, ale i v mnoha jiných aplikacích, že mu budeme věnovat poměrně velký prostor. Ve vokodérech je VQ používána na dvou místech:

1. pro kódování koeficientů $a_i$. Většinou nejsou kódovány přímo parametry $a_i$, ale odvozené koeficienty PARCOR, LAR nebo LSF (line spectral frequencies, též označované jako line spectral pairs LSP), které jsou ke kódování vhodnější. Kódová kniha VQ by pro plný počet koeficientů (často 10) musela obsahovat velký počet kódových vektorů, proto se vektor koeficientů často dělí do kratších "vektorků", které se kvantují samostatně. Hovoříme o split-VQ.
2. pro kódování buzení v metodách CELP (sekce 8.4.7).

### Princip VQ

Vycházíme z toho, že kódování $P$ - dimenzionálních vektorů je velmi nákladné ($P$ floatů je $P \times 4$ byte...). Skalární kvantování jednotlivých složek je plýtvání bity, lepší je umístit do $P$-rozměrného prostoru "typické" vektory a všechny kódované vektory na ně "zaokrouhlovat".

Obr. 8.13 ilustruje problém na kvantovaní dvourozměrných vektorů. Ke kvantování máme k disposici 4 bity. V případě skalárního kvantování každého rozměru nezávisle dostáváme velmi hrubé pokrytí dat a několik kombinací skalárních hodnot jsme vyplýtvali zcela zbytečně. Vektorové kvantovaní oproti tomu pokryje prostor optimálně svými kódovými vektory.



Obrázek 8.13: Skalární versus vektorová kvantizace. Nahoře: vektory, které máme kvantovat. Vlevo dole: skalární kvantizace se čtyřmi bity. Vpravo: vektorová kvantizace se čtyřmi bity.

### Kódová kniha a centroidy

Při vektorovém kvantování chceme $P$-rozměrné vektory

$$\mathbf{x} = [x_1, \ldots, x_P]^T \tag{8.10}$$

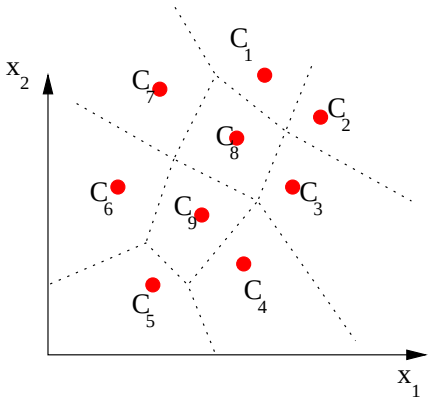
efektivně vyjádřit pomocí *kódové knihy*

$$\mathbf{Y} = \{\mathbf{y}_i; 1 \leq i \leq L\}. \tag{8.11}$$

DEF

Každý vektor je pak representován pouze indexem příslušného vektoru z kódové knihy: $u \in 1 \ldots L$. Kódové vektory se též nazývají *rekonstrukční* nebo *výstupní*.

Ke každému z nich patří buňka $C_i$ (nazývaná také Voronoiův region). Příklad pro 2-rozměrné vektory $\mathbf{x} = [x_1, x_2]^T$ je na Obr. 8.14 (kódové vektory $\mathbf{y}_i$ jsou značeny červenými puntíky):




Obrázek 8.14: Voronoiovy regiony a centroidy.

Při *kvantování* přiřadíme vektoru $\mathbf{x}$ ten kódový vektor, do jehož buňky $\mathbf{x}$ patří:

$$Q(\mathbf{x}) = \mathbf{y}_i \quad \text{pokud} \quad \mathbf{x} \in C_i. \tag{8.12}$$

Kódový vektor $\mathbf{y}_i$ se dá vyjádřit pouze indexem $i$. Abychom mohli VQ matematicky zapsat, musíme mít pro každou buňku tzv. *centroid* a musí být nadefinována *vzdálenost*. Definicí vzdálenosti je celá řada, nejběžnější je kvadratická míra:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})} = \sqrt{\sum_{k=1}^{P} |x_k - y_k|^2}. \tag{8.13}$$

Centroid musí mít ke všem vektorům v buňce $C_i$ minimální vzdálenost. Při použití kvadratické míry je centroid aritmetickým průměrem vektorů. Kódový vektor se obvykle rovná centroidu:

$$\mathbf{y}_i = \frac{1}{M_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}, \tag{8.14}$$

Máme-li soubor vektorů $\mathbf{x}(1) \ldots \mathbf{x}(N)$, můžeme *kvalitu* kvantifikátoru změřit pomocí globální či průměrné vzdálenosti (nebo zkreslení):

$$D_{VQ} = \frac{1}{N} \sum_{n=1}^{N} d\left(\mathbf{x}(n), Q[\mathbf{x}(n)]\right). \tag{8.15}$$

Teoreticky je toto zkreslení rovno:

$$D_{VQ} = \int_{\mathbf{x}} d\left(\mathbf{x}, Q[\mathbf{x}]\right) p(\mathbf{x}) d\mathbf{x}, \tag{8.16}$$

kde $p(\mathbf{x})$ je hustota rozložení pravděpodobnosti výskytu vektorů v prostoru a $Q[\mathbf{x}]$ je příslušný kódový vektor. Integrujeme přes celý prostor.

Teoreticky má pro dané $L$ a danou definici vzdálenosti $d(\cdot, \cdot)$ existovat *optimální* kvantifikátor, který $D_{VQ}$ minimalizuje. Bohu žel neumíme takový kvantifikátor analyticky spočítat...

### Vytvoření kvantifikátoru aneb učení kódové knihy

Kódovou knihu VQ nenalezneme v tabulkách a není možné odvodit vztahy, které by ji vytvořily analyticky. Je nutné ji *natrénovat* (naučit) na trénovací populaci vektorů. Na obrázku 8.13 je pro danou trénovací populaci zobrazen vektorový kvantifikátor se stejným počtem vektorů jako kvantifikátor skalární. Je zřejmé, že s VQ dosáhneme při stejném počtu hodnot (a tedy i bitů!) mnohem menšího zkreslení. Otázkou je, jak pro danou populaci trénovacích vektorů naučit kódovou knihu.

### $K$-means

$K$-means je standardní označení tohoto algoritmu. Nám jde o natrénování kódové knihy $\mathbf{Y}$ o $L$ vektorech, měli bychom tedy psát spíše $L$-means. Potřebujeme

- inicialisovanou kódovou knihu $\mathbf{Y}(0)$.
- trénovací vektory $\mathbf{x}(1) \ldots \mathbf{x}(N)$, které jsou representativní pro naši aplikaci.

Algoritmus postupuje v *iteracích*, které označíme pořadovým číslem $k$. Hodnoty po inicialisaci označíme $k = 0$. Dále budeme potřebovat kriterium pro zastavení iterací. Bude jím *práh* $\varepsilon$ pro relativní změnu celkového zkreslení $D_{VQ}$. Algoritmus vypadá takto:

- **Inicialisace:** $k = 0$, definujeme $\mathbf{Y}(0)$.
- **Krok 1:** Nejprve přiřadíme trénovací vektory buňkám — "zakódujeme je:"

$$Q[\mathbf{x}] = \mathbf{y}_i(k) \;\text{ pokud }\; d(\mathbf{x}, \mathbf{y}_i(k)) \leq d(\mathbf{x}, \mathbf{y}_j(k)) \;\text{ pro }\; j \neq i,\; j \in 1 \ldots L \tag{8.17}$$

  Vektor $\mathbf{x}$ pak náleží buňce $C_i(k)$. Všimněme si, že kódové vektory i buňky nesou označení "generace" kódové knihy $k$. Celkové zkreslení vypočteme podle rovnice 8.15.

- **Krok 2:** Je-li relativní pokles zkreslení menší než nastavený práh:

$$\frac{D_{VQ}(k-1) - D_{VQ}(k)}{D_{VQ}(k)} \leq \varepsilon, \tag{8.18}$$

  ukončíme algoritmus a prohlásíme $k$-tou kódovou knihu za výsledek: $\mathbf{Y} = \mathbf{Y}(k)$

- **Krok 3:** Počítáme nové centroidy každé buňky, uděláme z nich nové kódové vektory:

$$\mathbf{y}_i(k+1) = Cent(C_i(k)) = \frac{1}{M_i(k)} \sum_{\mathbf{x} \in C_i(k)} \mathbf{x}, \tag{8.19}$$

  kde $M_i(k)$ je počet trénovacích vektorů přiřazených buňce $i$ při kódové knize generace $k$. Dále inkrementujeme: $k = k + 1$, návrat na krok 1.

Problém algoritmu $K$-means tkví především v *inicialisaci kódové knihy.* Tu můžeme inicialisovat pomocí zcela náhodných hodnot. Můžeme také náhodně vybrat $L$ vektorů z trénovacího setu a prohlásit je za původní kódovou knihu. Ani jedna metoda nezaručuje, že se algoritmus během iterací nezhroutí (pokud k některému z kódových vektorů není v kódování přiřazen ani jeden vektor, objeví se v (8.19) dělení nulou a nedobrovolně končíme...).

**Linde – Buzo – Gray**

Tento algoritmus řeší inicialisaci postupným dělením kódové knihy. Její velikost se postupně zvětšuje: $1 \rightarrow 2 \rightarrow 4 \rightarrow \ldots \rightarrow L$. Velikost kódové knihy musí být mocninou 2. LBG je jakousi "nadstavbou" algoritmu $K$-means: pro každou velikost kódové knihy spouštíme $K$-means tak, jak byl popsán v předcházející sekci. "Velké" iterace LBG budeme značit $r = 0 \ldots R-1$, kde $L = 2^R$.

- **Inicialisace:** $r = 0$, kódová kniha $\mathbf{Y}(0)$ má 1 kódový vektor, který je centroidem *všech* trénovacích vektorů.
- **Fáze 1:** z kódové knihy o $2^r$ vektorech uděláme dvakrát větší kódovou knihu (o $2^{r+1}$ vektorech) tak, že kódové vektory *rozštěpíme*:

$$\mathbf{y}_i(r) \rightarrow \begin{cases} \mathbf{y}_{2i-1}(r+1) = \mathbf{y}_i(r) + \boldsymbol{\Delta} \\ \mathbf{y}_{2i}(r+1) = \mathbf{y}_i(r) - \boldsymbol{\Delta} \end{cases}, \tag{8.20}$$

  kde $\boldsymbol{\Delta}$ je malý vektorek, kterým od sebe dva nové kódové vektory "odtáhneme".
- **Fáze 2:** spustíme $K$-means pro $\mathbf{Y}(r+1)$.
- **Fáze 3:** je-li $r+1 = R$, konec, výsledná kódová kniha je $\mathbf{Y} = \mathbf{Y}(r+1)$. Jinak zpět na fázi 1.

Obrázek 8.15 ilustruje učení VQ s $L$=8 kódovými vektory metodou LBG pro LPC-cepstrální vektory ze známého testovacího signálu "létající prase".

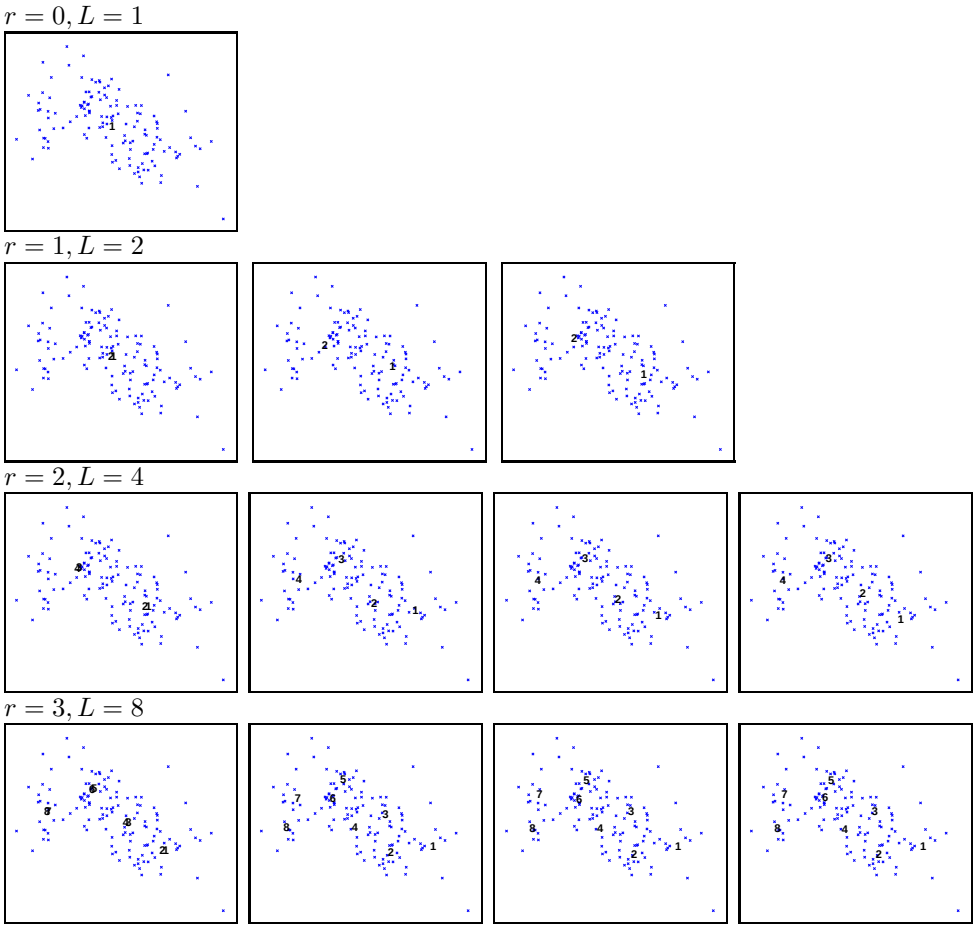
**Varianty VQ**

Trénování kódové knihy, ale v případě velkých $K$ i vlastní kódování jsou velmi výpočetně náročné, proto se snažíme o optimalisaci VQ:

- split-VQ: vektor je rozdělen na několik sub-vektorů s méně koeficienty. Použití několika menších codebooků (typicky 3-3-4 pro $P$=10).
- algebraická VQ: kódové vektory nejsou rozmístěny libovolně, ale jejich pozice jsou deterministické (např. uniformě na hyper-ploše), není nutné porovnávat vstup se všemi kódovými vektory.
- náhodný codebook (pro trénování): pro velká $K$ se kvalita natrénovaného codebooku blíží náhodnému, proto trénování není potřeba.
- tree-structured VQ: zapamatujeme si generace při trénování LBG a předpokládáme, že když byl vektor $\mathbf{y}^k$ v generaci $k$ kódové knihy přiřazen nějakému kódovému vektoru, pak pro generaci $k+1$ může náležet jen jeho dětičkám. Tato metoda je suboptimální, ale potřebuje jen $2 \log_2 K$ srovnání na rozdíl od $K$.
- multi-stage VQ: jsou použity 2 codebooky, ve druhém se kvantuje chyba prvního. Při dekódování jsou kódové vektory z obou sečteny.

### 8.4.2 Dlouhodobý prediktor – long term predictor (LTP)

Dlouhodobý prediktor je první z technik, používanou pro kvalitní kódování buzení. Víme, že chybový signál LPC má charakter šumu, ale pouze krátkodobě. Pro znělé hlásky je signál v delším časovém horizontu korelován (tedy je podobný) po periodách základního tónu (Obr. 8.16).




Obrázek 8.15: LBG metoda při trénování kódové knihy o velikosti $L = 8$. Křížky označují trénovací vektory. Čísla jsou kódové vektory. V řádcích jsou vidět jednotlivé iterace $K$-means.

