MATEMATICKO-FYZIKÁLNA FAKULTA
UNIVERZITY KOMENSKÉHO
ÚSTAV INFORMATIKY

---

Diplomová práca

# Experimentálny systém rozpoznávania reči

Marek Nagy

BRATISLAVA 1999

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne a použil som len literatúru uvedenú v zozname.

V Bratislave, 30. apríla 1999.

Hlavne by som chcel poďakovať môjmu školiteľovi Jánovi Šefránkovi z ÚI MFF za pomoc pri písaní a pri experimentoch. Rovnako i Ľube Konrádovej z katedry psychológie FiF za pomoc pri hľadaní vhodnej literatúry k psychologickým aspektom rozpoznávania reči. A tiež Milanovi Ruskovi z ÚTRR SAV a Kataríne Hennelovej z katedry humanistiky MFF za informatívne sprístupnenie súkromných knižníc.

V neposlednej miere by som chcel poďakovať i všetkým obetavým pokusným osobám, menovite Nikolke za jej trpezlivosť.

Ďalej ďakujem svojím rodičom za obetavosť a námahu, vďaka ktorej som mohol v týchto pohnutých časoch študovať.

# Obsah



...

# 1. Úvod

Existuje už mnoho prác v oblasti rozpoznávania reči, avšak väčšinou pre anglo-saské jazyky. Z praktického hľadiska ich je dostupných iba zopár (top secret). Tieto sa naviac ťažko prispôsobujú na slovenské pomery (hlavne štatisticky založené prístupy). Najviac z nich používa rozpoznávanie na základe obmedzeného slovníka. Táto metóda sa vo väčšine aplikácií javí dostačujúca. Nerieši však problém plynulého rozhovoru s počítačom. Riešenie tohto problému by malo široké využitie, či už pre sluchovo postihnutých, kompresiu telefonických hovorov, ... Rovnako by slúžilo ako medzistupeň pri komunikácii (aj s porozumením) s počítačom hovoreným slovom (odveký sen umelej inteligencie).

Hlavnou motiváciou pre zahájenie bádateľských výskumov bola knižka [3]. Po jej prvotnom prelistovaní a neskoršom prečítaní som sníval iba o rozpoznávači a syntetizátore (počítač povie, čo má napísané). Po zbežnom preniknutí do problematiky som nadobudol pocit, že problém syntetizátorov je obstojne riešený. Otvorenou oblasťou je zatiaľ rozpoznávač. Ako by to malo fungovať? Je to iba mechanický kognitívny proces? A mnoho ďalších otázok sa mi valili hlavou. Jednou takou najzákladnejšou je zrejme otázka: Možno oddeliť rozpoznávanie reči od samotného významu? Odpoveďou je možno márna snaha rozpoznávačov založených iba na mechanických prístupoch.

Ďalším problémom bola voľba vhodnej výšky nasadenej latky. Rozpracovať iba jednu techniku alebo sa pokúsiť o hlavný cieľ. Zvíťazila moja dobrodružná povaha a hlavný cieľ: **rozpoznávanie reči**.

Nasleduje moja stručná špecifikácia nastoleného problému:

- Užívateľ hovorí do mikrofónu a text sa zobrazuje.
- Príchod nového užívateľa spôsobí automatickú adaptáciu systému.
- Predpoklad: užívatelia nehovoria naraz.
- Užívateľ hovorí plynule bez obmedzení.
- Užívateľ nemusí explicitne upozorniť na začiatok prejavu.
- Systém neustále konvertuje. (Odmlku chápe ako medzeru.)
- Systém pracuje v reálnom čase.

Keďže za mnou nestál žiadny priemyselný gigant, moje hardwarové podmienky boli skromné. Hlavne podmienka reálneho času sa realizovala ťažko. Patrila však medzi jedny z najlákavejších.

Ešte uvediem stručný prehľad používaných techník v rozpoznávaní reči:

- Wave - samotná vlna.
- Furierova transformácia (FT) - konštrukcia frekvenčného spektra (príliš strapaté).
- Lineárna predikcia (LP) - pomocou nej sa dá skonštruovať obálka frekvenčného spektra. Rýchlejší výpočet kepstrálnych koeficientov. Vhodné príznaky pre klasifikáciu.
- Kepstrum (cez FT alebo LP) - vhodné na tvorbu frekvenčnej obálky. Zisťovanie prítomnosti hlasivkového tónu.
- Dynamické programovanie - väčšinou pri porovnávaní vĺn.
- Skryté markove modely (HMM) - väčšinou modelovanie štatistických situácii.
- Pravdepodobnostné gramatiky - syntaktické konštrukcie jazyka i s pravdepodobnosťami výskytov.
- Neurónové siete (NS) - odstraňovanie šumu, rozpoznávanie, syntéza slov.

# 2. Analýza a návrh systému

"Mami, kedy príde nová tacocha," pýta sa malá Zuzanka. Pritom ani nevie, že použila nezmyselné slovo. Z jej pohľadu všetko začínajúce na "ma" má svoj ekvivalent s "ta". Súvis so slovami mama a tata. Za povšimnutie tiež stojí dvojica bielok–žielok. Dieťa neuvedomujúc si čo robí, prísne syntakticky aplikuje pravidlá pre komunikáciu rečou. Vývoj reči od detského veku približuje knižka [2]. Priblížme si z nej aspoň spektrum slovných druhov v jednotlivých časových obdobiach vývinu dieťaťa. Skôr ide o približné údaje zisťované podľa dievčatka. Vo veku 1rok a 3mesiace jej slovná zásoba obsahovala 100% podstatných mien. Neskôr vo veku 1 rok 8 mesiacov to už bolo 78% podstaných mien a 22% slovies. Vo veku 2 rokov sa zastúpenie zmenilo na 63% a 23%, zvyšok boli už ostatné slovné druhy. Súvis možno hľadať s fyzickými zmenami dieťaťa. Kým bolo málo pohyblivé ozrejmovalo si (ocumlávalo, chytalo, hrýzlo) iba pomenovania predmetov–hmotných objektov. Neskôr s rozvojom chôdze nadobudlo pocity dynamiky a pohybu (utekám tam, kde je mamička; som rýchlejšie ako gúľajúca sa lopta). Za povšimnutie stojí odmietanie a nechápanie abstraktných pojmov. "Ale veď topánky nerastú na strome," opravuje malý Miško otecka, ktorý mu číta rozprávku. Hoci dieťa vie (dôkazom je oprava otecka), že to nie je pravda, rado si vypočuje takúto rozprávku, hlavne, keď má priestor blysnúť sa svojou múdrosťou.

V knižkách [5] a [6] sú rozpracované rozličné metódy učenia sa. Ako podporný prostriedok je rýchločítanie. Idea spočíva v tom, aby sa text nečítal v zvukovej podobe, ale iba jednoducho prezrel. V mysli sa nezoberáme samotným slovom, ale významom, ktorý nám naskočí pri pohľade naň. Takýto spôsob čítania však treba nacvičiť. Keďže v škole detičky nútia ukazovať si prštekom na písmenká, ktoré sa práve čítajú. Problémy s písaním a čítaním sú rozobraté v knižke [18]. Každý si isto spomenie na časy, keď sa učil cudzí jazyk. Niekto nám povedal cudzojazyčnú vetu. Kým sme si preložili v mysli slovíčka, nebadane sa nám vytratil celkový zmysel vety. Museli nám tú vetu opakovať ešte raz a ešte raz ... Podobne sa cíti dieťa, ktoré má problémy poskladať si slovo naraz. Tu prístupy štýlu "prečítaj to rýchlo, Marienka" nepomôžu. A ak sa problémy prenesú do vyššieho ročníka, kedy už dieťa musí svižne narábať s čítaním, prejavuje sa to v zhoršených štúdijných výsledkoch. Nedá mi nespomenúť riziká tzv. IQ testov. Testy sú spracované tak, aby výsledky cieľovej skupiny testu zodpovedali normálnemu rozloženiu pravdepodobnosti. Keby bolo veľa detí s poruchou čítania (dislexia), ostatným deťom by sa IQ zvýšilo, ale nie vďaka tomu, že by boli na tom rozumovo lepšie. V nevýhode sú jedinci, ktorý sa vymykajú väčšine.