Pokud budeme chtít buzení efektivně zakódovat, budeme chtít, aby se co nejvíce podobalo bílému šumu. S dlouhodobou podobností na Obr. 8.16 máme problém a snažíme se ji odstranit. Pomůže nám k tomu dlouhodobý prediktor (LTP), již zmiňovaný v kapitole o detekci základního tónu, s přenosovou funkcí

$$-bz^{-L} \tag{8.21}$$

který předpovídá vzorek $s(n)$ z minulého vzorku $s(n-L)$ ($L$ je perioda základního tónu ve vzorcích – lag). Chybový signál dlouhodobého prediktoru pak dostaneme jako:

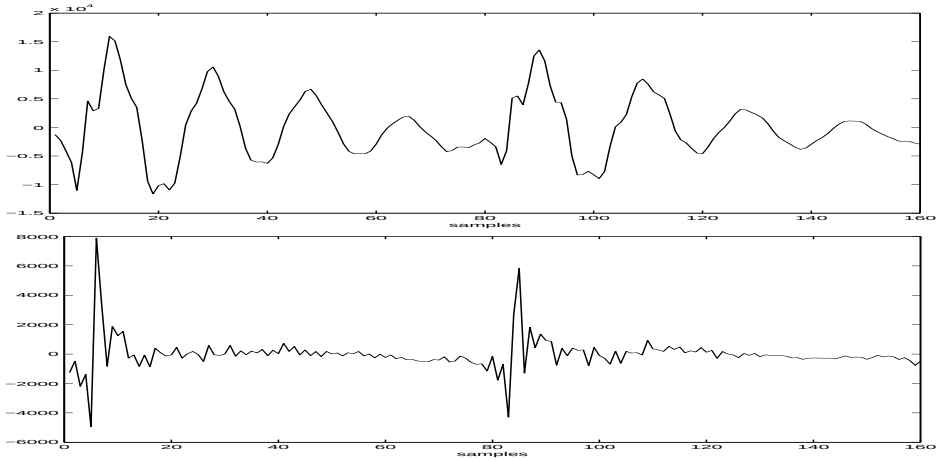
$$e(n) = s(n) - \hat{s}(n) = s(n) - [-bs(n-L)] = s(n) + bs(n-L), \tag{8.22}$$

takže se dá celá operace zapsat analogicky s krátkodobou LP jako filtrace filtrem

$$B(z) = 1 + bz^{-L}. \tag{8.23}$$

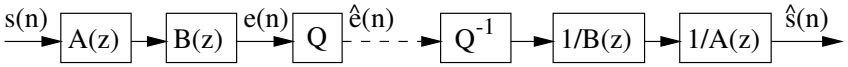
Můžeme samozřejmě nadefinovat i složitější LTP, který se nebude "dívat" jen na jeden, ale na několik vzorků v minulosti:

$$B(z) = 1 + \sum_{i=-k}^{k} b_i z^{-L+i} \tag{8.24}$$




Obrázek 8.16: Korelace chybového signálu LPC po periodách základního tónu.

Po zpracování LTP se pak chybový signál skutečně blíží bílému šumu a dobře se kóduje. LTP je součástí kodérů pro GSM, zmíněných v dalších sekcích. S použitím LTP mají kodér a dekodér následující strukturu na Obr. 8.17.




Obrázek 8.17: Kodér s dlouhodobým prediktorem.

### 8.4.3 Analýza syntézou

Tato metoda se používá pro vyhledávání optimálního buzení (samozřejmě kódovaného, pokud bychom buzení nekódovali, bude to přímo chybový signál po krátkodobé a dlouhodobé predikci). Ve většině kodérů není možné najít optimální buzení analyticky, proto se zkouší všechna možná buzení, vždy se vytvoří kompletní řečový signál, ten se srovná s originálem, vypočte se chyba, a buzení, které má nejmenší energii chyby, "vyhrává" (Obr. 8.18). Této metodě se také říká **uzavřená smyčka – closed-loop**. Pro odlišení od buzení $e(n)$ je rozdílový signál mezi dekódovaným a originálním (chyba) označen jako $ch(n)$.

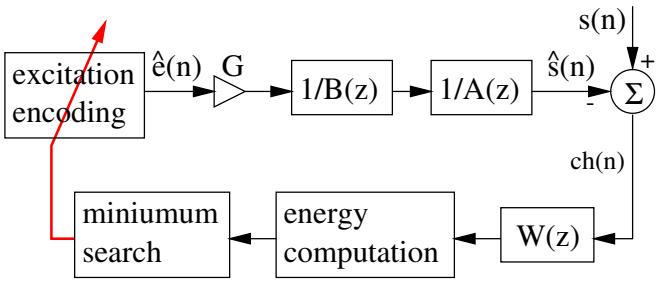
DEF

### 8.4.4 Perceptuální filtr

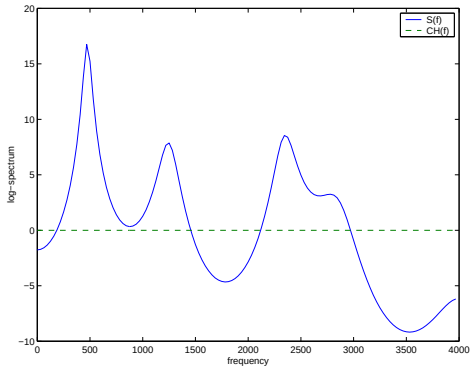
Ve schematu na předcházejícím obrázku je záhadná krabička $W(z)$. Jedná se o **perceptuální filtr**. K čemu slouží ? Jedná se o trik, který přibližuje kódování lidskému slyšení a vyhledává buzení tak, aby byl syntetizovaný signál co "nejpříjemnější" pro naše ouška. Představme si, že ve schématu filtr $W(z)$ není. Podle minima energie bylo vybráno buzení, které produkuje signál $ch(n)$ s plochým spektrem. Na Obr. 8.19 se podívejme na srovnání spektra chyby $CH(f)$ s vyhlazeným spektrem řečového signálu[4]

DEF

[4]na signál byla aplikována preemfáze, jinak by se výšky formantů postupně zmenšovaly.




Obrázek 8.18: Princip hledání optimálního buzení pomocí analýzy syntézou.




Obrázek 8.19: Srovnání spektra řeči s plochým spektrem chyby. Zatímco v oblastech formantů bude chyba maskována, v oblastech "ticha v řeči" bude jasně slyšitelná.

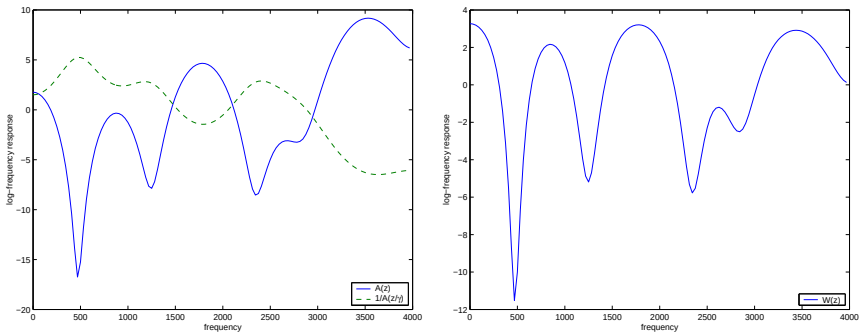
V oblasti formantů je řeč podstatně "silnější" než chyba a chyba tedy v těchto oblastech bude *maskována*: nebudeme ji slyšet. Jiná je situace v "údolích" mezi formanty, např. v pásmu 1500–2000 Hz: tam je spektrum chyby dokonce výše než spektrum řeči, v tomto pásmu budeme slyšet velmi rušivý šum.

Před výpočtem energie by tedy bylo vhodné říci: "V oblastech, kde jsou formanty, může být chyba jaká chce, protože ji stejně nebudu slyšet. Pojďme se spíše zaměřit na oblasti mezi formanty, kde má řečový signál malou energii". Toto budeme realizovat pomocí perceptuálního filtru $W(z)$, který chybový signál potlačí v oblasti formantů (při výpočtu energie se tyto oblasti neuplatní, jinými slovy "bude jedno, co tam bude"), a naopak zesílí v citlivých oblastech mezi formanty.

Perceptuální filtr se často realizuje jako:

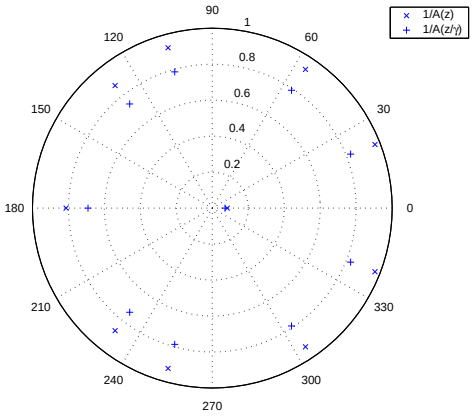
$$W(z) = \frac{A(z)}{A(z/\gamma)}, \quad \text{kde} \quad \gamma \in [0.8, 0.9] \tag{8.25}$$

Na frekvenční charakteristiku tohoto filtru se podíváme podrobněji (Obr. 8.20). Čitatel $A(z)$ má frekvenční charakteristiku inverzní k vyhlazenému spektru řeči. Filtr $\frac{1}{A(z/\gamma)}$ má podobnou frekvenční charakteristiku jako $\frac{1}{A(z)}$ (tedy vyhlazené spektrum řeči), ale díky "přitažení pólů" směrem ke středu jednotkové kružnice nebudou špičky tak ostré. Násobení těchto dvou frekvenčních charakteristik dá požadovanou frekvenční charakteristiku $W(f)$, která funguje podle našich představ.




Obrázek 8.20: Čitatel a jmenovatel perceptuálního filtru (vlevo) a jeho výsledná kmitočtová charakteristika.

Pro zajímavost se na Obr. 8.21 můžeme podívat na póly přenosových funkcí $\frac{1}{A(z)}$ a $\frac{1}{A(z/\gamma)}$. Dokázali byste odvodit, proč jsou póly $\frac{1}{A(z/\gamma)}$ blíže ke středu jednotkové kružnice?



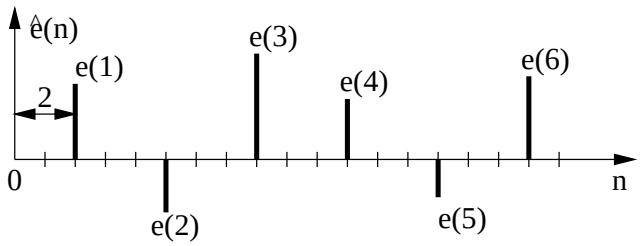

Obrázek 8.21: Póly přenosové funkce $\frac{1}{A(z)}$ a $\frac{1}{A(z/\gamma)}$.

### 8.4.5 Kódování buzení v kratších rámcích

Zatímco LP analýza probíhá v rámcích "obvyklé" délky (běžně 20 ms – 160 vzorků pro $F_s = 8\ kHz$), buzení je často kódováno v kratších rámcích - typicky 40 vzorků. Následující dva odstavce presentují dvě široce používané metody kódování buzení.

### 8.4.6 Regular-Pulse Excitation, Long Term Prediction (RPE-LTP)

Tento postup je používán v mobilních telefonem GSM bez EFR (Enhanced Full Rate). Buzení je v rámcích o 40 vzorcích kódováno tak, že je chybový signál podvzorkován, a je kvantována pouze poloha prvního impulsu (nultý, první, druhý) a pak velikosti jednotlivých impulsů (Obr. 8.22). Jedná se tedy o buzení pravidelnými impulsy (regular pulses).

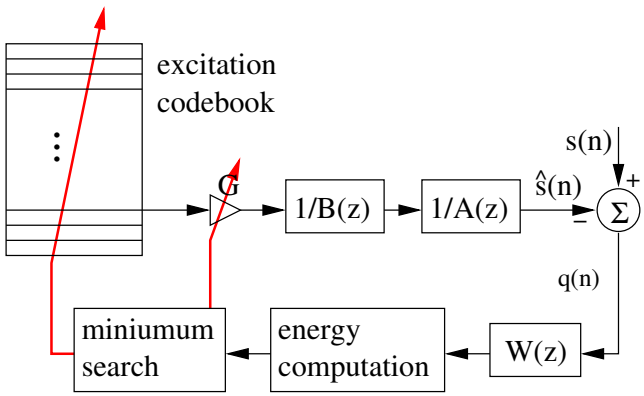



Obrázek 8.22: Budící impulsy u RPE-LTP.

### 8.4.7 Codebook-Excited Linear Prediction CELP

DEF

U této metody, používané např. v mobilních telefonem GSM s EFR (tedy ve většině Vašich přístrojů), je buzení kódováno pomocí vektorového kvantování. Výpočet energie a výběr minima řídí výběr optimálního buzení z kódové knihy. Základní struktura s perceptuálním filtrem je na Obr. 8.23.




Obrázek 8.23: Výběr optimálního buzení v kódové knize CELP.

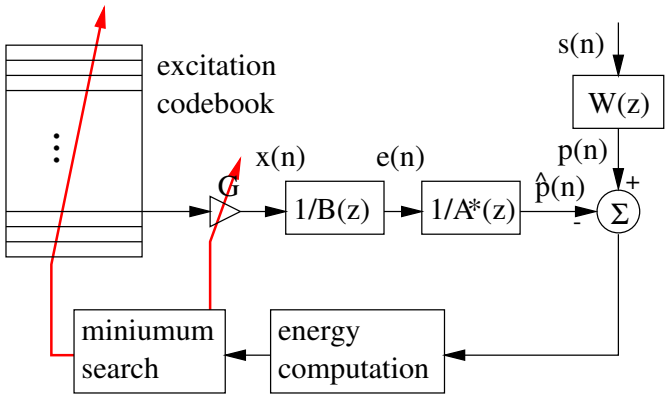
#### Přesun perceptuálního filtru

Vidíme, že každý testovaný signál musí být filtrován $W(z) = \frac{A(z)}{A^\star(z)}$, což představuje moc práce (mnoho CPU operací). Filtrování je ovšem lineární, takže $W(z)$ můžeme přesunout do obou větví: na vstup a do signálu za sekvencí $\frac{1}{B(z)}$ a $\frac{1}{A(z)}$. Můžeme zjednodušit na: $\frac{1}{A(z)}\frac{A(z)}{A^\star(z)} = \frac{1}{A^\star(z)}$, což toto je nový filtr, který musíme použít za $\frac{1}{B(z)}$ (Obr. 8.24)

#### Odstranění odezvy filtru naprázdno.

Filtr $\frac{1}{A^\star(z)}$ neodpovídá jen na buzení, které se zuřivě snažíme najít, ale má i vlastní paměť (příspěvek od minulých rámců). Označíme jeho impulsní odezvu $h(i)$:

$$\hat{p}(n) = \underbrace{\sum_{i=0}^{n-1} h(i)e(n-i)}_{\text{tento rámec}} + \underbrace{\sum_{i=n}^{\infty} h(i)e(n-i)}_{\text{minulé rámce } \hat{p}_0(n)} \tag{8.26}$$




Obrázek 8.24: CELP s perceptuálním filtrem ve vstupní větvi a s modifikovaným filtrem ve větvi generování dekódovaného signálu.

Odezva od minulých rámců nezávisí na tom, co teď hledáme – můžeme ji spočítat jen jednou a odečíst od porovnávaného signálu (vstup filtrovaný $W(z)$), dostaneme:

$$\hat{p}(n) - \hat{p}_0(n) = \sum_{i=0}^{n-1} h(i)e(n-i) \tag{8.27}$$

V dalším postupu vezmeme v úvahu long-term predictor:

$$\frac{1}{B(z)} = \frac{1}{1 - bz^{-M}}, \tag{8.28}$$

kde $M$ je optimální lag, můžeme zapsat $e(n)$ jako

$$e(n) = x(n) + be(n-M), \tag{8.29}$$

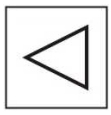
v rovnici pro filtrování dostaneme místo $e(n-i)$:

$$\hat{p}(n) - \hat{p}_0(n) = \sum_{i=0}^{n-1} h(i)[x(n-i) + be(n-M-i)]. \tag{8.30}$$

to dále rozložíme:

$$\hat{p}(n) - \hat{p}_0(n) = \sum_{i=0}^{n-1} h(i)x(n-i) + b\sum_{i=0}^{n-1} h(i)e(n-M-i), \tag{8.31}$$

protože filtrování je lineární operace. Druhý výraz je ve skutečnosti **minulé buzení** filtrované $\frac{1}{A^\star(z)}$ a násobené LTP gainem $b$. Namísto LTP si v CELP kodéru můžeme představit druhou kódovou knihu, která bude obsahovat minulá buzení. Zde si ji představujeme jako skutečnou kódovou knihu, s řádky $e(n-M)$ (nebo $\mathbf{e}^{(M)}$ ve vektorové notaci), v reálných aplikacích je to prostě kus zpožděného budícího signálu.



Výsledný tvar rovnic pro CELP je:

$$\hat{p}(n) - \hat{p}_0(n) = \sum_{i=0}^{n-1} h(i)e(n-i) = \tag{8.32}$$

$$= \sum_{i=0}^{n-1} h(i)\left[g^{(j)}u^{(j)}(n-i) + be(n-i-M)\right] = \tag{8.33}$$

$$= \hat{p}_2(n) + \hat{p}_1(n), \tag{8.34}$$

**Nalezení parametrů pro kódování**

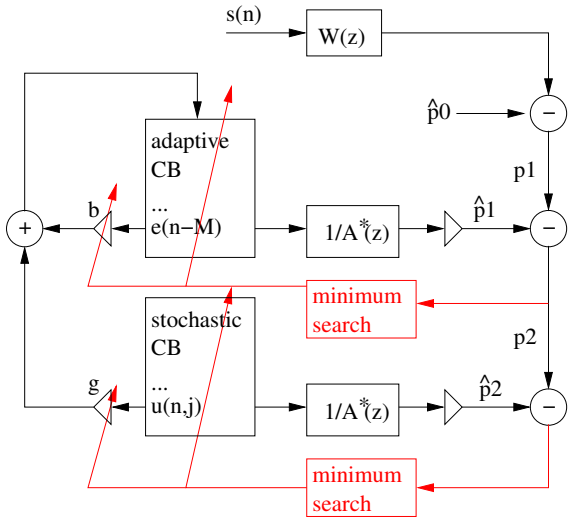
Úkoly jsou následující:

- musíme najít gain pro stochastickou kódovou knihu $g$
- nejlepší vektor ze stochastické kódové knihy $\mathbf{u}^{(j)}$
- gain adaptivní kódové knihy $b$
- nejlepší vektor z adaptivní kódové knihy $\mathbf{e}^{(M)}$.