Podobné problémy sú aj s poruchami písania (disgrafia). Či sa jedná o pravopis alebo samotné napísanie slova.

Doterajšie úvahy nasvedčujú tomu, že proces porozumenia nie je len mechanický prepis do postupnosti symbolov. Dôležitý faktor zohráva i kognitívny aparát. Z hľadiska rozpoznania to je tiež podobné. Prípad disgrafie nasvedčuje, že dotyčná osoba nevie "rozdeliť" slová na písmenká t.j. vnútorne pracuje bez nich. Čiže aj "príjem" slov nie je založený na rozkúskovaní prijatého slova na hlásky.

V [7] sa zapodievajú schopnosťou dieťaťa adaptovať sa na ten, ktorý jazyk. Prišli k záveru, že dieťa do 1.roku života je vybavené akýmsi univerzálnym jazykom, ktorý neskôr prispôsobý jazyku hlavnej starajúcej sa osoby. Neskôr už nie je schopné rozlišovať jemné odtienky cudzieho jazyka.

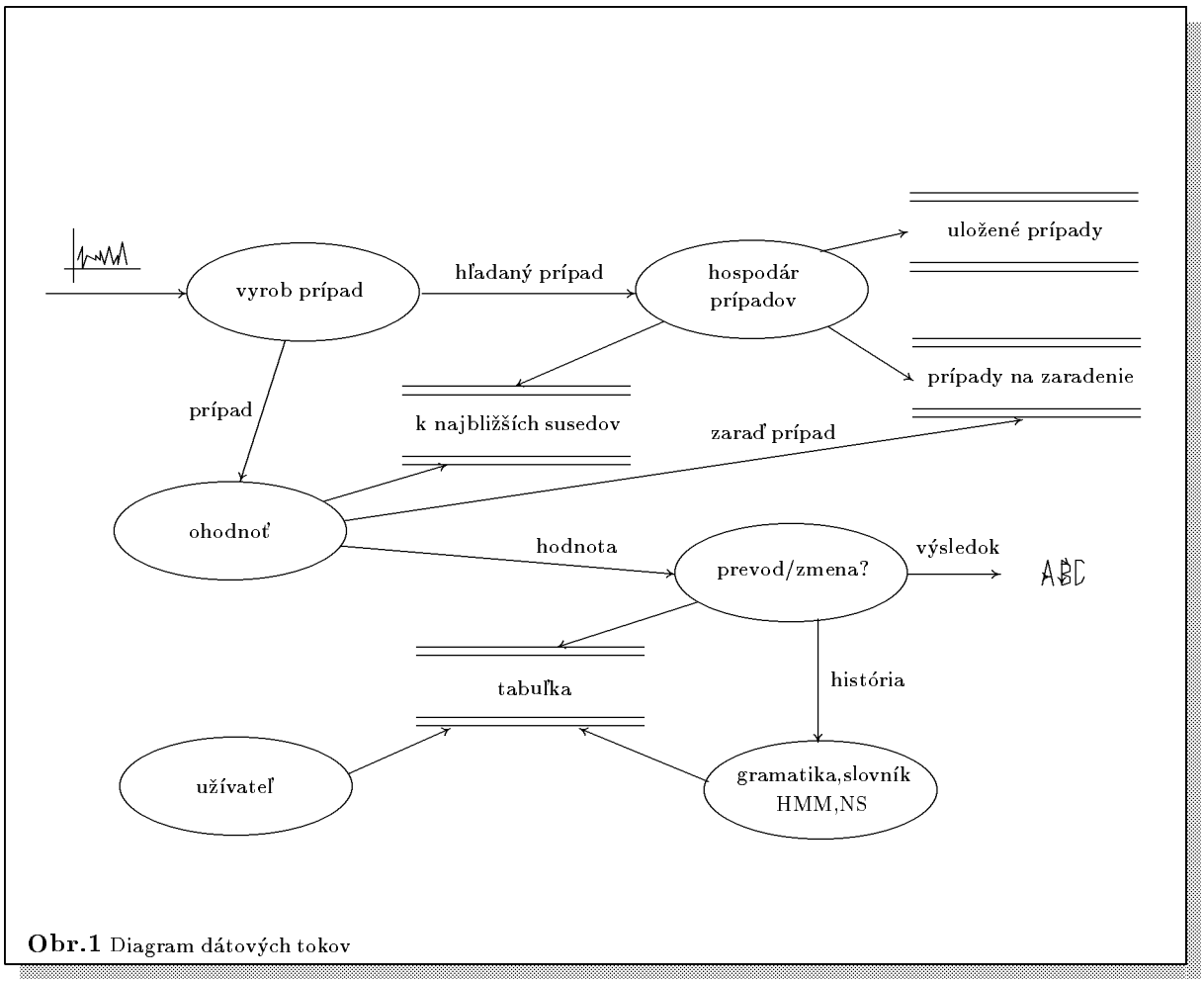
V snahe priblížiť sa hore spomenutým skutočnostiam som sa rozhodol pre nasledovné riešenie. Podľa psycho–fyzikálnych poznatkov je vhodné previesť akustický signál (pozdĺžne vlnenie) na frekvenčné spektrum. Čiastočne to zodpovedá funkcii ucha. Ďalej treba zvoliť vhodný spôsob napodobňovania práce kognitívneho aparátu. Signál rozdelím do časových úsekov (skôr pre výpočtovú zložitosť), s ktorými sa pracuje ako s vektormi vo vektorovom priestore. Analógia s prácou mozgu sa hľadá vo vytváraní zhlukov, ktorým sa priradí význam. Pri príchode nových vektorov sa vytvorí nová oblasť t.j. niečo sa učíme. Vnútorne s tým kognitívny aparát vie narábať. Keď však treba komunikovať, je nutné, aby hovorca a poslucháč mali dohodnutú sémantiku tejto oblasti. Sémantika sa zistí z kontextu, alebo je potrebné zadať ju explicitne.

Je vhodné mať mechanizmus zabúdania. Ak sa niektoré oblasti nepoužívajú, postupne vymiznú (tiež to pomôže pri množstve skladovaných vektorov). V ďaka tomuto spôsobu by mal byť systém schopný adaptovať sa na nového rečníka (oblasti sa prerozdelia). Pokiaľ nie sú radikálne rozdiely medzi rečníkmi, je možné "vytušiť" význami nových oblastí. Keď však Mirka brble, náhodný poslucháč si s tým neporadí, ale mamina jej rozumie dokonale (už si ju Mirka zacvičila).