Teoreticky bychom měli zkoušet všechny kombinace, což by bylo velmi náročné, použijeme tedy suboptimální proceduru:

- nejprve nalezneme $\hat{p}_1(n)$ hledáním v adaptivní kódové knize. Výsledkem je lag $M$ a gain $b$.
- Odečteme $\hat{p}_1(n)$ od $p_1(n)$ a dostaneme "chybový signál druhé generace", který by nám měla vygenerovat stochastická kódová kniha.
- Najdeme $\hat{p}_2(n)$ hledáním ve stochastické kódové knize. Výsledkem je index $j$ a gain $g$.
- Na konci se obvykle jen re-optimalizují gainy (příspěvky obou kódových knih).

Tyto operace reflektuje finální struktura CELP kodéru na Obr. 8.25.




Obrázek 8.25: Skutečná struktura kodéru CELP s adaptivní a stochastickou kódovou knihou.

**Varianty CELP**

Kódování CELP existuje mnoho různých variant, které se liší použitými kódovými knihami a jejich prohledáváním. Standardem jsou dvě kódové knihy, z nichž je jedna adaptivní (je plněna minulými rámci buzení) a jedna stochastická. Při generování buzení se sečtou vektory z obou kódových knih vynásobené různými koeficienty (gainy).

Pro urychlení prohledávání se kódové knihy používají různě strukturované. GSM-EFR používá algoritmus ACELP, kde 'A' značí 'algebraic'. Stochastická kódová kniha tu tedy není zcela náhodná, ale je organizována tak, aby se urychlilo hledání optimálního buzení. Tento algoritmus produkuje bitový tok 12.2 kbit/s.

GSM-HR (half-rate, 6.5 kbit/s) využívá algoritmu VS-CELP (vector-sum). Kódová kniha je zde zkonstruována pomocí několika ortogonálních bázových vektorů, při konstrukci buzení se sčítají s různými násobícími koeficienty. Opět se tak dá podstatně urychlit vyhledávání.

## 8.5 Příklady kodérů

### 8.5.1 GSM full-rate ETSI 06.10 – RPE-LTP

Schémata kodéru a dekodéru jsou na Obr. 8.26 a 8.27. Základní údaje:

- Krátkodobá analýza (v rámcích 20 ms 160 vzorků), koeficienty LPC filtru převedeny na 8 LAR.
- dlouhodobou analýzou (LTP) v rámcích 5 ms (40 vzorků) dostaneme lag a gain.
- Buzení kódováno v rámcích o 40 vzorcích kódováno tak, že je chybový signál podvzorkován s faktorem 3 (14,13,13), a je kvantována pouze poloha prvního impulsu (0,1,2,3(!))
- velikosti jednotlivých impulsů (Obr. 8.22) jsou kvantovány pomocí adaptivní PCM.
- výsledkem je rámec o 260 bitech $\times$ 50 = 13 kbit/s.

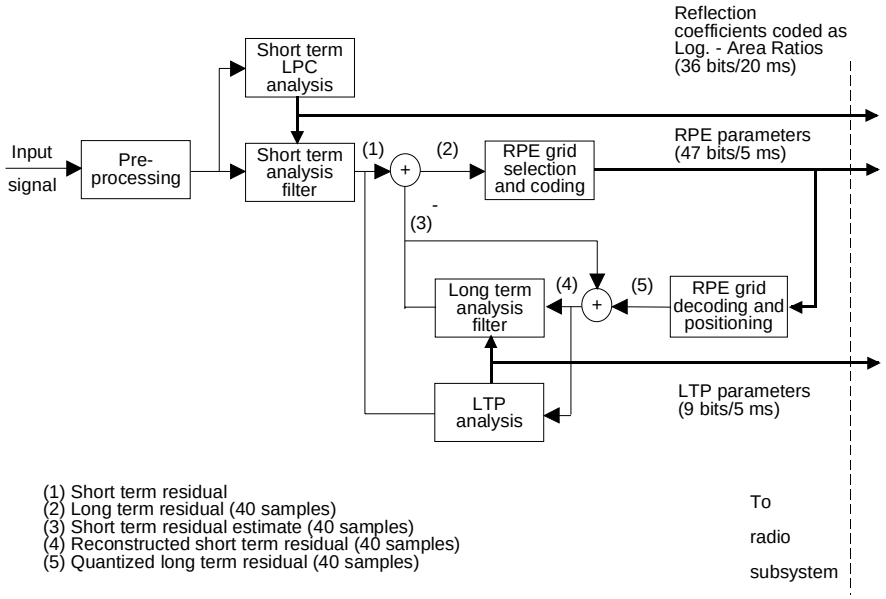
Více se dozvíte z normy 06.10, ke stažení z `http://pda.etsi.org`

### 8.5.2 GSM enhanced full-rate (EFR) ETSI 06.60 – ACELP

Tento kodér (Obr. 8.28 a 8.29) je typu CELP s "inteligentní" algebraickou kódovou knihou. Základní parametry:

- opět rámce 20 ms (160 vzorků).
- Krátkodobý prediktor – 10 koeficientů $a_i$ ve dvou sub-rámcích, převedeny na line-spectral pairs, ze dvou sub-rámců společně kvantovány pomocí split-matrix quantization (SMQ).
- 4 sub-rámce po 40 vzorcích (5 ms) pro buzení.
- Odhad lagu nejprve open-loop, potom closed-loop okolo hrubého odhadu, fractional pitch s rozlišením 1/6 vzorku.
- stochastický codebook: algebraická kódová kniha - může obsahovat pouze 10 nenulových impulsů, které mohou být pouze +1 nebo -1. To vede k rychlému vyhledávání (rychlé korelace - pouze sčítání, ne násobení), atd.
- 244 bitů na rámec $\times$ 50 = 12.2 kbit/s.

Více viz norma 06.60, ke stažení z `http://pda.etsi.org`




Obrázek 8.26: Kodér RPE-LTP.

## 8.6 Čtení a materiály o kódování řeči

- Na WWW stránce Andrease Spaniase z Arizona University:
  `http://www.eas.asu.edu/~spanias`
  v sekci Publications/Tutorial Papers je ke stažení výborná přehledová práce: "Speech Coding: A Tutorial Review", z níž byla část otištěna v Proceedings of the IEEE, Oct. 1994.
- na téže stránce v sekci Software/Tools/Demo - Matlab Speech Coding Simulations naleznete software pro FS1015, FS1016, RPE-LTP a další. Příjemné hraní.
- Normy ETSI pro mobilní telefony jsou zdarma ke stažení z:
  `http://pda.etsi.org/pda/queryform.asp`
  Jako klíčová slova můžete zadat např "gsm half rate speech". K mnoha normám jsou k disposici také zdrojové kódy kodérů v jazyce C.

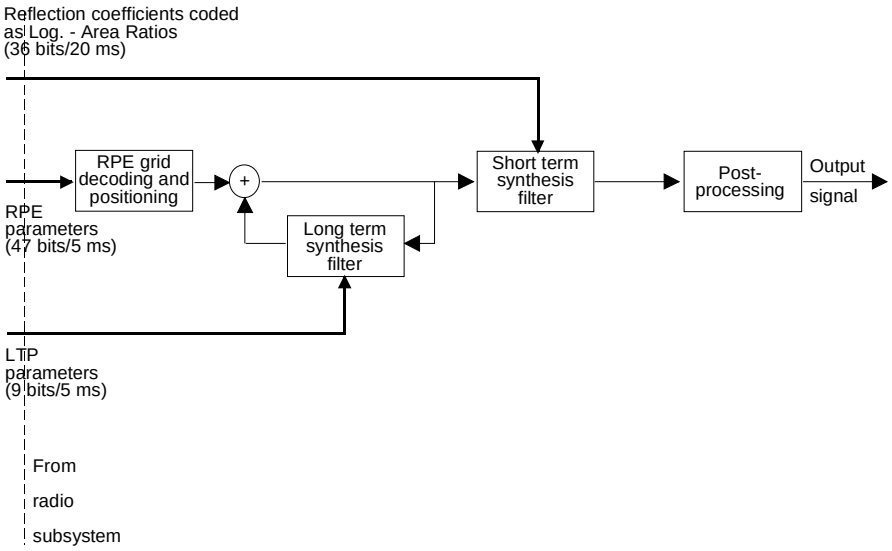
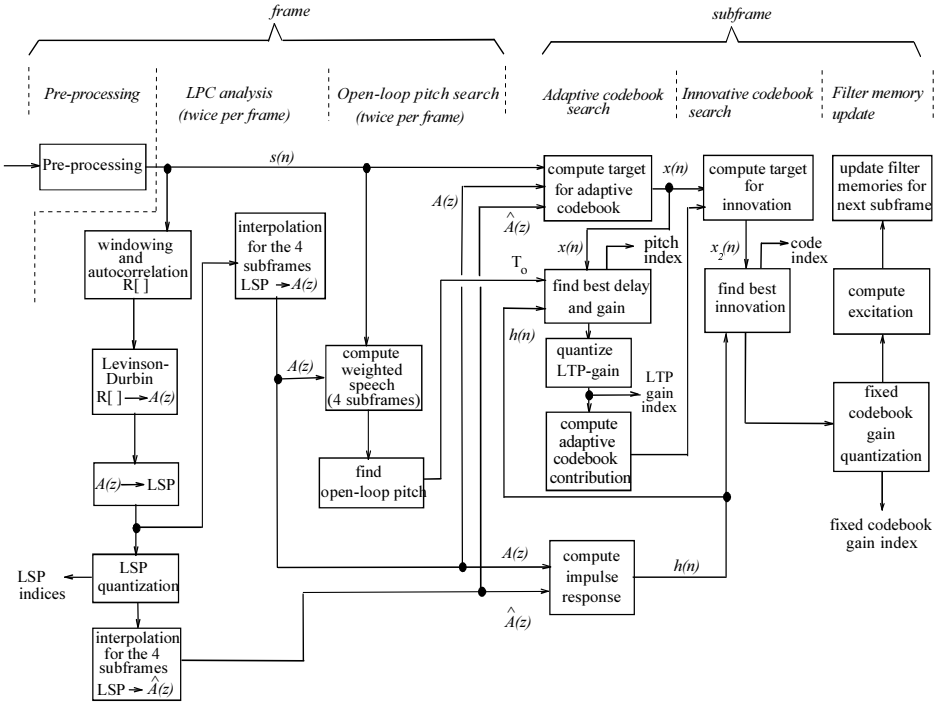
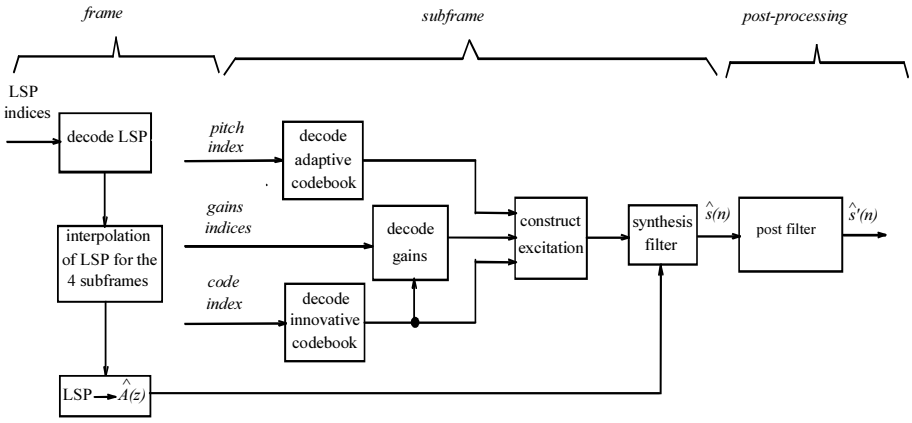



**Figure 1.2: Simplified block diagram of the RPE - LTP decoder**

Obrázek 8.27: Dekodér RPE-LTP.




Obrázek 8.28: Kodér ACELP.




Obrázek 8.29: Dekodér ACELP.

# Kapitola 9

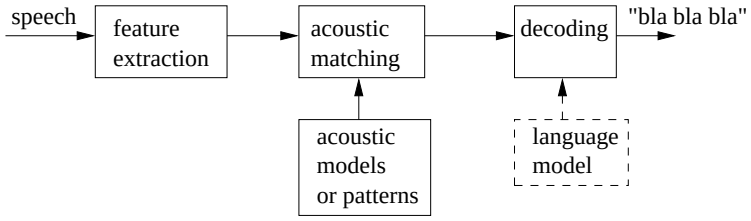
# Úvod do rozpoznávání řeči


Úkolem pro rozpoznávání řeči (speech recognition) je zjistit, **co bylo řečeno**. Podle složitosti klasifikujeme systémy pro rozpoznávání do těchto kategorií:

- izolovaná slova – ovládání mobilních telefonů hlasem. Potřebují voice activity detector nebo push-to-talk.
- spojená slova (omezený slovník) – např. číslovky při zadávání telefonního čísla nebo čísla kreditní karty. Rozpoznávání je většinou řízeno nějakou sítí nebo jednoduchou gramatikou.
- plynulá řeč s velkým slovníkem (large vocabulary continuous speech recognition LVCSR) – nejtěžší úkol, potřebuje informace o akustice, ale také o struktuře jazyka (jazykový model - language model) a výslovnostní slovník (pronunciation dictionary). Pracují s menšími jednotkami než se slovy (60 tisíc slov se nedá naučit. . . ). Využívají fonémy, kontextově závislé fonémy.

## 9.1 Struktura rozpoznávače

je na Obr. 9.1.



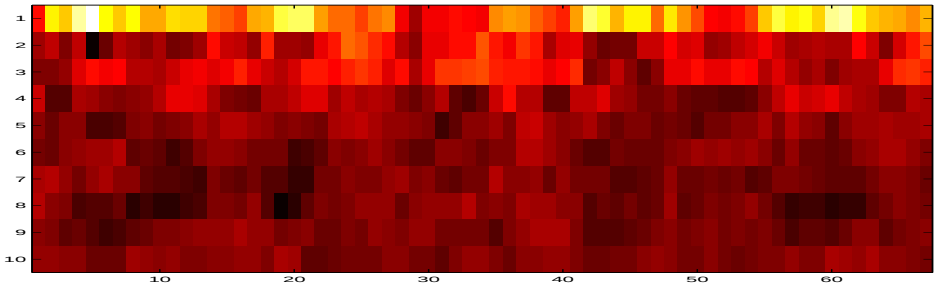

Obrázek 9.1: Typická struktura rozpoznávače řeči.

### 9.1.1 Parametrisace – feature extraction

Má za úkol

- omezení množství dat.
- "odrušení" složek, které nás nezajímají (pitch, střední hodnota, fáze)
- parametry musí být dobré pro rozpoznávač (např. rozdíl dvou vektorů by měl mít nějaký smysl, nebo nekorelovanost).

Paremetrisace většinou využívá spektrální analýzu (Mel-frekvenční cepstrální koeficienty) nebo LPC analýzu (LPC-cepstrum). Jejím výsledkem je sekvence vektorů $\mathbf{O} = [\mathbf{o}(1), \mathbf{o}(2), \ldots, \mathbf{o}(T)]$, které si můžeme představit jako velkou matici (Obr. 9.2).

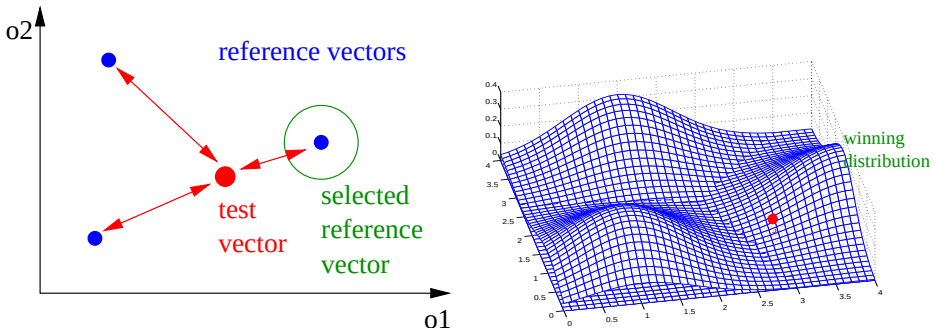



Obrázek 9.2: Parametry na vstupu rozpoznávače.