Na obrázku obr.1 možno vidieť stručnú schému navrhovaného systému. Signál vstupuje do systému. Následne je spracovaný (**vyrob prípad**), použité sú rôzne matematické a štatistické techniky. V ďalšom sa prezentuje ako prípad obsahujúci všetky dôležité vlastnosti. Prípady sú spracovávané a uložené (**hospodár prípadov**), použitá je vhodná stromová štruktúra. Nový prípad je rozhodnutý na základe

predchádzajúcich prípadov (hľadáme oblasť, ku ktorej prislúcha). Zodpovedá tomu proces **ohodnoť**. (Vietor fúka od Prípadovo-orientovaného uvažovania.) Predpokladá sa, že prípady vytvárajú zhluky, ktorým treba priradiť sémantiku (**prevod/zmena?**). Mala by byť možnosť priameho zásahu užívateľa do sémantiky (**užívateľ**). Toto však nestačí, pretože reč je dynamický a kontextový systém a preto treba zahrnúť aj nejaký časový vývoj (**gramatika,slovník,HMM,NS**).

Pri postupnej implementácii som sa snažil implementovať jednotlivé procesy. V kapitole 3 sú rozobraté psycho–fyzikálne aspekty vnímania zvuku človekom. T.j. ako vlastne funguje sluchový orgán a ako sú človekom vnímané zvukové vstupy.
V kapitole 4 je rozobraná problematika viažuca sa k hospodárovi prípadov.
V kapitole 5,6 sú prezentované matematické techniky použité pri procese výroby prípadu.
V kapitole 7 je popísaný proces ohodnotenia, prevodu a užívateľa.
Procesom gramatiky,slovníka,HMM,NS sa v predloženej práci i napriek mojej snahe nezaoberám (nezostal čas).



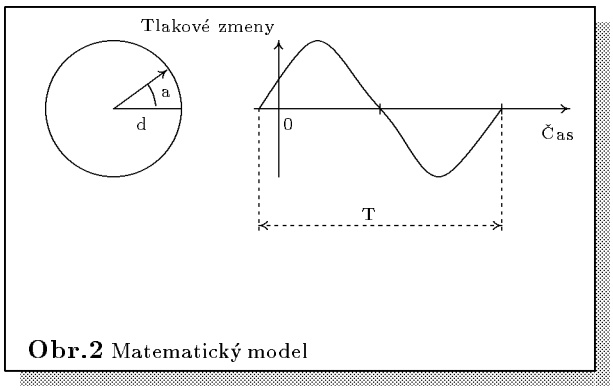

**Obr.1** Diagram dátových tokov

# 3. Psycho-fyzikálne aspekty

## 3.1. Fyzika zvuku.

Zvuk, to sú tlakové zmeny. Šíri sa prostredníctvom častíc či už vo vzduchu, vo vode alebo v kove. Častice si vzájomne odovzdávajú energiu a tým sprostredkúvajú šírenie zvuku.

Zvuk sa správa ako pozdĺžne vlnenie. Je charakterizované frekvenciou, amplitúdou a fázou. Viď. obr.2.




**Obr.2** Matematický model

## 3.2. Furierov rozklad

Reprezentácia zvukovej vlny Furierovým rozkladom (bližšie v ďalšom texte) zodpovedá činnosti ľudského ucha, ktoré však nie je také presné ako matematický model.

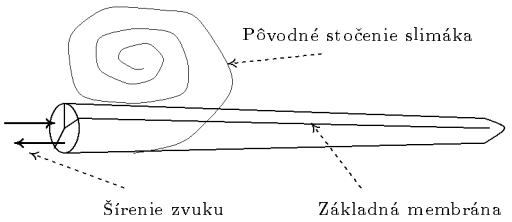
Ak zahráme dva tóny naraz, stále môžme počuť obidva tóny oddelene. (Georg Ohm, Ohmov zákon) To je rozdiel oproti svetelným vlnám kedy dochádza k zmiešaniu (červená spolu so zelenou sa zmiešava a vidíme žltú farbu). Tento rozdiel je spôsobený rôznym princípom daných receptorov.

## 3.3. Intenzita zvuku

Kedže rozdiel medzi prahom počuteľnosti a prahom bolesti je nesmierny, zavádza sa logaritmická stupnica. $10log(I/I_0)$. Jednotky sú decibely $dB$ (desatiny bellu, Alexander Graham Bell). Hodnota vyjadruje porovnanie dvoch intenzít. Štandardne sa volí $I_0$ ako prah počuteľnosti pre frekvenciu zvuku $1000Hz$. Pozri obr.7.

## 3.4. Mechanizmus ucha

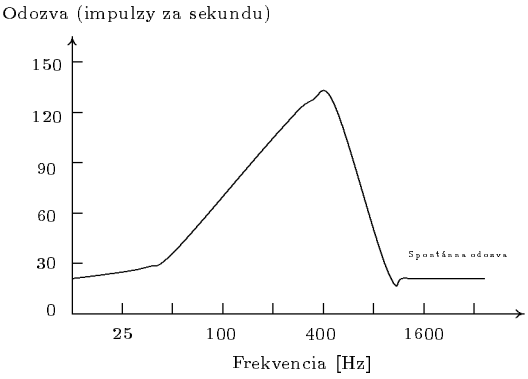
Tlakové zmeny rozkmitajú bubienok. Vibrácie sa pomocou ušných kostičiek (stmienok, kladivko, nákovka) prenašajú na tekutinu ušného slimáka. Vnútro slimáka je pozdĺžne rozdelené na tri časti. Zvuk sa šíri jednou časťou dovnútra (oválny otvor). Na konci ušného slimáka sú časti prepojené a tak zvuk bez odrazu vychádza druhou časťou slimáka von (okrúhly otvor).




**Obr.3** Pohľad na roztočeného ušného slimáka

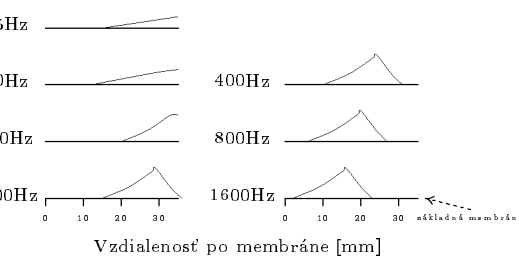
Membrána obsahuje zoskupenia (priečne) 3 až 5 (+1) neurónov. Celkovo je ich približne 23500. S mozgom sú spojené približne 30000 nervovými vláknami. Vibrácie membrány stláčajú neuróny a tie následne produkujú elektrické impulzy.

Špeciálnymi elektródami bola zmeraná aktivita daného neurónu pri rozličných frekvenciách zvuku. V kľude neurón produkuje spontánne impulzy (približne $15Hz$). Postupne sa predkladali sínusové signály daných frekvencií zvuku. Aktivitu neurónu vidno na obr.4.




**Obr.4** Odozva neurónu pre rozličné frekvencie

Zistilo sa, že membrána je rozdelená na oblasti, ktoré reagujú na danú frekvenciu signálu. Tieto oblasti sú určené vzdialenosťou pozdĺž membrány (od vstupu).