### 9.1.2 Akustické zpracování – acoustic matching

Má za úkol vyrovnat se se dvěma zdroji variability:

1. v **prostoru parametrů** - člověk nikdy neřekne jednu věc stejně, vektory parametrů se tedy **vždy liší**. Metody fungující pro text, kde existuje 1-1 korespondence, nebudou fungovat. Obr. 9.3 ukazuje, jak se v rozpoznávání řeči s touto variabilitou vyrovnáváme:
   (a) Měřením vzdálenosti mezi vektory.
   (b) Statistickým modelováním.

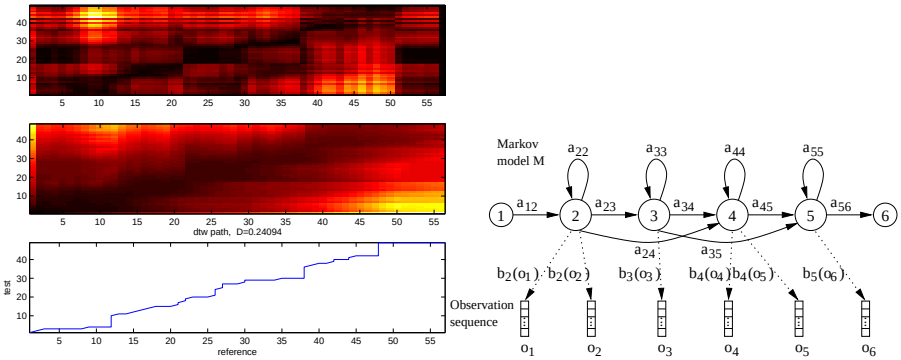



Obrázek 9.3: Variabilita v prostoru parametrů: měření vzdáleností mezi vektory a statistické modelování.

2. v **čase** – lidé nikdy neřeknou jednu věc se stejným časováním. S touto variabilitou se vyrovnáváme (Obr. 9.4) pomocí:
   (a) srovnávacích cest (máme-li k disposici referenční matici vektorů pro srovnání)
   (b) stochastického automatu, kde budou jednotlivé stavy schopny zpracovat proměnná množství vstupních vektorů

### 9.1.3 Dekódování

V případě izolovaných je velmi jednoduché, jedná se jen o výběr maxima pravděpodobnosti nebo minima vzdálenosti). U LVCSR je mnohem složitější –




Obrázek 9.4: Variabilita v čase: srovnávací cesta v metodě DTW a stochastický stavový automat v HMM.

dekodér musí pracovat se složitými akustickými modely (trifóny), modelem jazyka (language model) a výslovnostním slovníkem. Používá se Viterbiho algoritmus, A*-search, best-first decoding, konečné stavové automaty, omezení prohledávacího prostoru (beam-search), atd.

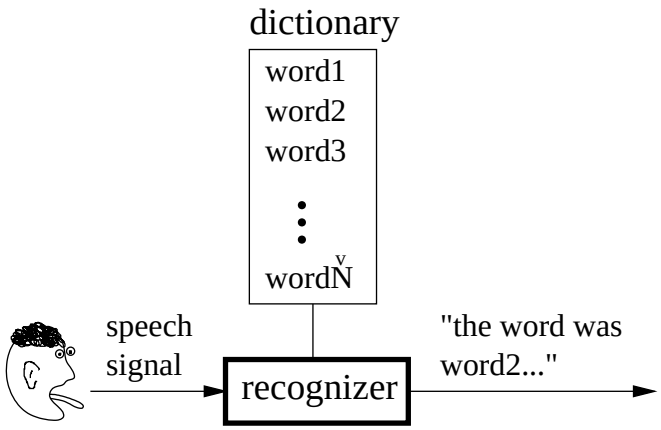
## 9.2 Rozpoznávání izolovaných slov

V následujících kapitolách budeme demonstrovat dvě techniky na rozpoznávání izolovaných slov. Bude tedy dobré si tuto úlohu definovat.

Ve vstupní řeči je nutné nejprve izolovaná slova najít:

- můžeme použít push-to-talk knoflík (zdá se Vám možná zastaralý, ale rozpoznávání v autech se dělá právě takto, i když se z komerčních důvodů používá spíše termín "push to activate")...
- detekovat řečovou aktivitu – např. detektorem založeným na energii.

Rozpoznávač má za úkol klasifikovat příchozí matici parametrů $\mathbf{O} = [\mathbf{o}(1), \ldots, \mathbf{o}(T)]$ jako jedno ze slov $w_1 \ldots w_{\check{N}}$, kde $\check{N}$ je počet slov (viz Obr. 9.5). V následující kapitole 10 uvidíme, že rozpoznávání lze provádět tak, že ve slovníku rozpoznávače uložíme přímo matice parametrů (tzv. reference) pro jednotlivá slov. Kapitola 11 pak pojednává o tom, jak lze taková slova modelovat statistickými modely.




Obrázek 9.5: Úkol rozpoznávače izolovaných slov.

# Kapitola 10

# Dynamické borcení času – DTW


Tato kapitola by se měla spíše jmenovat "rozpoznávání pomocí srovnávání s referencemi", ale jelikož podstatným úkolem je časové srovnání, označuje se takové rozpoznávání často zkratkou této srovnávací techniky: DTW.

Při srovnávání vstupní matice $\mathbf{O} = [\mathbf{o}(1), \ldots, \mathbf{o}(T)]$ máme ve slovníku k disposici **referenční** matice parametrů pro slova, která chceme rozpoznávat:

$$\mathbf{R}_1 \ldots \mathbf{R}_{\check{N}}$$

Na vstup rozpoznávače přijde **testovací** matice parametrů $\mathbf{O}$ a my chceme určit, ke kterému referenčnímu slovu test patří.

## 10.1 Problémy srovnání referenční a testovací matice

Kdyby měla slova jen jeden vektor, bylo by srovnání jednoduché – stačilo by určit libovolnou vzdálenost mezi dvěma vektory, nejpoužívanější z nich je Euklidova:

$$d(\mathbf{o}, \mathbf{r}_i) = \sqrt{\sum_{k=1}^{P} |o(k) - r_i(k)|^2}.$$

Pak by se vybrala minimální vzdálenost a ta by definovala rozpoznané slovo.

Slova Bohu žel pouze jeden vektor **nemají**: Potřebujeme určit *vzdálenost* (či *podobnost*) referenční sekvence vektorů o délce $R$:

$$\mathbf{R} = [\mathbf{r}(1), \ldots, \mathbf{r}(R)]$$

a testovací sekvence vektorů o délce $T$:

$$\mathbf{O} = [\mathbf{o}(1), \ldots, \mathbf{o}(T)]$$

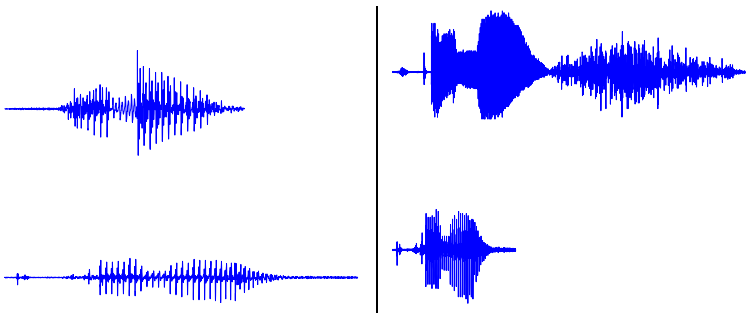
Bylo by možné sečítat vzdálenosti jednotlivých vektorů přes celé slovo ? Jakých ? Jak ? Slova **nejsou nikdy stejně dlouhá** $R \neq T$.

### 10.1.1 Lineární srovnání

U jednoho ze slov (například testovacího) použijeme transformační funkci času, která je "natáhne" na druhé slovo:

$$D(\mathbf{O}, \mathbf{R}) = \sum_{i=1}^{R} d[\mathbf{o}(w(i)), \mathbf{r}(i)] \tag{10.1}$$

kde $w(i)$ je definována tak, aby srovnání bylo lineární. V levé části Obr. 10.1 tato technika ještě bude fungovat a podaří se jí referenční a testovací slovo zarovnat. V pravé části téhož obrázku ale došlo k chybě určení hranic referenčního slova a vidíme, že při lineárním srovnání by se celá druhá polovina testovacího slova "přetáhla"přes šum přítomný v referenčním slově. Algoritmus by tedy vyprodukoval velkou vzdálenost, i když se jedná o stejné slovo!



Obrázek 10.1: Lineární srovnání referenčního (nahoře) a testovacího (dole) slova. V levé části by se lineární zarovnání povedlo, v pravé ne (přítomnost dlouhého šumu v referenčním slově).

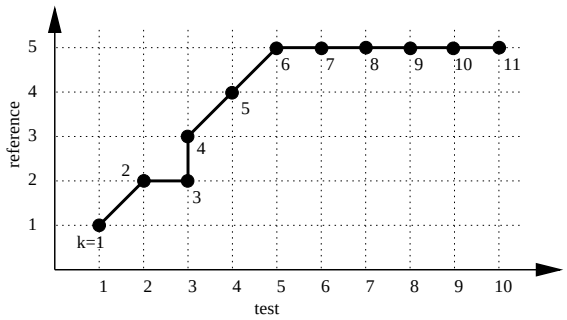
## 10.2 Dynamické borcení času

V této technice je srovnání *řízeno* přímo vzdálenostmi jednotlivých vektorů, hovoříme o **Dynamickém borcení času**. Definujeme obecnou časovou proměnnou $k$ a a zavedeme *dvě* transformační funkce:

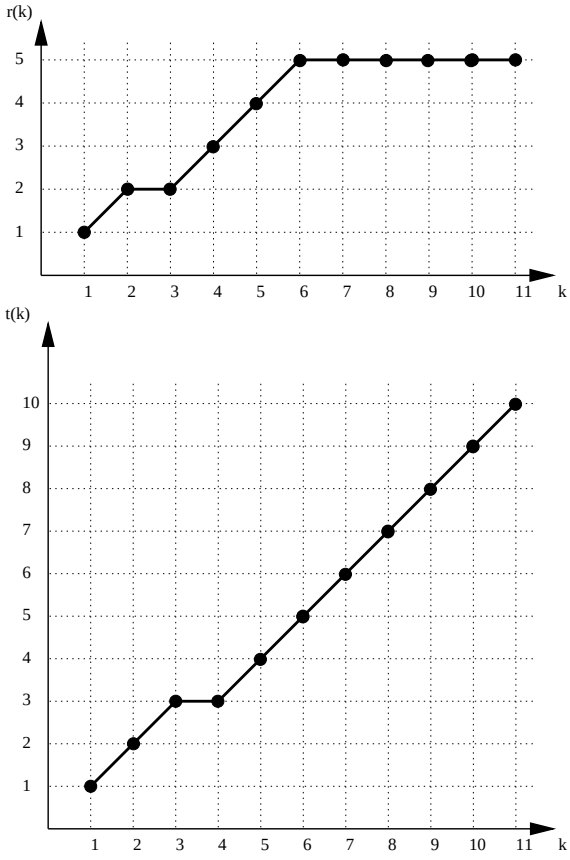
DEF

- $r(k)$ pro referenční sekvenci.
- $t(k)$ pro testovací sekvenci.

Pak můžeme srovnání jednotlivých vektorů zakreslit pomocí **cesty** (Obr. 10.2). Počet kroků cesty označíme jako $K$. Referenci budeme zobrazovat na svislé ose, test na vodorovné. Z této cesty můžeme odvodit průběh funkcí $r(k)$ a $t(k)$, které "krokují" jednotlivé sekvence, viz Obr. 10.3.




Obrázek 10.2: Cesta pro srovnání dvou sekvencí.




Obrázek 10.3: Funkce $r(k)$ a $t(k)$ pro krokování referenční a testovací sekvence.

Cesta $C$ je jednoznačně dána svou délkou $K_C$ a průběhem funkcí $r_C(k)$ a $t_C(k)$. Pro tuto cestu je vzdálenost sekvencí $\mathbf{O}$ a $\mathbf{R}$ dána jako:

$$D_C(\mathbf{O},\mathbf{R}) = \frac{\sum_{k=1}^{K_C} d[\mathbf{o}(t_C(k)),\mathbf{r}(r_C(k))]W_C(k)}{N_C} \tag{10.2}$$

kde $d[\mathbf{o}(\cdot),\mathbf{r}(\cdot)]$ je vzdálenost dvou vektorů, $W_C(k)$ je váha odpovídající $k$-tému kroku cesty a $N_C$ je normalisační faktor závislý na vahách.

Vzdálenost sekvencí $\mathbf{O}$ a $\mathbf{R}$ je dána jako *minimální vzdálenost* přes soubor všech možných cest (všechny možné délky, všechny možné průběhy):

$$D(\mathbf{O},\mathbf{R}) = \min_{\{C\}} D_C(\mathbf{O},\mathbf{R}). \tag{10.3}$$

Je třeba vyřešit 3 věci:


1. přípustné průběhy funkcí $r(k)$ a $t(k)$. Není možné, aby se cesta vracela zpět, "skákala" přes několik vektorů, atd.
2. definovat váhovací funkci a normalisační faktor.
3. vyrobit algoritmus, který vypočítá $D(\mathbf{O},\mathbf{R})$ co nejrychleji.

### 10.2.1 Omezení cesty

1. **Počáteční a koncové body**

$$\left.\begin{matrix} r(1)=1 \\ t(1)=1 \end{matrix}\right\} \text{začátek} \qquad \left.\begin{matrix} r(K)=R \\ t(K)=T \end{matrix}\right\} \text{konec} \tag{10.4}$$

2. **Lokální souvislost a lokální strmost**

$$\begin{matrix} 0 \leq r(k) - r(k-1) \leq R^\star \\ 0 \leq t(k) - t(k-1) \leq T^\star \end{matrix} \tag{10.5}$$

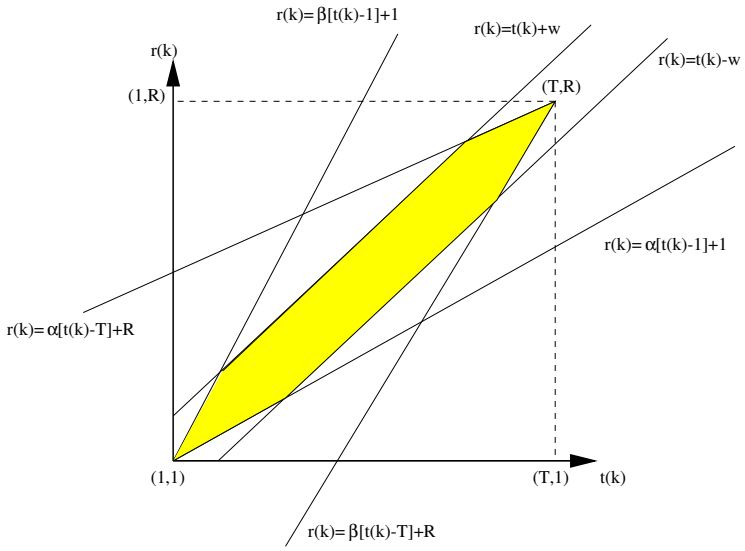
   v praxi se volí $R^\star, T^\star$=1,2,3.

   - $R^\star, T^\star$=1: Každý vektor se musí vzít alespoň jedenkrát. $r(k) = r(k-1)$ znamená, že se opakuje.
   - $R^\star, T^\star > 1$: Vektor(y) se mohou přeskočit.

3. **Globální vymezení cesty DTW:** vymezení přípustné oblasti pomocí přímek:

$$\begin{matrix} 1 + \alpha[t(k) - 1] \leq r(k) \leq 1 + \beta[t(k) - 1] \\ R + \beta[t(k) - T] \leq r(k) \leq R + \alpha[t(k) - T] \end{matrix} \tag{10.6}$$

   Tyto podmínky vymezují maximální "strmost" nebo "placatost" cesty DTW – viz Obr. 10.4

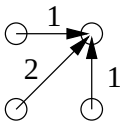



Obrázek 10.4: Globální omezení cesty DTW. Povolená oblast, kde se může cesta nacházet, je vyznačena žlutě.

### 10.2.2 Definice váhových funkcí

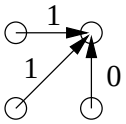
Váhová funkce $W(k)$ závisí na lokálním "posunu" cesty. 4 typy:

- **typ a)** symetrická: $W_a(k) = [t(k) - t(k-1)] + [r(k) - r(k-1)]$.

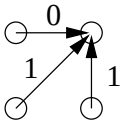



- **typ b)** asymetrická:

b1) $W_{b1}(k) = t(k) - t(k-1)$




b2) $W_{b2} = r(k) - r(k-1)$




- **typ c)** $W_c(k) = \min\{t(k) - t(k-1), r(k) - r(k-1)\}$
- **typ d)** $W_d(k) = \max\{t(k) - t(k-1), r(k) - r(k-1)\}$

### 10.2.3 Normalisační faktor

$$N = \sum_{k=1}^{K} W(k) \tag{10.7}$$

Pro váhovací funkci typu a) je normalisační faktor:

$$N_a = \sum_{k=1}^{K} [t(k) - t(k-1) + r(k) - r(k-1)] = t(K) - t(0) + r(K) - r(0) = T + R \tag{10.8}$$

Pro váhovací funkci typu b1) je normalisační faktor $N = T$.
Pro váhovací funkci typu b1) je normalisační faktor $N = R$.
Pro typy c), d) faktor silně závislý na průběhu cesty, je lepší použít konstantu: $N = T$.