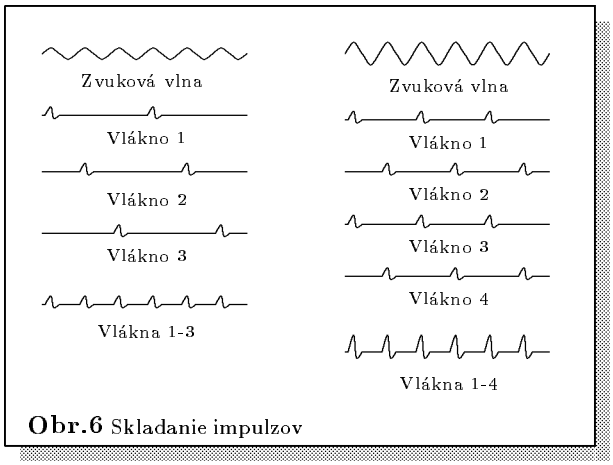



**Obr.5** Aktivované oblasti pozdĺž membrány od vstupu

## 3.5. Kódovanie frekvencie

Aktívny neurón produkuje elektrické impulzy synchrónne so svojou kritickou frekvenciou. Ak je príliš

vysoká, pulzuje s polovičnou, štvrtinovou, ... frekvenciou. Ako kódujú teda neuróny vysoké frekvencie? V danej frekvenčne citlivej oblasti ich je viacej. Každý pulzuje synchrónne, avšak pomalšie. Výsledný súčet však poskladá danú frekvenciu.




**Obr.6** Skladanie impulzov

## 3.6. Kódovanie intenzity

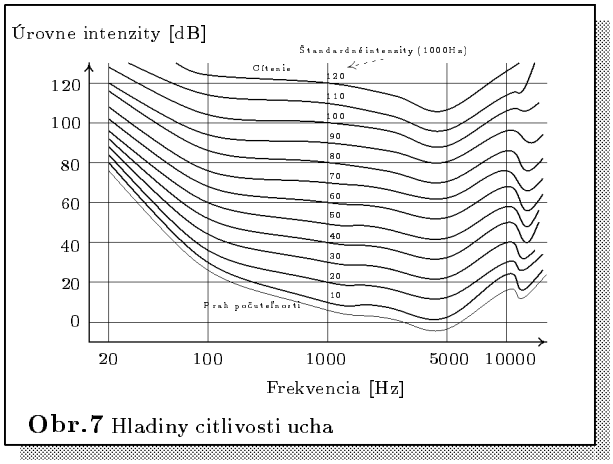
Neurón reaguje na zmenu intenzity zvýšením rýchlosti impulzov. Podobne reagujú i jeho susedia. Výsledný súčet sa prejaví ako intenzívnejšie impulzy. Princíp je znázornený na obr.6.

## 3.7. Intenzita a frekvencia

Z fyzikálneho pohľadu zvýšením intenzity vzrastie i hlasitosť zvuku. Podobne je to i s frekvenciou, čím vyššia frekvencia tým vyšší zvuk. Ucho však tak nefunguje. Do hry vstupuje i samotný mechanizmus prevodu zvuku na nervové impulzy. Napríklad zmenou frekvencie sa mení počutá hlasitosť i výška.

## 3.8. Hlasitosť zvuku

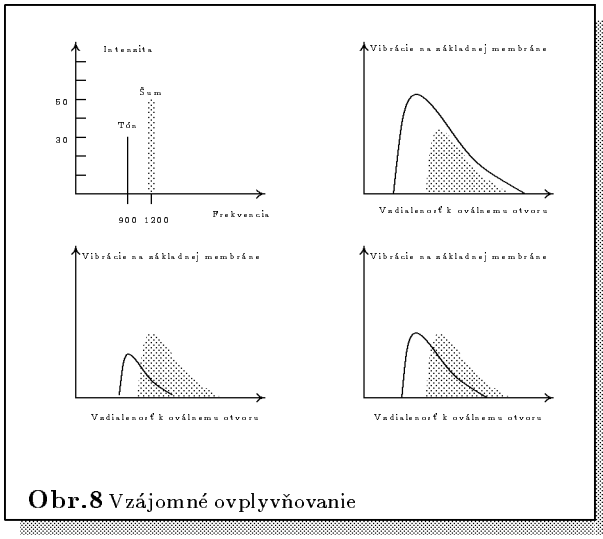
Pre každú frekvenciu sú úrovne hlasitosti rozdelené ináč. Ucho je najcitlivejšie pre frekvencie 200-5000$Hz$.



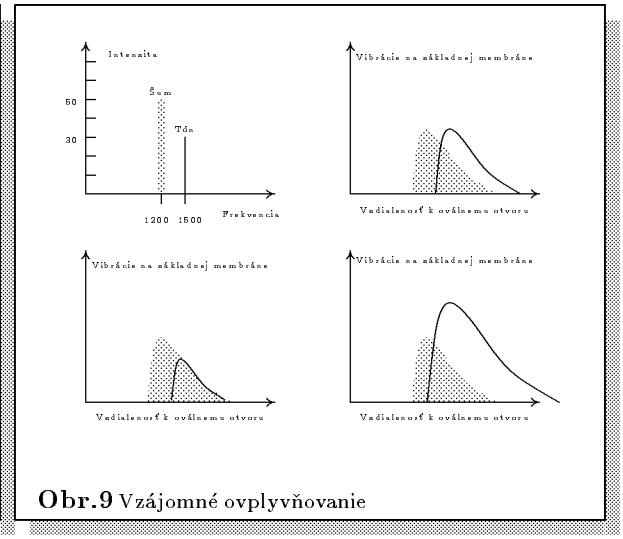

**Obr.7** Hladiny citlivosti ucha

Toto spôsobuje problémy pri nahrávaní koncertných vystúpení. Intenzity jednotlivých frekvencií sú nižšie, čo má za následok zmenu počúvanej nahrávky. Hlavne v oblasti nižších frekvencií, kde je ucho najnecitlivejšie. Rieši sa to elektrickými zariadeniami, ktoré tento fenomén kompenzujú.

Problém je tiež so vzájomným ovplyvňovaním jednotlivých frekvencií v uchu. Pri dostatočnej intenzite jednej frekvencie táto môže prekryť vnímanie frekvencie druhej (s nižšou intenzitou). Jedným z vysvetlení tohto fenoménu sú obr.8 a obr.9.




**Obr.8** Vzájomné ovplyvňovanie

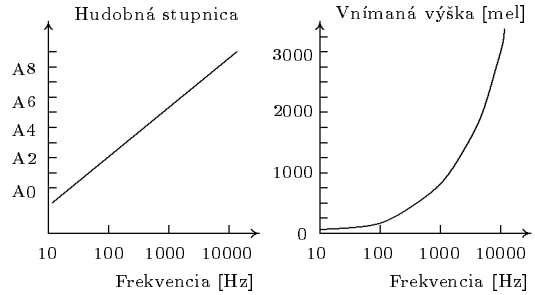



**Obr.9** Vzájomné ovplyvňovanie

Skusmo bol odvodený vzťah $J = k.I^{0.3}$, kde $J$ je vnímaná intenzita. Bolo zvolené pásmo $1000 Hz$.

## 3.9. Výška zvuku

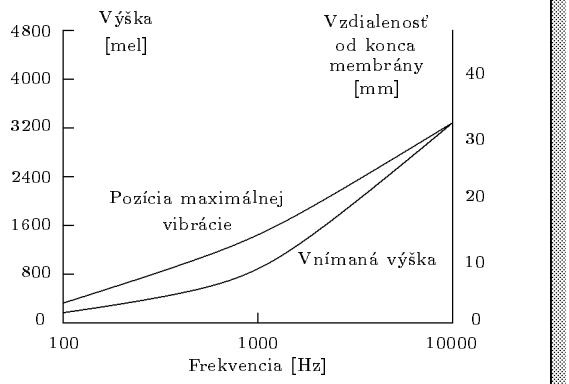
Rozdiely sú i v tejto oblasti vnímania zvuku.




**Obr.10** Súvislosť medzi frekvenciou a vnímanou výškou zvuku

Heuristický vzťah: mels= $2410 log(1.6 * 10^{-3} * f + 1)$. To zapríčiňuje rozdiely v melódii zahranej o niekoľko kláves ďalej a pôvodnej melódii (klavír).

Vysvetlením môže byť pozícia na základnej membráne v uchu. Vnímaná výška má podobný priebeh ako vzdialenosť pozdĺž membrány.

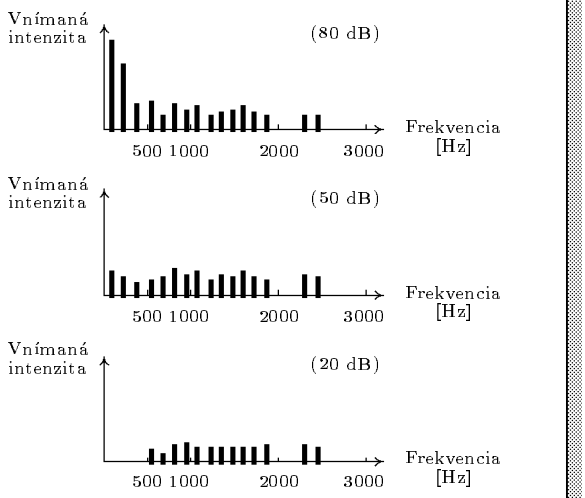



**Obr.11** Súvislosť vzdialenosti a vnemu

Hustota neurónov na membráne je po prvú otočku slimáka približne rovnaká. Potom sa však znižuje. Každý $1mel$ zodpovedá približne posunu o 12 neurónov.

Najmenší vnímateľný rozdiel (jednotka $jnd$) medzi dvomi frekvenciami ($f_1 - f_2$) tiež súvisí so vzdialenosťou pozdĺž membrány. V tomto prípade je však lepšou aproximáciou počet neurónov. $1jnd$ zodpovedá dvom centrám aktivity (frekvenciám), medzi ktorými je pozdĺž membrány 52 neurónov. Pre nižšie frekvencie sme schopní odlíšiť iba vzdialenejšie frekvencie ako pri vyšších frekvenciách (lebo hustota neurónov).

Tón klavíra však nezodpovedá iba sínusovému priebehu vlny, ale sa skladá z viacerých frekvenčných zložiek. Každá zo zložiek má iné kritérium pre vnímanú hlasitosť. Prečo teda, keď zahráme ten istý tón slabšie a slabšie, stále vnímame ten istý tón? Veď nižšie frekvenčné zložky sú už dávno pod hranicou vnímateľnosti?




**Obr.12** Stále slabší a slabší tón po stlačení klávesu klavíra.

Odpoveďou je jav zmiešavania frekvencií. Zmiešaním $1000Hz$ a $1100Hz$ sa objaví i frekvencia $100Hz$. Dva prístupy:

- Nelinearita prenosu zvuku z ušného bubienka na tekutinu zapríčiní i pridanie tejto frekvencie, ktorá je už ďalej rozpoznávaná klasicky. Táto frekvencia môže byť pridaná aj inými otrasmi, ktoré sú následne prenesené na tekutinu (nie len bubienok). (Lenže treba si uvedomiť, že táto je následne utlmená vlastnosťou ucha, pozri obr.7. Teda táto teória nič nevysvetľuje.)
- Všetko funguje ako bolo popísané, avšak mozog si túto frekvenciu poskladá sám z pulzovania všetkých neurónov. Takže, ucho tam nehrá úlohu. Dôkaz tejto teórie spočíva v prekrytí pásma $0 - 500Hz$ intenzívnym šumom, ktorý prekryje podľa prvej teórie vzniknutú frekvenciu. (Použijeme $1000Hz$,

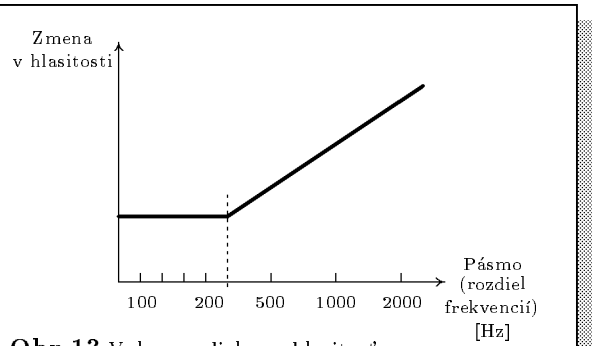
$1200Hz$, $1400Hz$, ... , $2200Hz$). Naďalej však vnímame i $200Hz$. A keď prekryjeme šumom pásmo nad $500Hz$ frekvenciu $200Hz$ už nevnímame (hovorí to proti prvej teórii).

Proti druhej teórii svedčí to, že výška toho istého tónu je rôznymi ľudmi počutá rôzne. Toto by sa dalo vysvetliť tým, že neuróny nie sú rozostavené presne rovnako u každého. Tiež proti druhej teórii svedčí i nutnosť veľkej zmeny tónu pri zmenách vo vyšších frekvenciách. Ako napríklad prikrytie uší rukami, pri ozvenách, v koncertných halách, ... Toto sa však nedeje. Výsledný tón ostáva nezmenený.

Preto sa pristupuje ku kombinácii obidvoch teórií.

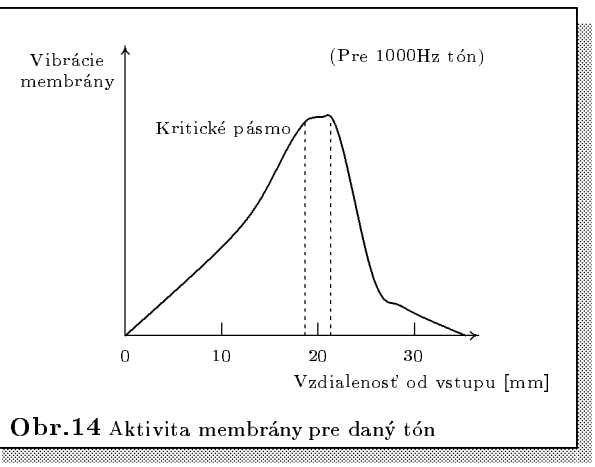
## 3.10. Kritické pásmo

Skúmala sa vnímaná hlasitosť dvoch sínusových tónov v závislosti od ich frekvenčnej vzdialenosti. Od istej hranice sa so zvyšujúcou vzdialenosťou zvyšovala i hlasitosť daného páru tónov. Grafické znázornenie vidieť na obr.13.




**Obr.13** Vplyv rozdielu na hlasitosť

Táto hranica sa nazýva kritickým pásmom. Ak sa tóny nachádzajú v kritickom pásme, sčítava sa ich energia (intenzita) a vníma sa zodpovedajúca hlasitosť. Pre tóny dostatočne vzdialené (mimo kritického pásma) sa sčítava ich jednotlivá vnímaná hlasitosť. Prvý prípad bude v konečnom dôsledku vnímaný slabšie ako druhý prípad. Pre ozrejmenie viď. obr.7.