### 10.2.4 Lokální omezení cesty

Tabulka 10.1 udává typy lokálních omezení cesty a z nich vyplývající faktory $\alpha$ a $\beta$. Význam veličiny $g(n, m)$ bude vysvětlen později.

## 10.3 Efektivní výpočet $D(\mathbf{O}, \mathbf{R})$

Výpočet minimální vzdálenosti

$$D(\mathbf{O}, \mathbf{R}) = \min_{\{C\}} D_C(\mathbf{O}, \mathbf{R}). \tag{10.9}$$

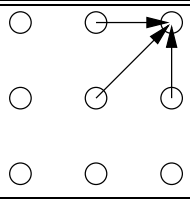
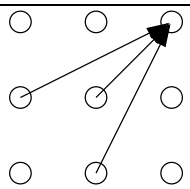
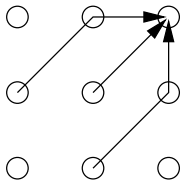
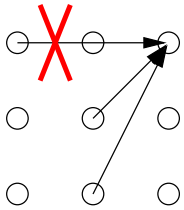
je jednoduchý, pokud normalisační faktor $N_C$ není funkcí cesty a můžeme psát:

$$N_C = N \quad \text{pro} \quad \forall C$$

Toto naštěstí většinou platí a tedy:

$$D(\mathbf{O}, \mathbf{R}) = \frac{1}{N} \min_{\{C\}} \sum_{k=1}^{K_C} d[\mathbf{o}(t_C(k)), \mathbf{r}(r_C(k))] \tag{10.10}$$

Rozhodnutí o průběhu optimální cesty si nemusíme nechávat až na konec, ale můžeme se rozhodnout pro každou dvojici hodnot $t$ a $r$. Až dojdeme na konec obou promluv (časy $T$ a $R$), budeme mít k disposici vzdálenost pro optimální cestu. Postup je následující:

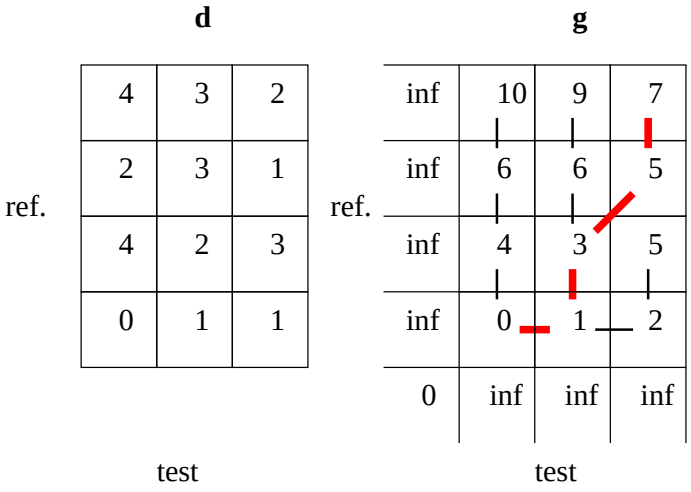
| Typ DTW | | $\alpha$ | $\beta$ | Typ $w(k)$ | $g(n,m)$ |
|---|---|---|---|---|---|
| I. |  | 0 | $\infty$ | a | $\min \begin{cases} g(n,m-1)+d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-1,m)+d(n,m) \end{cases}$ |
| | | | | d | $\min \begin{cases} g(n,m-1)+d(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-1,m)+d(n,m) \end{cases}$ |
| II. |  | $\frac{1}{2}$ | 2 | a | $\min \begin{cases} g(n-1,m-2)+3d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-2,m-1)+3d(n,m) \end{cases}$ |
| | | | | d | $\min \begin{cases} g(n-1,m-2)+d(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-2,m-1)+d(n,m) \end{cases}$ |
| III. |  | $\frac{1}{2}$ | 2 | a | $\min \begin{cases} g(n-1,m-2)+2d(n,m-1)+d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-2,m-1)+2d(n-1,m)+d(n,m) \end{cases}$ |
| IV. |  | $\frac{1}{2}$ | 2 | b1 | $\min \begin{cases} g(n-1,m)+kd(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-1,m-2)+d(n,m) \end{cases}$ kde $k=1$ pro $r(k-1)\neq r(k-2)$ $k=\infty$ pro $r(k-1)=r(k-2)$ |

Tabulka 10.1: Typy lokálních omezení cesty a z nich vyplývající faktory $\alpha$ a $\beta$. $g(n,m)$ jsou vztahy pro výpočet částečné kumulované vzdálenosti. U typu IV není povolen pohyb dvakrát za sebou jen ve směru $t$.

1. do mřížky **d** o velikosti $T \times R$ si zapíšeme vzdálenosti referenčních a testovacích vektorů, každý s každým, viz Příklad.
2. definujeme mřížku **g** s *částečnou kumulovanou vzdáleností*. Oproti mřížce **d** má **g** navíc ještě nultý řádek a nultý sloupec, které inicialisujeme na:
$$g(0,0)=0, \quad \text{a} \quad g(0,m\neq 0)=g(n\neq 0,0)=\infty.$$
3. Částečnou kumulovanou vzdálenost spočítáme pro každý bod takto:
$$g(m,n)=\min_{\forall\text{předchůdci}}[g(\text{předchůdce})+d(m,n)w(k)] \qquad (10.11)$$
   - možní předchůdci jsou dáni pomocí tabulky lokálních omezení cesty.
   - váha $w(k)$ odpovídá pohybu z předchůdce do bodu $[m,n]$.
   - vztahy pro výpočet Částečné kumulované vzdálenosti jsou tabelovány v Tabulce 10.1.
4. Konečná minimální normovaná vzdálenost je pak dána:
$$D(\mathbf{O},\mathbf{R})=\frac{1}{N}g(T,R) \qquad (10.12)$$

## Příklad

Viz Obrázek 10.5 Výsledek:




Obrázek 10.5: Příklad. **d** je mřížka lokálních vzdáleností. **g** je mřížka částečných kumulovaných vzdáleností.

- máme vzdálenost $D = \frac{1}{3+4}7 = 1$.
- můžeme zpětně "odkrokovat optimální cestu" (cesta má 5 kroků): $t(k) = [1\ 2\ 2\ 3\ 3]$, $r(k) = [1\ 1\ 2\ 3\ 4]$.

## 10.4 Realisace rozpoznávače založeného na DTW

Opakování: co vlastně chceme ? Přijde slovo **O**; chceme ho zatřídit do třídy $\omega_r$. Máme $\check{N}$ tříd, které odpovídají slovům (např. "jedna", "dvě", "křížek", atd.).

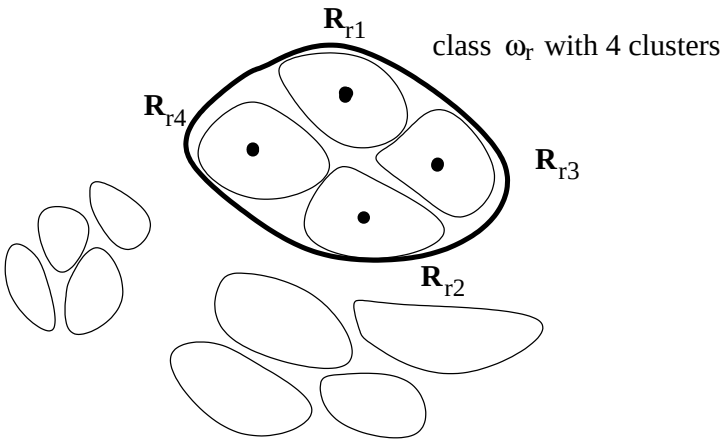
### 10.4.1 Trénování – tvorba referencí neboli vzorů

pří trénování máme sekvence od jednoho nebo více řečníků, a víme, kam která patří.

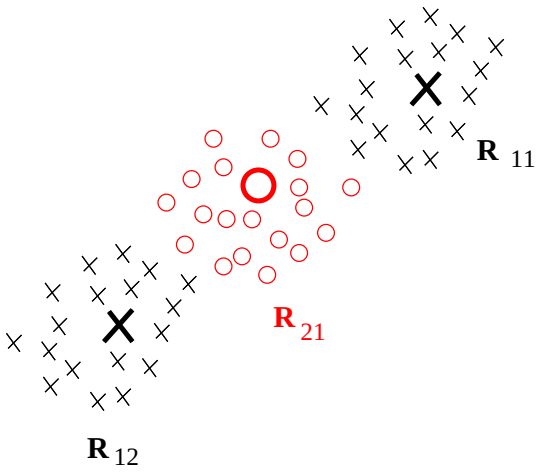
1. **nejjednodušší:** ve třídě $\omega_r$ máme jen jednu referenci $\mathbf{R}_r$.
2. **složitější:** pro každou třídu $\omega_r$ máme více referencí: $\mathbf{R}_{r,1} \ldots \mathbf{R}_{r,\check{N}_r}$. Tyto můžeme mít ve slovníku uložené tak, jak byly vytvořeny, nebo je lineárně normalisovat na jednotnou délku:

$$\bar{R} = \frac{1}{N} \sum_{r=1}^{N} \left[ \frac{1}{\check{N}_r} \sum_{i=1}^{\check{N}_r} R_{r_i} \right], \tag{10.13}$$

   kde $R_{r_i}$ je délka $i$-tého vzoru třídy $\omega_r$.
3. **vytvoření průměrného vzoru pro každou třídu** $\omega_r$:




Obrázek 10.6: Shluky.




Obrázek 10.7: Příklad úspěšné klasifikace při použití shlukování: v případě průměrování by se vzor třídy $\omega_1$ kryl se vzorem třídy $\omega_2$.

- lineární průměrování – bereme průměr vektorů lineárně srovnaných sekvencí. Nebezpečí: můžeme dostat zcela nesmyslný vzor ...
- dynamické průměrování:
  (a) najdeme vzor s o délkou, která se nám líbí.
  (b) průměrování se děje z vektorů přiřazených k tomuto prvnímu vzoru pomocí cesty DTW.

4. vytváření vzorů *shlukováním*, viz Obrázek 10.6. Shluky jsou tvořeny tak, aby si byly reference uvnitř shluků co nejvíce podobné, mezi shluky co nejméně podobné. Existují automatické algoritmy pro shlukování, např. Mac Queenův: nejprve se všechny reference přiřadí do jednoho shluku. Pak se odštěpí nejvzdálenější od středu, "přeshlukuje se", atd. (analogie s VQ). Shluky jsou representovány *centroidy* $\mathbf{R}_{ri}$. Výhoda shlukování oproti průměrování spočívá v tom, že třídy mohou mít složitější tvar, viz Obr. 10.7.

## 10.5 Rozpoznávání (klasifikace)

Je-li každá třída representována jen jednou referencí, je to jednoduché:

$$\omega_r^\star = \arg\min_r D(\mathbf{O}, \mathbf{R}_r) \quad \text{pro} \quad r = 1, \ldots, N \tag{10.14}$$

Mám-li pro každou třídu více referencí, je možné použít dvě metody:

1. **1-NN** (nearest neighbor) neboli nejbližší soused:

$$\omega_r^\star = \arg\min_{r,i} D(\mathbf{O}, \mathbf{R}_{r_i}) \quad \text{pro} \quad \begin{matrix} r = 1, \ldots, N \\ i = 1, \ldots, N_r \end{matrix} \tag{10.15}$$

2. $k$**-NN** ($k$ nearest neighbors) neboli $k$ nejbližších sousedů:

   - pro každou třídu nejprve spočítám všechny vzdálenosti $D(\mathbf{O}, \mathbf{R}_{ri})$ a seřadím je podle velikosti od nejlepší po nejhorší:

$$D(\mathbf{O}, \mathbf{R}_{r_{(1)}}) \leq D(\mathbf{O}, \mathbf{R}_{r_{(2)}}) \leq \ldots \leq D(\mathbf{O}, \mathbf{R}_{r_{(N_r)}}) \tag{10.16}$$

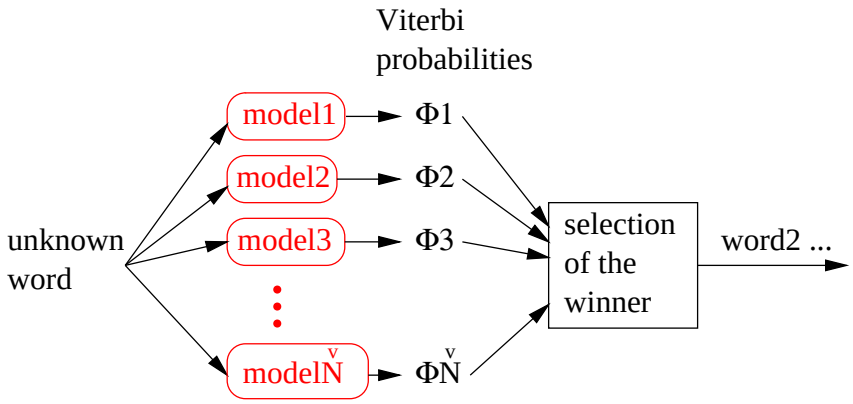
   - o klasifikaci $\mathbf{O}$ do třídy $\omega_r$ se pak rozhodnu podle průměrné vzdálenosti $k$ nejbližších sousedů:

$$\omega_r^\star = \arg\min_r \frac{1}{k} \sum_{i=1}^{k} D(\mathbf{O}, \mathbf{R}_{r_{(i)}}) \tag{10.17}$$

# Kapitola 11

# Skryté Markovovy modely – HMM

Zopakujme si, že úkolem, na kterém budeme vysvětlovat, je rozpoznávání izolovaných slov (viz sekce 9.2). Nechceme-li používat referenční matice, musíme použít statistické modely jednotlivých rozpoznávaných slov. Rozpoznávač pak bude mít základní strukturu podle Obr. 11.1.




Obrázek 11.1: Rozpoznávač se statistickými modely slov.

## 11.1 Stavební bloky rozpoznávače

Pro konstrukci rozpoznávače máme k disposici následující:

### 11.1.1 Teorii statistického rozpoznávání vzorů

má základní rovnici (Bayesův vztah):

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{p(\mathbf{x})}, \tag{11.1}$$

- kde $P(\omega_j|\mathbf{x})$ je posterior třídy $\omega_j$, známe-li datový vektor $\mathbf{x}$,
- $p(\mathbf{x}|\omega_j)$ je likelihood[1] datového vektoru $\mathbf{x}$ známe-li třídu $\omega_j$.
- $P(\omega_j)$ je prior třídy $\omega_j$
- $p(\mathbf{x})$ je evidence (normalizační funkce taková, aby $P(\omega_j|\mathbf{x})$ byla slušná pravděpodobnost).

[1] českým ekvivalentem tohoto anglického termínu je "věrohodnost", ale my budeme používat zažitý anglický termín.

Na evidenci $p(\mathbf{x})$ nehledíme, protože je stejná pro všechny třídy a u jednoduchých rozpoznávačů předpokládáme, že priory všech tříd budou stejné $P(\omega_j) = \omega$.

Proto nám pro nalezení rozpoznaného slova bude stačit hledat maximální likelihood:

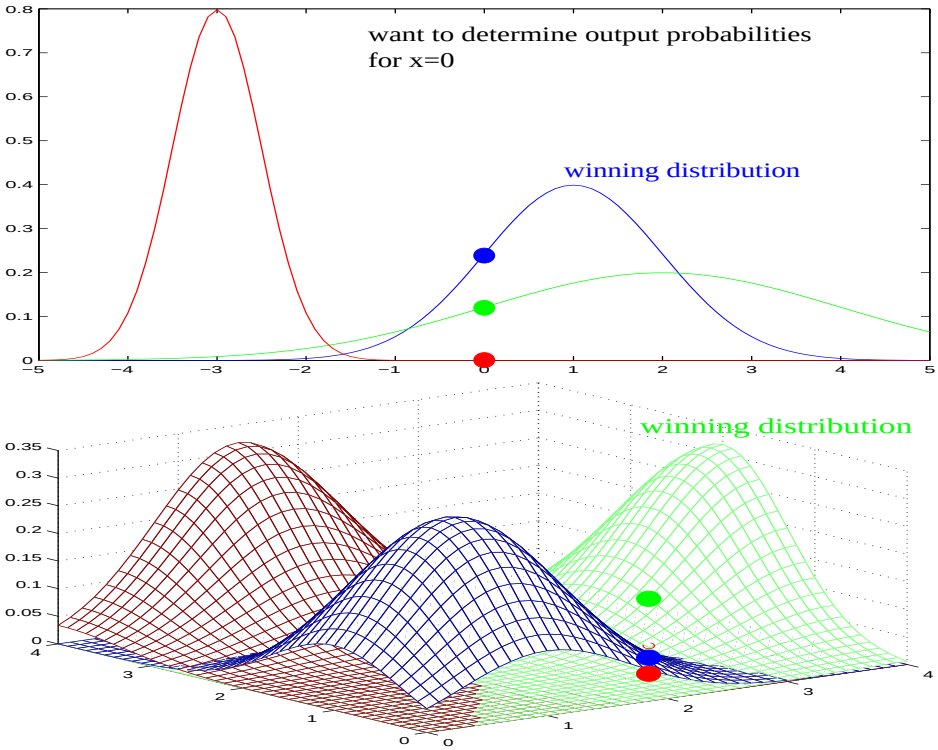
$$\omega_j^\star = \max_j p(\mathbf{x}|\omega_j)$$

### 11.1.2 Modelování jednotlivých tříd

Pro rozpoznávání jednotlivých vektorů dokážeme třídy namodelovat Gaussovými rozděleními nebo dokonce jejich směsí:

$$p(\mathbf{x}|\omega_j) = \sum_{i=1}^{M} \alpha_{ji} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_{ji}),$$

jejichž ilustrace je na Obr. 11.2 a 11.3.