**Obr.14** Aktivita membrány pre daný tón

Vysvetlenie treba hľadať vo funkcii membrány. Dokladuje to obr.14. Toto pásmo je však pre jednotlivé frekvencie rozdielne. Závisí od pozdĺžneho počtu neurónov, ktorý je 1300. (Jednotkou bude $1cb$.)

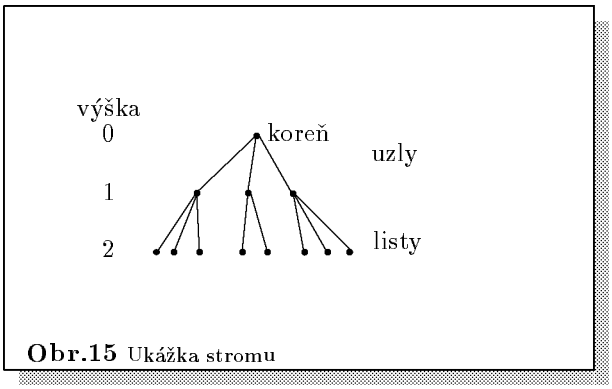
Za povšimnutie stojí: $1mel = 12$neurónov, $1jnd = 52$neurónov, $1cb = 1300$neurónov.

# 4. Strom ako prehľadávacia štruktúra

## 4.1. Problém množstva vektorov

Hlavným problémom je **efektívna** organizácia množstva vektorov (hyperobdĺžnikov) za účelom vyhľadávania a prehľadávania (indexovania) s menším časom, ako je potrebný na **lineárne** prehľadanie. Vzhľadom na to, že množstvá dát sú oveľa väčšie ako primárna pamäť počítača, do popredia sa dostáva i počet prístupov na disk. Nezanedbateľná je i povaha dát. Ak sú statické, núkajú sa lepšie a efektívnejšie techniky. Napríklad redukcia dimenzie **Karhunen-Loéve rozvojom**. S výhodou sa do fázy predspracovania zahŕňajú časovo náročné výpočty, ktoré potom v priebehu prehľadávania nefigurujú. Horšie je to s dynamickými dátami (t.j. okrem vyhľadávania sa dáta i pridávajú a odoberajú), kde ťažko možno predvídať ako nové dáta ovplyvnia už existujúcu prehľadávaciu štruktúru. Lokálne úpravy nemusia v konečnom dôsledku znamenať globálne optimálnu zmenu štruktúry. A vybudovať štruktúru vždy odznovu, znamená venovať tomu viac času, ako je potrebné na lineárne prehľadanie dát.

O lepšej časovej zložitosti prehľadávania treba hovoriť opatrne. Cieľom je pravdaže $O(\log n)$ oproti $O(n)$. Pri statických dátach sú predpoklady sa k tomu priblížiť väčšie ako pri dynamických.




**Obr.15** Ukážka stromu

Ako organizačná štruktúra pre množinu vektorov bol zvolený strom. Každý podstrom je charakterizovaný **obalom**. Obal vymedzuje tú časť priestoru, v ktorej sa nachádzajú všetky obaly daného podstromu. Na najnižšej úrovni vymedzuje obal množinu bodov. Toto uľahčuje prehľadávanie prípadov. Hlavne problém hľadania **k najbližších susedov**. Vďaka obalom sa prehľadávací proces môže **zamerať** na tú časť priestoru (na ten obal), ktorá je nádejnejšia pre výskyt hľadaných susedov.

Z hľadiska efektívnejšieho prehľadávania sa cielene minimalizuje pokryv a prekryv (bližšie zoznam pojmov). Zmenšením pokryvu sa redukuje veľkosť výplne (bližšie zoznam pojmov). Prekryv je z hľadiska prehľadávania rizikovejší, pretože dochádza k zbytočnému viacnásobnému prehľadávaniu oblasti prekrytia.

Základným problémom je voľba metriky vektorového priestoru, vzhľadom na ktorú bol budovaný strom a metriky, vzhľadom na ktorú prebieha hľadanie k najbližších susedov. Používa sa **euklidovská** metrika alebo **vážená euklidovská** metrika. Problém váženej euklidovskej metriky je riešený v [22]. Autori rátajú s rozdielnou maticou pre tvorbu stromu a pre vyhľadávanie.

Ďalším problémom je voľba obalu. Ako obal sa môžu použiť hyperobdĺžniky, hyperkocky, hypersféry, hyperelipsy, ... a tiež ich kombinácie. Obalom sa však nemusí chápať iba ohraničený útvar, tiež sa volia hyperroviny a ich prieniky. Cieľom je čo **najtesnejšie** ohraničiť množinu obalov. T.j. čo najmenší hyperobjem. Výhodné je, aby sa obaly na jednej úrovni čo najmenej prekrývali. Pre hyperobdĺžniky je navrhnuté riešenie v [21]. Čím sú väčšie prekrytia, tým viac času strávime pri vyhľadávaní k najbližších susedov. Výhody a nevýhody hyperobdĺžnikov a hypersfér sú rozobraté v [19], kde to riešia voľbou obalu, ktorý vznikne prienikom hyperobdĺžnika a hypersféry. O najtesnejšiu aproximáciu hypersféry sa snažia v [20]. Problém je v tom, že výpočet najtesnejšej hypersféry je časovo veľmi náročný. Hlavnou nevýhodou hypersféry a hyperkocky je ich rovnomerná rozpínavosť všetkými dimenzionálnimi smermi a tým rýchlejšie zväčšovanie ponímaného hyperobjemu. Je to badateľné hlavne pri množine vektorov, kde

sa dimenzie správajú podľa rôznych štatistických rozdelení. Nárast ohraničujúcej hypersféry potrebný pre jednu dimenziu, môže byť zbytočný pre inú dimenziu. (Preto sa zavádza hyperelipsa, ktorá rieši tento problém.)

Najmä pri statických množinách vektorov sa znižuje ich dimenzia a tým sa zvyšuje rýchlosť. Najlepšou redukciou dimenzie pri najmenšej chybe je Karhunen-Loéve rozvoj. Za nové bázové vektory podpriestoru sú vybraté vlastné vektory.

Metódy vo všeobecnosti:

a) vektory – hlavná idea je rozdeliť celý vektorový priestor do disjunktných subregiónov, ktoré obsahujú zvolený počet vektorov. Pridaním nového vektoru môže nastať preplnenie regiónu, čo sa rieši **rozdelením** pomocou hyperroviny. Rozdelenia delíme podľa:
   - pozície
     pevné - nezohľadňujú sa vektory (napr. rozdeliť región vždy na polovicu)
     adaptovateľné - podľa povahy vektorov (napr. rozdeliť región na rovnaký počet vektorov v každej časti, odhaliť prirodzené zlučovanie sa vektorov a neporušiť ho delením, ... )
   - dimenzie
     rozdelenie v jednej dimenzii
     rozdelenie vo viacerých dimenziách
   - pôsobiska
     lokálne - rozdelí iba daný subregión
     globálne - rozdelí všetky zasiahnuté regióny

b) hyperobdĺžniky
   - Transformácia do viacrozmerného priestoru, dimenzia sa zdvojnásobí. Ďalej postupujeme ako s vektormi.
   - Transformácia viacrozmerného priestoru do jednorozmerného. Používajú sa priestor vypĺňajúce krivky. Cieľom transformácie je, aby vektory, ktoré sú blízko vo viacrozmernom priestore, boli blízko i v jednorozmernom priestore. Ďalej postupujeme štandardnými technikami nad jednorozmernými dátami.
   - Podobne ako vektory, avšak treba dať pozor na prípad, keď deliaca hyperrovina prechádza nejakým hyperobdĺžnikom. Riešenia:
     1) rozdeliť hyperobdĺžnik na dve časti
     2) všeobecná orientácia hyperroviny (nielen kolmé smery)
     3) prekrývanie sa subregiónov

Obvyklé operácie:

- vektorový dotaz – nájdi všetky objekty obsahujúce daný vektor
- regiónový dotaz – nájdi všetky objekty, ktoré zasahujú do regiónu
- nájdi k najbližších susedov (**kns**) – k danému vektoru nájdi k najbližších vektorov podľa zvolenej metriky
- pridaj,odober – vektor pridaj, vektor odober (manipulácia so štruktúrou)

Heslovité využitie:

- Case-based reasoning aplikácie
- organizácia pravidiel v expertnom systéme (rýchle nájdenie vhodného pravidla)
- počítačová grafika (miesto vektorov sa prehľadávajú hyperobdĺžniky, ktoré slúžia ako obálky telies)
- databázy sekvencií napr. DNA
- kartografia
- VLSI návrhy (veľa obdĺžnikov reprezentujúcich elektronické elementy)
- počítačové videnie a robotika
- databáza obrázkov (črty, charakteristiky tvoria vektor) napr. na CD-rome

## 4.2. Použité pojmy

**strom** –graf, kde medzi ľubovolnými dvoma vrcholmi existuje práve jedna cesta

**uzol** –vrchol stupňa $\geq 2$

**list** –vrchol stupňa = 1

**koreň** –hlavný zvolený vrchol

**otec** –predchodca vrchola v smere ku koreňu

**syn** –nasledovník vrchola v smere od koreňa

**bratia** –vrcholy majúce spoločného otca

**výška vrchola** –dĺžka cesty od koreňa k vrcholu

**výška stromu** –maximálna z výšok vrcholov daného stromu

**vektor,vektorový priestor,podpriestor,dimenzia,báza** –štandardné použitie

**euklidovská metrika** $-d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T(\vec{x}-\vec{y})}$

**vážená euklidovská metrika** $-d(\vec{x},\vec{y},\mathbf{W}) = \sqrt{(\vec{x}-\vec{y})^T\mathbf{W}(\vec{x}-\vec{y})}$, kde $\mathbf{W}$ je symetrická matica váh

**hyperrovina** –podpriestor menšej dimenzie

**hypersféra** $-\{\vec{x},\quad d(\vec{s},\vec{x}) <= r\}$

**hyperobdĺžnik** $-\{\vec{x},\quad \forall i : a_i <= x_i <= b_i\}$

**oblasť,región** –časť vektorového priestoru; kompaktná množina vektorov

**prekryv (overlap)** –prienik oblastí uzlov v danej výške stromu

**výplň (dead-space)** –prázdny priestor pokrývaný uzlami

**pokryv (coverage)** –celková oblasť, ktorú zahŕňajú uzly v danej výške stromu

## 4.3. Karhunen-Loéve rozvoj

Majme množinu vektorov $\vec{x}_1,\ldots\vec{x}_p$ z vektorového priestoru $R^n$. Hľadáme transformáciu $\mathbf{W}$, ktorá transformuje vektory do vektorového priestoru $R^m$, kde $m < n$. Touto transformáciou vznikne chyba, ktorú chceme minimalizovať.

$$\min\sum_i ||\vec{x}_i - \vec{x}'_i||^2, \qquad \vec{x}'_i = a_{i1}.\vec{e}_1 + \cdots + a_{im}.\vec{e}_m,$$

kde $\vec{e}_1,\ldots,\vec{e}_m$ sú bázou priestoru $R^m$, pričom je požadované, aby $\vec{e}_1,\ldots,\vec{e}_m$ boli ortonormálne t.j.

$$\begin{aligned} <\vec{e}_i,\vec{e}_j> &= 0, \qquad i \neq j \\ <\vec{e}_i,\vec{e}_j> &= 1, \qquad i = j. \end{aligned}$$

Počítajme ďalej

$$\min\sum_i ||\vec{x}_i - \vec{x}'_i||^2 = \min\sum_i <\vec{x}_i - \vec{x}'_i, \vec{x}_i - \vec{x}'_i> = \min\sum_i \vec{x}_i{}^T.\vec{x}_i + \vec{x}'_i{}^T.\vec{x}'_i - 2.\vec{x}_i{}^T.\vec{x}'_i$$

Z hľadiska minimalizácie má ďalej zmyseľ uvažovať iba:

$$\min\sum_i((\sum_j a_{ij}.\vec{e}_j)^T.\sum_j a_{ij}.\vec{e}_j - 2.\vec{x}_i{}^T.\sum_j a_{ij}.\vec{e}_j) = \min -\sum_i\sum_j {a_{ij}}^2$$

t.j.

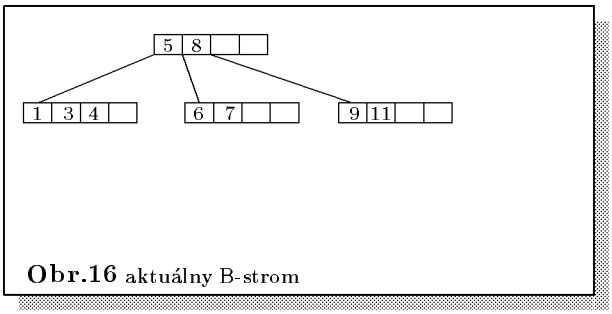
$$\max\sum_j\sum_i(\vec{x}_i{}^T.\vec{e}_j).(\vec{x}_i{}^T.\vec{e}_j) = \max\sum_j\sum_i \vec{e}_j{}^T.(\vec{x}_i.\vec{x}_i{}^T).\vec{e}_j =$$

$$\max\sum_j \vec{e}_j{}^T.(\sum_i \vec{x}_i.\vec{x}_i{}^T).\vec{e}_j = \max\sum_j \vec{e}_j{}^T.\mathbf{M}.\vec{e}_j$$

Za $\vec{e}_1,\ldots,\vec{e}_m$ sú zvolené vlastné vektory matice $\mathbf{M}$, prislúchajúce najväčším $m$ vlastným číslam matice $\mathbf{M}$.

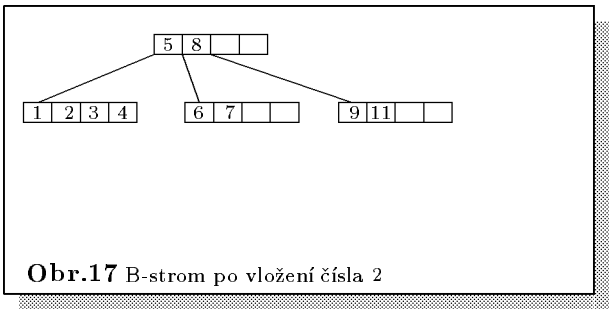
## 4.4. B-strom

B-strom je dátová štruktúra, ktorá umožňuje hľadanie, vkladanie, odoberanie a prehľadávanie medzi číslami. Každý uzol má $n+1$ až $2n+1$ synov a obsahuje $n$ až $2n$ čísel, ktoré sú vzostupne usporiadané. Koreňový uzol môže mať podľa okolností i menej synov. Ak sa pri hľadaní dané číslo nenachádza v uzle pokračuje sa v prehľadávaní toho podstromu, do ktorého intervalu dané číslo padne. Obrázok zobrazuje príklad stavu štruktúry.