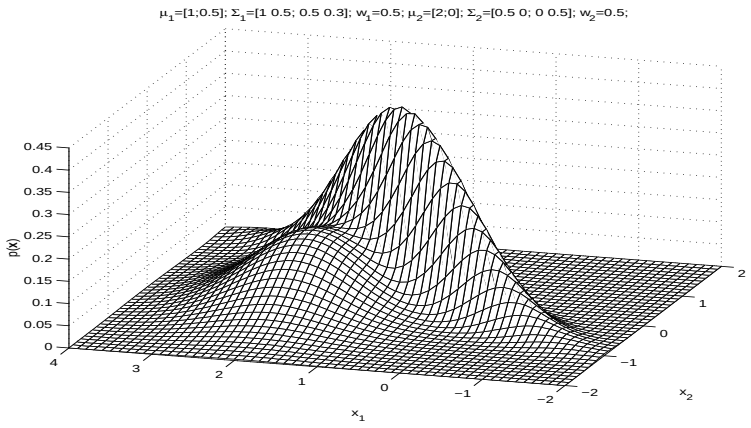



Obrázek 11.2: Ilustrace modelování jednotlivých tříd pomocí jednorozměrných (vlevo) a dvourozměrných (vpravo) Gaussovských rozdělení.

a dokážeme jejich parametry těchto rozdělení $\boldsymbol{\Theta}$ (míchací koeficienty, střední hodnoty a kovarianční matice) odhadnout pomocí iterativního algoritmu EM:

1. parametry na začátku nějak odhadneme (výběr vektoru nebo generování náhodného čísla).
2. s parametry $\boldsymbol{\Theta}^{(i-1)}$ se pro každou Gaussovku spočítá "occupation count":

$$\gamma_l(k) = \frac{\alpha_l^{(i-1)} \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_l^{(i-1)}, \boldsymbol{\Sigma}_l^{(i-1)})}{\sum_{i=1}^{M} \alpha_i^{(i-1)} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i^{(i-1)}, \boldsymbol{\Sigma}_i^{(i-1)})}$$




Obrázek 11.3: Směs Gaussovských rozdělení.

3. a nové parametry jsou dány (omlouvám se za nepřítomnost stříšek ˆ) ...

$$\alpha_l^{(i)} = \frac{1}{n}\sum_{k=1}^{n}\gamma_l(k)$$

$$\boldsymbol{\mu}_l^{(i)} = \frac{\sum_{k=1}^{n}\gamma_l(k)\mathbf{x}_k}{\sum_{k=1}^{n}\gamma_l(k)}$$

$$\boldsymbol{\Sigma}_l^{(i)} = \frac{\sum_{k=1}^{n}\gamma_l(k)(\mathbf{x}_k-\boldsymbol{\mu}_l^{(i)})(\mathbf{x}_k-\boldsymbol{\mu}_l^{(i)})^T}{\sum_{k=1}^{n}\gamma_l(k)}$$

4. opakujeme body 2 a 3 dokud se nesnižuje celková log-likelihood:

$$\ln p(D|\boldsymbol{\Theta}) = \sum_{k=1}^{n}\ln\sum_{i=1}^{M}\alpha_{ji}\mathcal{N}(\mathbf{x}_k;\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_{ji})$$

nebo nás to nepřestane bavit (fixní počet iterací).

### 11.1.3 Od jednoho vektoru k maticím O

V rozpoznávání řeči ovšem nemáme jen jeden vektor, ale celou vstupní sekvenci:

$$\mathbf{O} = [\mathbf{o}(1), \mathbf{o}(2), \ldots, \mathbf{o}(T)], \tag{11.2}$$

a víme, že lidé neřeknou nikdy nic stejně rychle a se stejným časováním (viz diskuse v minulé kapitole). Jak budeme takovou sekvenci vektorů modelovat ?

- Nápad 1: mít 1 gaussovku na slovo, postupně jí předkládat vektory a hodnoty sčítat ? To asi není dobrý nápad, slova "paří" by "řípa"například generovala stejnou likelihood.
- Nápad 2: každé slovo budeme muset representovat **sekvencí více Gaussovek**.
  - jedna nezávislá Gaussovka na každý vektor ? Jak to ale udělat, když je vektorů pokaždé jiný počet ?
    budeme muset definovat **model, kde se Gaussovky budou moci opakovat !**

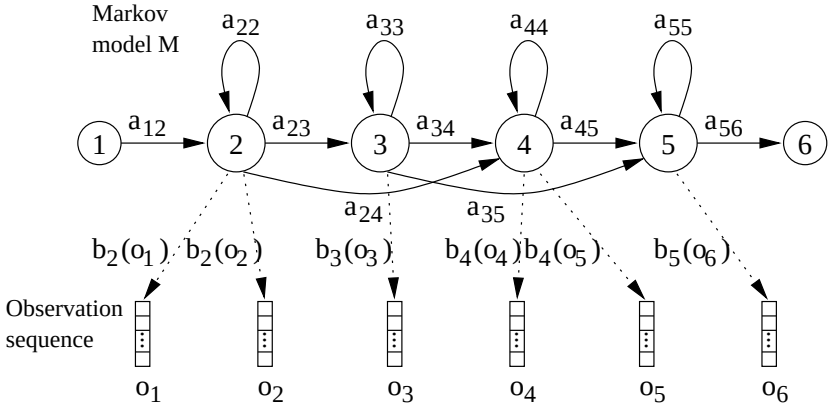
–

## 11.2 Skrytý Markovův model — HMM

DEF

v této sekci budeme uvažovat model pouze pro jednu třídu, ostatní budou podobné, byť s jinými parametry. Struktura modelu je znázorněna na Obr. 11.4. Stavy vyznačené většími kroužky jsou **vysílací (emitting)**, representují tedy vždy nějaký vstupní vektor. Termín "vysílací" je poněkud zavádějící, stavy ve skutečnosti žádné vektory nevysílají, ale produkují likelihoody, se kterými by dané vektory vyslaly. Můžeme to formulovat i tak, že tyto stavy "pojídají" vektory.

První a poslední stav jsou zvláštní – nevysílací. Používáme je jako konektory při napojování modelů, žádné vektory nebaští.




Obrázek 11.4: Skrytý Markovův model.

### 11.2.1 Přechodové pravděpodobnosti $a_{ij}$

kvantifikují, jak pravděpodobné je v modelu přeskočit ze stavu $i$ do stavu $j$ při posunu času z $t$ do $t+1$. Jedná se o skutečné *pravděpodobnosti*, proto respektují:

$$\sum_j a_{ij} = 1, \tag{11.3}$$

jinými slovy: součet všech přechodových pravděpodobností vycházejících z jednoho stavu musí být jednička. V příkladu modelu na Obr. 11.4 máme pouze tři typy:

- $a_{i,i}$ pravděpodobnost setrvání ve stavu.
- $a_{i,i+1}$ pravděpodobnost přechodu do následujícího stavu.
- $a_{i,i+2}$ pravděpodobnost přeskočení následujícího stavu (*nepoužívá se*, pouze někdy pro modely "krátkého ticha").

Přechodové pravděpodobnosti je možné zapsat následující maticí:

$$\mathbf{A} = \begin{bmatrix} 0 & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{11.4}$$

Vidíme, že matice má prvky pouze na diagonále a nad ní – musíme se v modelu zastavit nebo jít dál, nelze jít dozadu.

### 11.2.2 Likelihood, že je ve stavu $j$ vyslán vektor $\mathbf{o}(t)$

"Vysílací" likelihood $b_j[\mathbf{o}(t)] = p(\mathbf{o}(t)|j)$ může být:

- opravdová pravděpodobnost – u **diskrétních** HMM jsou vektory jsou pomocí "předkvantovány" pomocí vektorové kvantizace na symboly, stavy pak obsahují tabulky vysílacích pravděpodobností jednotlivých symbolů. Diskrétní HMM se ovšem příliš nepoužívají a nebudeme je probírat.
- hodnota funkce hustoty rozdělení pravděpodobnosti (pozor, není pravděpodobnost), dána většinou směsí Gaussovek:

$$b_j[\mathbf{o}(t)] = \sum_{i=1}^{M} \alpha_{ji} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_{ji}),$$

- Nebo jednou Gaussovkou, což bude náš školní případ:

$$b_j[\mathbf{o}(t)] = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$


Tyto modely nazýváme **CD-HMM – continuous density hidden Markov models.**

#### Poznámky k funkcím hustoty rozdělení pravděpodobnosti

Pokud nejsou parametry korelovány (nebo doufáme, že nejsou), jejich kovarianční matice je diagonální a namísto $P \times P$ kovariančních koeficientů stačí odhadnout pouze $P$ směrodatných odchylek (rozptylů). To vede k jednodušším modelům a k jejich odhadu bude tedy dostatek dat. Hodnota rozdělení je navíc dána pouze součinem 1-rozměrných Gaussových rozdělení (bez determinantu a inverze):

$$b_j[o(t)] = \prod_{i=1}^{P} \mathcal{N}(o(t); \mu_{ji}, \sigma_{ji}) = \prod_{i=1}^{P} \frac{1}{\sigma_{ji}\sqrt{2\pi}} e^{-\frac{[o(t)-\mu_{ji}]^2}{2\sigma_{ji}^2}} \tag{11.5}$$

Sady parametrů nebo celé *stavy* mohou být sdílené mezi modely, to vede k menšímu množství parametrů, spolehlivějšímu odhadu, a k potřebě méně paměti.

## 11.3 Likelihood generování celé sekvence O modelem $M$

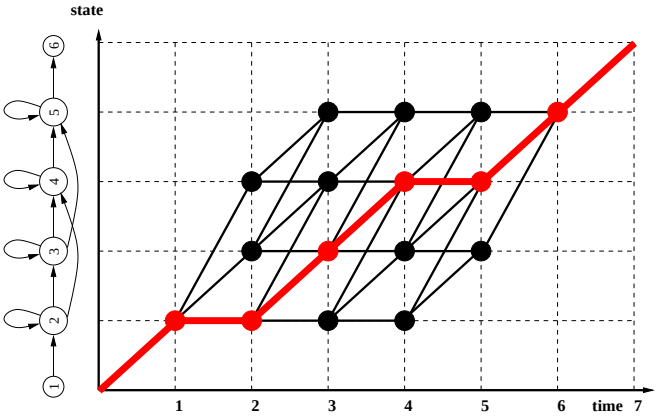
Budeme postupovat po krocích. Nejprve nadefinujme **stavovou sekvenci**, která stavy přiřadí vektorům (nebo vektory rozhází stavům, jak chcete), např: $X = [1\ 2\ 2\ 3\ 4\ 4\ 5\ 6]$. Ilustrace této i jiných stavových sekvencí je na Obr. 11.5 ve formě grafu.

Stavová sekvence má $T + 2$ prvků, protože první a poslední stav tam musí být **vždy** a nepatří k nim **žádný vektor**. Při práci s časem umisťujeme první stav do neexistujícího času 0 a poslední stav do neexistujícího času $T + 1$.

### 11.3.1 Likelihood generování O po cestě $X$:

je definována:

$$p(\mathbf{O}, X|M) = a_{x(o)x(1)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}_t) a_{x(t)x(t+1)},$$




Obrázek 11.5: Stavové sekvence definující přiřazení vektorů ke stavům. Stavová sekvence $X = [1\ 2\ 2\ 3\ 4\ 4\ 5\ 6]$ je vyznačena červeně.

Tento vztah můžeme shrnout do zjednodušující věty "pustíme vektory do modelu po stavové sekvenci $X$ a násobíme všechny pravděpodobnosti a likelihoody, které potkáme po cestě".

Jak však definovat *jednu* likelihood generování sekvence vektorů modelem ?

a) Baum-Welch:

$$p(\mathbf{O}|M) = \sum_{\{X\}} p(\mathbf{O}, X|M),$$

bereme sumu přes *všechny stavové sekvence* délky $T + 2$.

b) Viterbi:

$$p^\star(\mathbf{O}|M) = \max_{\{X\}} p(\mathbf{O}, X|M),$$

je likelihood optimální cesty.

## 11.4 Trénování HMM a co je ve skrytých Markovových modelech skrytého

Trénování modelů je schematicky znázorněno na Obr. 11.6.

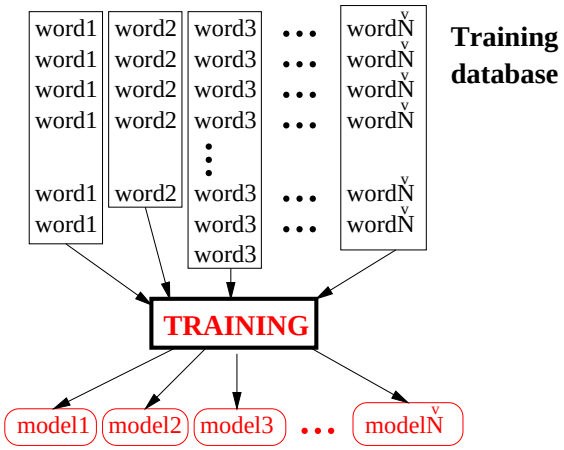
Trénování budeme opět ukazovat pouze na 1 modelu, soubor parametrů $\boldsymbol{\Theta}$ obsahuje: přechodové pravděpodobnosti $\alpha_{ij}$, vektorové střední hodnoty $\boldsymbol{\mu}_j$ a kovarianční matice $\boldsymbol{\Sigma}_j$ nebo vektory směrodatných odchylek $\boldsymbol{\sigma}_j$.

Podobně jako u Gaussian Mixtures, analyticky nejdou parametry odhadnout, proto se používá Expectation Maximization s kritériem:

$$\sum_t p(\mathbf{o}(t)|\boldsymbol{\Theta}^{i-1}) \times \log p(\mathbf{o}(t)|\boldsymbol{\Theta})$$

kde stavová sekvence $X$ je skrytá (hidden) informace. Po hutné matematice, která je mimo rozsahu tohoto kursu, vede odvození k následujícímu intuitivnímu postupu:

1. hrubý odhad parametrů.
2. odhad pravděpodobností $L_j(t)$, že se v čase $t$ nacházíme ve stavu $j$ – "occupation counts".




Obrázek 11.6: Ilustrace trénování modelů.

3. odhad nových parametrů, pravděpodobnosti $L_j(t)$ "měkce rozhazují vektory mezi stavy".
4. opakuj body 2 a 3.

Další podsekce presentují tyto kroky detailněji.

### 11.4.1 Inicialisace

Vše budeme ukazovat pouze na 1 trénovací sekvenci $\mathbf{O}$, generalisace na více sekvencí je jednoduchá. Předpokládáme, že HMM má $N$ stavů, z toho první a poslední jsou nevysílací.

Parametry modelu jsou **zhruba odhadnuty**:

- nastavíme $a_{ji}$ tak, aby definovaly požadovanou konfiguraci HMM, vložíme do nich nějaké konstanty.
- rozdělíme $\mathbf{O}$ na $N-2$ úseků o stejné délce, pak pro $j = 2 \ldots N-2$:

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{T_j} \sum_{t=t_{beg_j}}^{t_{end_j}} \mathbf{o}(t) \qquad \hat{\boldsymbol{\Sigma}}_j = \frac{1}{T_j} \sum_{t=t_{beg_j}}^{t_{end_j}} (\mathbf{o}(t) - \boldsymbol{\mu}_j)(\mathbf{o}(t) - \boldsymbol{\mu}_j)^T$$

### 11.4.2 Odhad state occupation functions

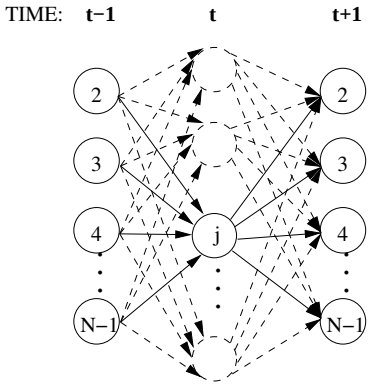
Likelihood "bytí ve stavu $j$ v čase $t$" $L_j(t)$ můžeme (viz. Obr. 11.7) spočítat jako sumu všech cest, které v čase $t$ projdou stavem $j$: $p(\mathbf{O}, x(t) = j|M)$

Potřebujeme ovšem zajistit, aby $L_j(t)$ byly opravdové pravděpodobnosti, tedy:

$$\sum_{j=1}^{N} L_j(t) = 1, \tag{11.6}$$

jinými slovy: příspěvek jednoho vektoru všem stavům musí být přesně 100%. Musíme normalizovat sumou likelihoodů všech cest, které projdou čímkoliv v čase $t$:

$$L_j(t) = \frac{p(\mathbf{O}, x(t) = j|M)}{\sum_j p(\mathbf{O}, x(t) = j|M)}. \tag{11.7}$$




Obrázek 11.7: Ilustrace likelihoodu "bytí ve stavu $j$ v čase $t$.

...jenže to už jsme opravdu normalizovali sumou **všech cest** modelem, takže můžeme rovnou normalizovat BAUM-WELCHOVÝM likelihoodem:

$$L_j(t) = \frac{p(\mathbf{O}, x(t) = j|M)}{p(\mathbf{O}|M)}. \tag{11.8}$$

### 11.4.3 Výpočet $P(\mathbf{O}, x(t) = j|M)$ pomocí dopředných a zpětných částečných likelihoodů

**Částečný dopředný likelihood** je definován jako suma likelihoodů všech částečných sekvencí, které začínají na začátku modelu a v čase $t$ se nacházejí ve stavu $j$:

$$\alpha_j(t) = p\left(\mathbf{o}(1)\dots\mathbf{o}(t), x(t) = j|M\right)$$

Částečné dopředné likelihood se počítají pomocí iterativního algoritmu:

- Inicialisace:
$$\alpha_j(1) = a_{1j}b_j[\mathbf{o}(1)] \text{ pro } 2 \le j \le N-1 \tag{11.9}$$
- Jdeme dále v čase, výpočet pro normální $\alpha_j(t)$ je znázorněn na Obr. 11.8
$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j[\mathbf{o}(t)] \text{ pro } 2 \le j \le N-1 \tag{11.10}$$
- Uzavření algoritmu:
$$\alpha_N(T+1) = \sum_{i=2}^{N-1} \alpha_i(T)a_{iN} \tag{11.11}$$

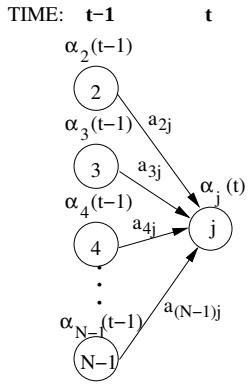
Je fajn, že jsme poslední $\alpha$-ou vypočetli Baum-Welchovu likelihood:

$$p(\mathbf{O}|M) = \alpha_N(T+1) \tag{11.12}$$

**Částečný zpětný likelihood** je definován jako suma likelihoodů všech sekvencí začínajících v posledním čase v posledním stavu a v čase $t$ se nacházejících ve stavu $j$):

$$\beta_j(t) = p\left(\mathbf{o}(t+1)\dots\mathbf{o}(T)|x(t) = j, M\right) \tag{11.13}$$

I tento likelihood se počítá iterativně:




Obrázek 11.8: Výpočet částečné dopředné pravděpodobnosti $\alpha_j(t)$.

- Inicialisace (odzadu!):

$$\beta_j(T) = a_{jN} \text{ pro } 2 \leq j \leq N-1 \tag{11.14}$$

- Jdeme pozadu v čase, výpočet pro normální $\beta_j(t)$ je ilustrován na Obr. 11.9.

$$\beta_j(t) = \sum_{i=2}^{N-1} a_{ji} b_i[\mathbf{o}(t+1)]\beta_i(t+1) \text{ pro } 2 \leq j \leq N-1 \tag{11.15}$$

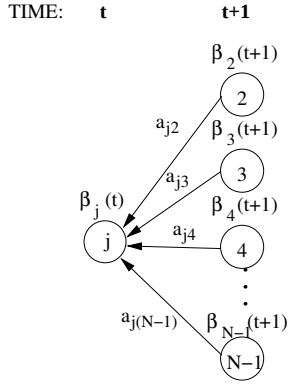
- Uzavření algoritmu:

$$\beta_1(0) = \sum_{i=2}^{N-1} a_{1i} b_i[\mathbf{o}(1)]\beta_i(1) \tag{11.16}$$

A zase jsme získali Baum-Welchovu likelihood:

$$P(\mathbf{O}|M) = \beta_1(0), \tag{11.17}$$

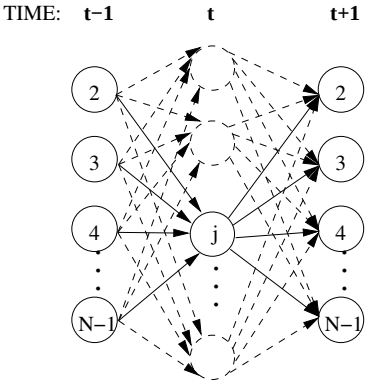
dokonce si porovnáním $\alpha_N(T+1)$ a $\beta_1(0)$ můžeme správnost výpočtu zkontrolovat.




Obrázek 11.9: Výpočet částečné zpětné pravděpodobnosti $\beta_j(t)$.

Máme-li k disposici hodnoty $\alpha$ a $\beta$ pro všechny stavy a všechny časy, budeme konečně schopni spočítat $L_j(t)$ (viz. Obr. 11.10):

$$L_j(t) = \frac{\alpha_j(t)\beta_j(t)}{p(\mathbf{O}|M)} \tag{11.18}$$




Obrázek 11.10: Výpočet $L_j(t)$.

### 11.4.4 Update parametrů

Při odhadu $\boldsymbol{\mu}_j$ a $\boldsymbol{\Sigma}_j$ slouží occupation counts $L_j(t)$ jako "měkký rozhazovač vektorů na stavy" (viz Obr. 11.11).

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{t=1}^{T} L_j(t)\mathbf{o}(t)}{\sum_{t=1}^{T} L_j(t)} \qquad \hat{\boldsymbol{\Sigma}}_j = \frac{\sum_{t=1}^{T} L_j(t)(\mathbf{o}(t)-\boldsymbol{\mu}_j)(\mathbf{o}(t)-\boldsymbol{\mu}_j)^T}{\sum_{t=1}^{T} L_j(t)}. \tag{11.19}$$

Odhad přechodových pravděpodobností je zapsán o něco složitěji:

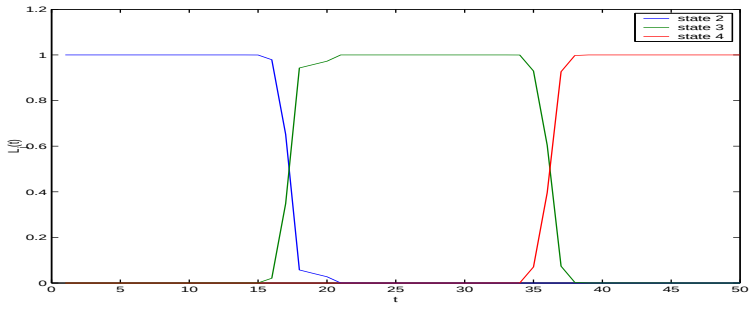
$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \alpha_i(t)a_{ij}b_j(\mathbf{o}(t+1))\beta_j(t+1)}{\sum_{t=1}^{T} L_j(t)}$$

Ve jmenovateli všech vztahů si všimněte sumy $\sum_{t=1}^{T} L_j(t)$. Vzhledem k tomu, že všechny rovnice jsou prakticky výpočtem vážených průměrů, je přirozené, že ve jmenovateli nacházíme součet vah.

Po updatu parametrů se **a můžeme se směle vrátit na odhad $L_j(t)$ a iterovat**

### 11.4.5 Technická realizace algoritmu trénování modelů

1. Alokuj akumulátor pro každý odhadovaný vektor/matici.
2. Spočítej dopředné a zpětné pravděpodobnosti $\alpha_j(t)$ a $\beta_j(t)$ pro všechny časy $t$ a všechny stavy $j$. Spočítej state occupation functions $L_j(t)$.

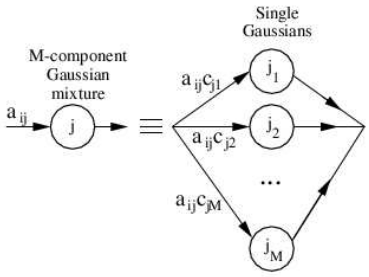



Obrázek 11.11: Ilustrace $L_j(t)$ pro HMM se třemi vysílacími stavy a skutečný řečový signál. Modrá $L_2(t)$ váží vektory pro druhý stav, zelená $L_3(t)$ váží vektory pro třetí stav a červená $L_2(t)$ váží vektory pro čtvrtý stav,

3. Ke každému akumulátoru přidej příspěvek od vektoru $\mathbf{o}(t)$ vážený příslušnou $L_j(t)$.
4. Použij konečnou hodnotu akumulátoru pro odhad vektoru/matice pomocí rovnice 11.19 — nesmíme zapomenout na normování výrazem $\sum_{t=1}^{T} L_j(t)$.
5. Pokud se hodnota $p(\mathbf{O}|M)$ od minulé iterace podstatně nezměnila, stop, jinak Goto 1.

### 11.4.6 Trénování modelů obsahujících směsi Gaussovek

Každý stav se směsí Gaussovek se dá (viz. Obr. 11.12) rozkreslit do několika stavů po jedné Gaussovce. Přechodové pravděpodobnosti jsou určeny váhami jednotlivých Gaussovek (mixture weights). Trénovací Baum-Welchův algoritmus probíhá stejně jako v předcházejícím případě.




Obrázek 11.12: Přechod od Gaussian mixture modelu k několika stavům.

### 11.4.7 Trénování na více promluvách

Trénování na jedné promluvě byl pouze školní příklad, v praxi trénujeme až na desetitisících promluv. Použijeme prakticky stejný algoritmus jako výše, ale:

- Dopředné a zpětné likelihoody se odhadnou na každé promluvě.
- Occupation counts se také odhadnou na každé promluvě.
- Provede se update akumulátorů také na každé promluvě.

- pak se to jen podělí a získají se nové hodnoty parametrů.
- více viz kompletní rovnice v HTK Book[2] sekce "Parameter re-estimation formulae".

## 11.5 Rozpoznávání s HMM

V minulých sekcích jsme viděli, jak natrénovat HMM. Zatím ale nevíme, jak s nimi bude možné rozpoznávat. Základní úlohu si opět ukážeme na rozpoznávání izolovaných slov, pak ji rozšíříme na rozpoznávání spojených slov a plynulé řeči.

Rozpoznávání musí pracovat podle schematu na Obr. 9.5:

- máme rozpoznat neznámou sekvenci $\mathbf{O}$.
- ve slovníku máme $\check{N}$ slov $w_1 \dots w_{\check{N}}$.
- pro každé máme natrénovaný model $M_1 \dots M_{\check{N}}$.
- Otázka zní: "Který model by generoval $\mathbf{O}$ s největším likelihoodem ?"

Zjednodušováním Bayesova vztahu (11.1) jsme si ukázali, že budeme hledat takový model, který pro sekvenci vektorů $\mathbf{O}$ dá maximální likelihood:

$$i^\star = \arg\max_i \{p(\mathbf{O}|M_i)\} \tag{11.20}$$

pro hledání jediného likelihoodu "vyslání" sekvence $\mathbf{O}$ modelem $M_i$ použijeme Viterbiho likelihood pro nejpravděpodobnější stavovou sekvenci:

$$p^\star(\mathbf{O}|M) = \max_{\{X\}} p(\mathbf{O}, X|M). \tag{11.21}$$

takže:

$$i^\star = \arg\max_i \{p^\star(\mathbf{O}|M_i)\}. \tag{11.22}$$

Teoreticky bychom měli vyhodnotit likelihoody **všech** možných stavových sekvencí $X$:

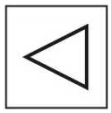
$$p(\mathbf{O}, X|M) = a_{x(o)x(1)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}_t) a_{x(t)x(t+1)},$$

a najít Viterbiho likelihood jako likelihood optimální cesty:

$$p^\star(\mathbf{O}|M) = \max_{\{X\}} p(\mathbf{O}, X|M),$$

To by ovšem bylo velmi náročné a pro složitější modely a dlouhé promluvy by rozpoznávání patrně nikdy neskončilo...

### 11.5.1 Viterbiho trik



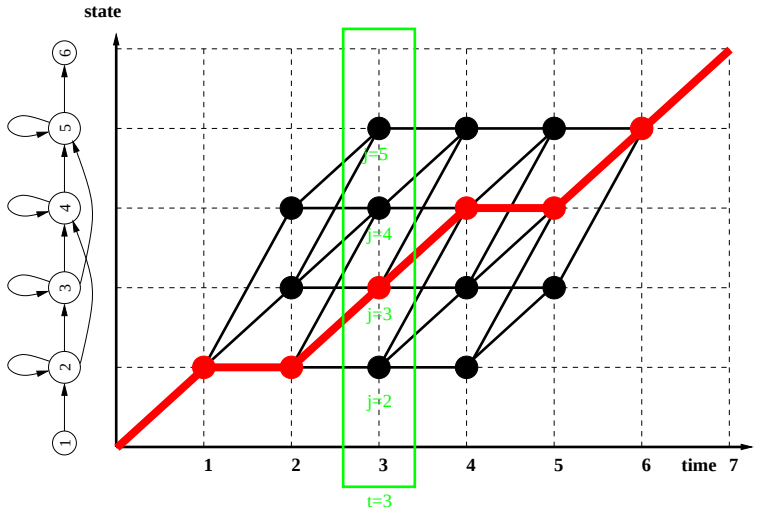
Naštěstí nemusíme nejprve počítat likelihood všech cest a pak teprve rozhodovat, která je nejlepší, ale můžeme evaluovat **všechny cesty současně** a rozhodnutí o nejlepší cestě dělat ne až na konci, ale pro každý čas $t$ a každý stav $j$ – viz. Obr. 11.13.


Definujeme **částečnou Viterbiho likelihood** jako likelihood nejlepší cesty končící ve stavu $j$ v čase $t$:

$$\Phi_j(t) = p^\star\left(\mathbf{o}(1) \dots \mathbf{o}(t), x(t) = j|M\right).$$

a počítáme ji opět iterativně:

[2] `http://htk.eng.cam.ac.uk`, vyžaduje registraci zdarma.




Obrázek 11.13: Ilustrace Viterbiho algoritmu: rozhodnutí o nejlepší cestě děláme v každém čase $t$ a v každém stavu $j$. Po zpracování času $t$ zelené okno posuneme do následujícího času $t+1$.

- inicialisace:
$$\Phi_j(1) = a_{1j} b_j[\mathbf{o}(1)] \quad \text{pro} \quad 2 \leq j \leq N-1.$$
- cyklus pro všechny časy $t$ a všechny stavu $j$ (ilustrace viz Obr. 11.14):
$$\Phi_j(t) = \max_i \{\Phi_i(t-1) a_{ij}\}\, b_j[\mathbf{o}(t)] \quad \text{pro} \quad 2 \leq j \leq N-1$$
- Uzavření algoritmu
$$\Phi_N(T+1) = \max_i \{\Phi_i(T) a_{iN}\} \tag{11.23}$$

Poslední $\Phi$ je požadovaná Viterbiho likelihood:

$$p^\star(\mathbf{O}|M) = \Phi_N(T+1)$$

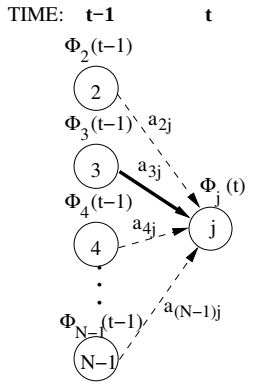
Výpočet obvykle probíhá v logaritmické oblasti: máme menší problémy s dynamikou, a všechny součiny přejdou na součty:

$$\Psi_j(t) = \max_i \{\Psi_i(t-1) + \log a_{ij}\} \log b_j[\mathbf{o}(t)]$$

Zkuste si zodpovědět otázku, zda je možné v logaritmické oblasti implementovat i Baum-Welchův algoritmus pro výpočet state occupation counts $L_j(t)$. Pokud ano, co bude hlavním problémem ?

## 11.5.2 Implementace Viterbiho pomocí token-passing

Viterbiho algoritmus se velmi dobře implementuje pomocí algoritmu **token-passing**, kde definujeme struktury, které přecházejí mezi stavy a při přechodu akumulují logaritmické pravděpodobnosti a likelihoody. Jelikož je těžké představit




Obrázek 11.14: Ilustrace jednoho kroku Viterbiho algoritmu: výpočet částečného Viterbiho likelihoodu $\Phi_j(t)$ pro stav $j$ a čas $t$. Tlustá šipka značí maximální hodnotu.

si žetony (tokens) akumulující cokoliv, ve výklad použije piva (Obr. 11.15), která jsou českému a slovenskému čtenáři mnohem bližší.

Každý stav modelu $j$ může držet půllitr s pivem. Pivo bude obsahovat log likelihood $\Psi_j(t)$. Pracujeme s log likelihoody, takže budeme pracovat se součty, budeme prostě dolévat log-likelihoody.

Algoritmus je zapsán jako:

**Inicialisace:** vlož prázdný půllitr do každého vstupního stavu modelu (běžně pouze stav 1.).

**Iterace:** pro časy $t = 1 \ldots T$:

- v každém stavu $i$, který obsahuje půllitr, tento naklonuj a pošli kopii do všech napojených stavů $j$. Po cestě dolij $\log a_{ij} + \log b_j[\mathbf{o}(t)]$.
- pokud se v nějakém stavu nachází více než jeden půllitr, nechej jen ten nejplnější, zahoď ostatní.

**Konec:** ze všech stavů $i$ spojených s výstupním stavem $N$, které obsahují půllitr, pošli jej do $N$ a dolij $\log a_{iN}$. V posledním stavu $N$, vyber jen nejplnější půllitr a zahoď ostatní. Hladinka piva ve stavu $N$ odpovídá log Viterbiho likelihoodu: $\log p^\star(\mathbf{O}|M)$.

## Příklad výpočtu Viterbiho likelihoodu pivo-passingem

x+y

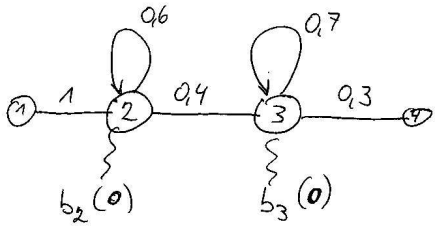
Na Obr. 11.16 máme model se dvěma vysílacími stavy.

Bude vhodné si spočítat hodnoty logaritmů přechodových pravděpodobností: $\log 1 = 0$, $\log 0.6 = -0.51$ $\log 0.4 = -0.92$, $\log 0.7 = -0.36$, $\log 0.3 = -1.20$

Máme spočítat Viterbiho likelihood pro sekvenci pěti vektorů, pro které máme před-počítané následující vysílací likelihoody. I ty bude vhodné převést na log:

| | $\mathbf{o}(1)$ | $\mathbf{o}(2)$ | $\mathbf{o}(3)$ | $\mathbf{o}(4)$ | $\mathbf{o}(5)$ |
|---|---|---|---|---|---|
| $b_2$ | 0.0340 | 0.0349 | 0.0398 | 0.0013 | 0.0001 |
| $b_3$ | 0.0098 | 0.0010 | 0.0033 | 0.0340 | 0.0129 |
| $\log b_2$ | -3.38 | -3.35 | -3.22 | -6.65 | -9.21 |
| $\log b_3$ | -4.63 | -6.91 | -5.71 | -3.38 | -4.35 |

Obrázek 11.15: Pivo v token-passingu. Hladinka piva odpovídá naakumulovanému log-likelihoodu.




Obrázek 11.16: Model pro příklad výpočtu Viterbiho likelihoodu pivo-passingem.

Průběh algoritmu a jeho výsledek jsou na Obr. 11.17.

### 11.5.3 Pivo passing pro rozpoznávání spojených slov

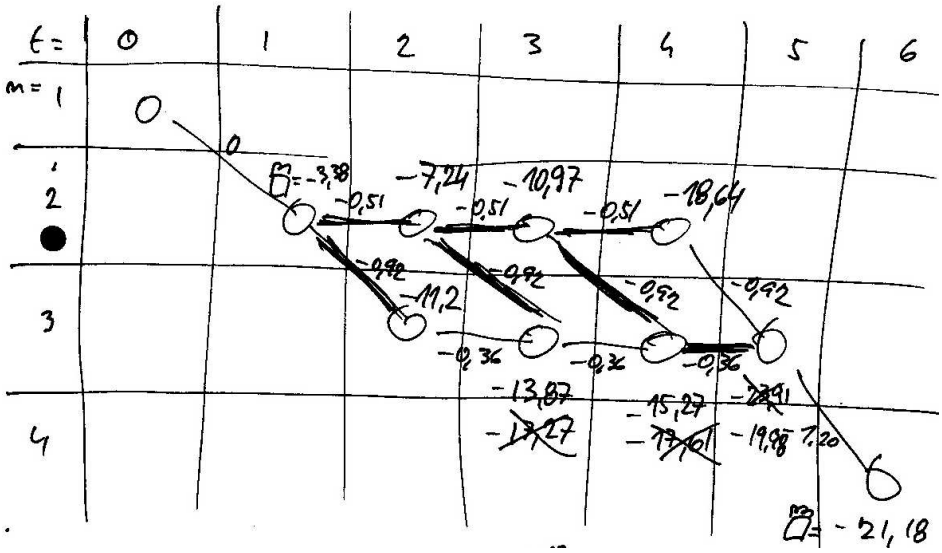
postavíme mega-model ze všech slov, "slepíme" první a poslední stavy - viz. Obr. 11.18. K lepení použijeme první a poslední nevysílací stavy.

Rozpoznávání pak probíhá podobně jako pro izolovaná slova, musíme si však pamatovat slova, kterými optimální půllitr prošel – na každý půllitr nalepíme lísteček, na který budeme zapisovat identity slov, kterými půllitr prošel (Obr. 11.19). Na konci rozpoznávání pak přečteme lísteček tokenu s nejlepším log-likelihoodem. Technicky jsou tyto "lístečky" realizovány tak, že na token se pomocí ukazatele zachytí struktura WLR (word-link-record), která obsahuje identitu právě opuštěného slova a čas. Pomocí WLR je pak u nejlepšího modelu možné trasovat, kudy prošel.

Předpokládejme, že modely slov se skládají z modelů fonémů. Jak byste zajistili, aby byla na konci rozpoznávání k disposici nejen informace o prošlých slovech a jejich časování, ale i o prošlých fonémech a jejich časování ?

## 11.6 Rozpoznávání řeči s velkým slovníkem

Při rozpoznávání s velkým slovníkem (typická velikost je 50000 slov pro angličtinu, ale několik stovek tisíc slov pro češtinu) není možné natrénovat modelu každého




Obrázek 11.17: Příklad pivo-passingu a jeho výsledek.

slov a je nutné použít menší jednotky. Pracujeme s **fonémy** nebo **kontextově závislými fonémy**.

Rozpoznávaná slova rozdělíme na fonémy, pro které máme natrénované modely (typicky má model fonému 3 vysílací stavy). Např:
"six" = s i k s.

### 11.6.1 Trénování modelů při rozpoznávání spojité řeči

Při rozpoznávání řeči s velkým slovníkem obvykle používáme modely fonémů nebo kontextově závislých fonémů. Při trénování máme ale obvykle k disposici jen jen ortografickou (slovní) transkripci, jako např.:

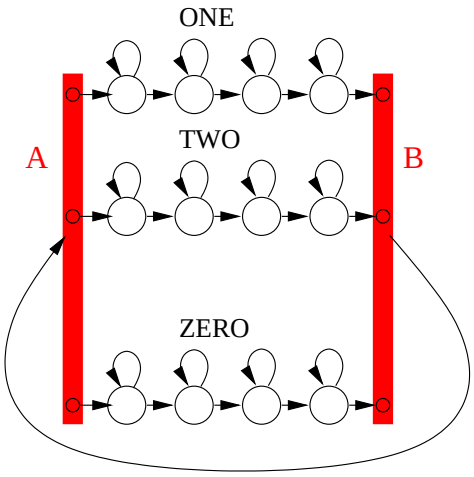
```
0 59392 The total of these three volumes is the final
        combustion chamber volume.
```

Řešení je v postavení trénovací HMM-sítě pro každou trénovací promluvu (na Obr. 11.20 je hypotetický případ věty, která by mohla obsahovat slova "dog", "cat" a "and", většinou je přepis přesnější...).

Při trénování má každý model své akumulátory a postup pro **každou trénovací promluvu** je následující:

- Převedeme grafémy (grafická podoba slov – písmena) na fonémy (grapheme to phoneme, g2p konverze).
- Postavíme "mega-model" spojený z modelů fonémů, pokud je více výslovnostních variant, dáme je do modelu paralelně.
- Vypočteme pro každý zúčastněný foném jeho $\alpha$, $\beta$ i $L_j(t)$.
- updatujeme akumulátory.

Po projití všech trénovacích promluv podtrhneme a sečteme, updatujeme parametry modelů. Úplné rovnice uvádí opět HTK Book, sekce "Embedded Model Reestimation (HEREST)".




Obrázek 11.18: Rozpoznávání spojených slov megamodelem "slepeným" z modelů slov.

### 11.6.2 Rozpoznávání pomocí fonémů

Použijeme algoritmus pivo-passing jako v předcházejícím případě, modely slov jsou poskládány z modelů fonémů (viz. Obr. 11.20). Všimněme si ale, že fonémy jsou silně závislé na svém, kontextu. Např. 'n' v "nothing" se podstatně liší od 'n' v "bank".

### 11.6.3 Kontextově závislé fonémy

context-dependent (CD) phoneme models vzniknou přidáním závislosti na kontextu. Klasicky bereme 1 foném vlevo, 1 foném vpravo, hovoříme o **trifonech (triphones)**. Slovo z předcházejícího příkladu by pak bylo:
"six" = s+i s-i+k i-k+s k-s.
Nevýhodou je mnoho různých trifonů (teoreticky $N^3$ pro počet fonémů $N$), nespolehlivý odhad na trénovacích datech a chybějící trifony.

DEF

### 11.6.4 Trifony se sdílenými stavy (tied-state triphones)

DEF

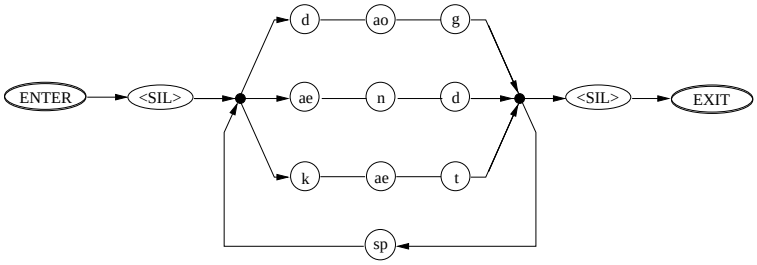
Některé stavy kontextově závislých trifonů budeme muset svázat ("tie"), abychom se vyhnuli problému nedostatečného množství dat a neexistujících trifónů. Skupiny pro vázání jsou většinou definovány pomocí foneticky motivovaných otázek, chceme-li například najít (tied-state triphones): vázání stavů pro podobné modely, menší množství dat. Vázání pomocí fonetických otázek. Chceme-li například najít skupiny pro vázání prostředních stavů ve třech trifónech:

p-a+f
u-a+n
g-a+r

může strom pro pokládání otázek vypadat podle Obr. 11.21. Pro rozhodnutí, zda pokračovat v dalším dělení stromu, musí být splněny dvě podmínky:

1. musí být k disposici dostatek trénovacích dat pro natrénování obou větví.
2. rozdělení na 2 větve musí mít positivní efekt, likelihood dat která vznikne natrénováním modelů podle nového dělení musí být větší než likelihood s


Obrázek 11.19: Obyčejné pivo a pivo se zaznamenanými slovy, kterými prošlo.




Obrázek 11.20: Síť pro trénování modelů.

původním dělením.

Výsledný strom dokáže produkovat modely i pro v trénování neviděné trifony – stačí postupovat podle fonetických otázek, na listech stromu najdeme pak výsledné funkce hustoty rozdělení pravděpodobnosti. Těchto akustických rozdělení bývá většinou v systémech pro rozpoznávání několik tisíc.

### 11.6.5 Jazykové modelování

DEF

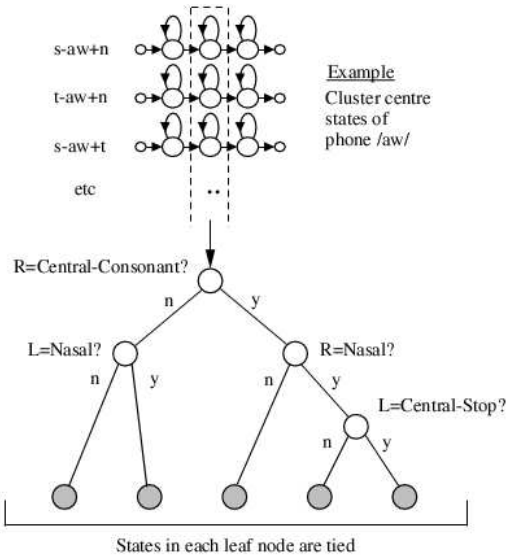
Při rozpoznávání s velkým slovníkem (několik desítek tisíc slov) není možné brát v úvahu pouze akustické modelování. **Jazykové modely – language models** udávají apriorní pravděpodobnost sekvencí slov[3]. Navrátíme-li se do teorie k základnímu vzorci pro rozpoznávání:

$$W_1^{N\star} = \arg\max_{\forall W_1^N} \left\{ P(W_1^N|\mathbf{O}) \right\},$$

který vyhodnocujeme pomocí Bayesova vzorce:

$$P(W_1^N|\mathbf{O}) = \frac{p(\mathbf{O}|W_1^N)P(W_1^N)}{p(\mathbf{O})}$$

[3]apriorní znamená, že k určení této pravděpodobnosti nemusíme vidět žádná vstupní akustická data.




Obrázek 11.21: Rozhodovací strom pro sdílení středního stavu trifónových modelů *-a+*.

stará se jazykové modelování o určení pravděpodobnosti $P(W_1^N)$:

- v ideálním případě bychom měli násobit podmíněné pravděpodobnosti:

$$P(W_1^N) = \prod_{i=1}^{N} P(W_1)\ P(W_2|W_1)\ P(W_3|W_1W_2)\ \ldots\ P(W_N|W_1 \ldots W_{N-1})$$

  ale pravděpodobnosti dlouhých sekvencí slov nejdou odhadnout...
- zkrátíme tedy historii na 1 (se současným slovem dvojice slov – bigramy) nebo 2 (trigramy).

**Odhad pravděpodobností LM**

na velkém korpusu textu odhadujeme pravděpodobnost pomocí

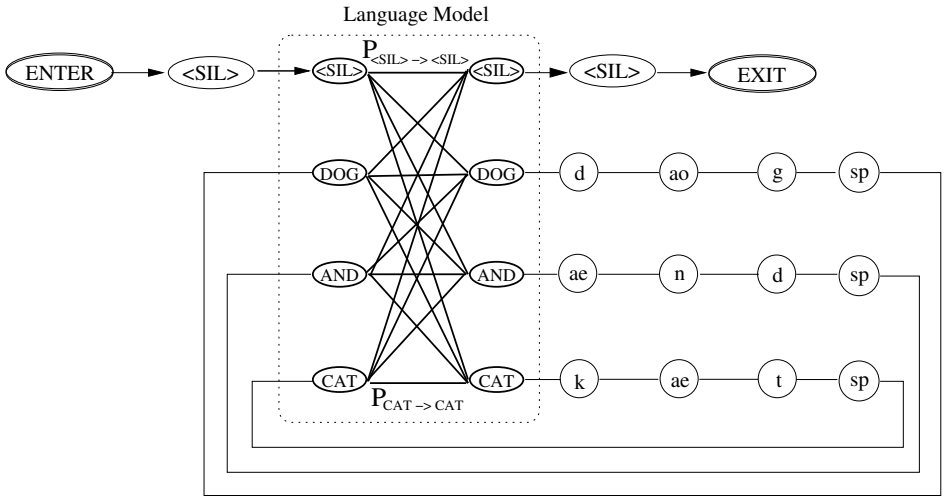
$$P(W|H) = N(W,H)/N(H)$$

kde $H$ je historie (1 slovo pro bigram a 2 slova pro trigram) a $W$ je současné slovo. Ani trigramy není možné spolehlivě odhadnout a chceme-li se vyvarovat nul v LM, musíme pracovat s **Back-off jazykovými modely**:

DEF

$$P(W|H) = \begin{cases} (N(W,H) - D)/N(H) & \text{for} \quad N(W,H) > p \\ b(H)P(W|H_{-1}) & \text{for} \quad N(W,H) \leq p \end{cases}$$

O jazykových modelech se detailně dozvíme v pokračovacím kursu SRE. Schéma sítě, která řídí rozpoznávač tří slov s jednoduchým bigramovým modelem je na Obr. 11.22.




Obrázek 11.22: Rozpoznávací síť pro 3 slova a bigramový jazykový model.