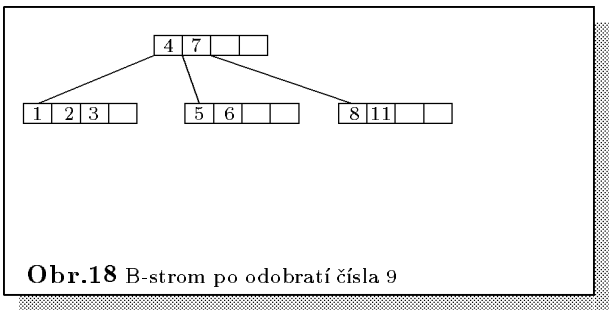

**Obr.16** aktuálny B-strom

Každý uzol si vlastne pamätá intervaly (jednorozmerné hyperobdĺžniky) pridelené jednotlivým synom. Hranice intervalov sú však zároveň i platnými údajmi (číslami).




**Obr.17** B-strom po vložení čísla 2

Ak je uzol alebo list plný dochádza k **rozdeleniu** na dva nové uzly alebo listy. Podobný prípad nastane i pri vymazávaní, kde však treba neúplný uzol doplniť buď pomocou súrodenca, alebo zrušiť uzol a prvky opäť vložiť do stromu.




**Obr.18** B-strom po odobratí čísla 9

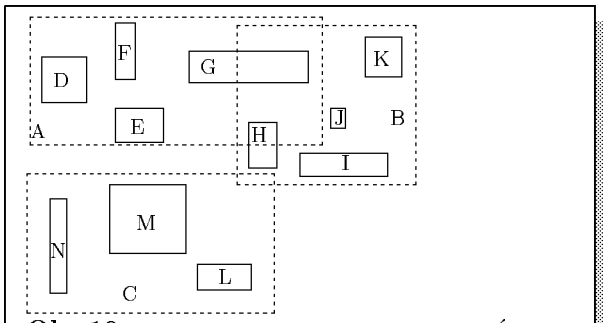
Pri vkladaní a odoberaní treba strážiť nie len konzistentnosť (počet synov uzla), ale i rovnomerný rast stromu a tým zaručiť optimálnu výšku stromu.

Prvkami nemusia byť len čísla, ale ľubovaľné usporiadané prvky. Napríklad slová prirodzeného jazyka, čo sa s obľubou používa v databázových systémoch.

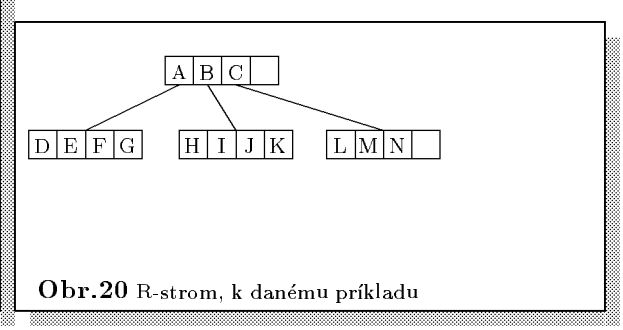
Táto štruktúra sa stala východiskom pre ďalšie štruktúry, ktoré zohľadňujú i mnohorozmernosť prvku a voľbu metriky.

## 4.5. R-strom

Táto organizačná štruktúra vznikla z potreby rýchleho vyhľadávania hyperobdĺžnikov, ktoré sa nachádzajú vo vymedzenej oblasti viacrozmerného priestoru. R-strom vznikol priamym rozšírením B-stromu do viac dimenzií. Listy sú tvorené jednotlivými hyperobdĺžnikmi, ich počet v liste je vymedzený maximálnou a minimálnou hranicou. Uzly obsahujú obaly - hyperobdĺžniky, ktoré zahŕňajú všetky hyperobdĺžniky prislúchajúceho syna. Možno vidieť na obr.19. Strom k danému príkladu je na obr.20. Hyperobdĺžnik $A$ **obaluje** $D, E, F, G$. Hyperobdĺžnik $B$ obaluje $H, I, J, K$. A napokon hyperobdĺžnik $C$ obaluje $L, M, N$. Obal bol vypočítaný ku zvoleným objektom a nie naopak. Preto $G$ nie je zahrnuté v obale $B$. Podobne to platí aj pre $H$.

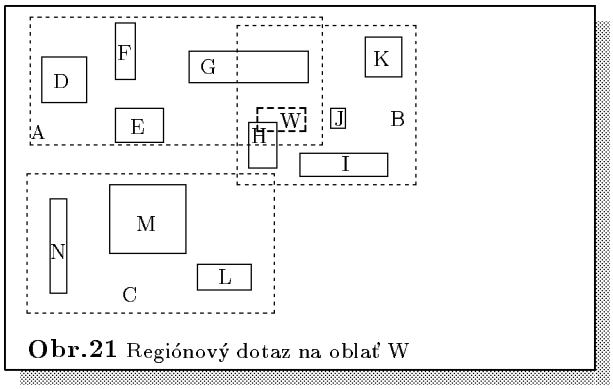



**Obr.19** Obaly, ktoré zahŕňajú dané hyperobdĺžniky




**Obr.20** R-strom, k danému príkladu

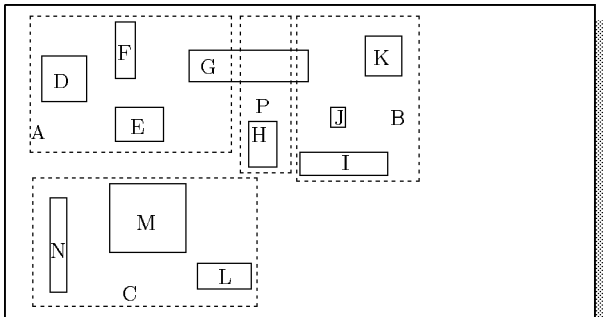
Najkritickejším miestom z hľadiska prehľadávania je prekryv. Na obr.21 je dotazovaná oblasť W. Pretože $W \cap A \neq \emptyset$ je nutné prehľadať syna $A$. Rovnako platí i $W \cap B \neq \emptyset$ a preto je nutné prehľadať i syna $B$. Týmto sa môže prehľadávanie degradovať na lineárne oproti požadovanému logaritmickému.



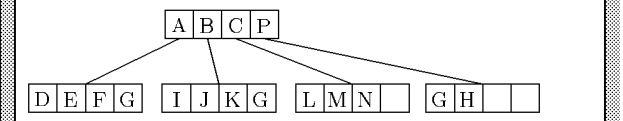

**Obr.21** Regiónový dotaz na oblať W

## 4.6. R$^+$-strom

Prekryvu sa dá vyhnúť, ak sa umožní rozdeľovať obalové hyperobdĺžniky v uzloch. Listové hyperobdĺžniky sa nerozdeľujú (môžu obsahovať netriviálne geometrické útvary, ... ) iba sa zopakujú. Strom a grafické znázornenie príkladu sú na obr.22 a obr.23. Hyperobdĺžnik $G$, na rozdiel od obr.19, je uvedený v podstrome $A, P, B$, aby sa pri dotazovaní bral do úvahy pre každý obal. (Treba si uvedomiť, že teraz sú obaly disjunktné.)




**Obr.22** Obaly. Príklad rozdelenia hyperobdĺžnika, ktorý je listový a tak sa len zopakuje.

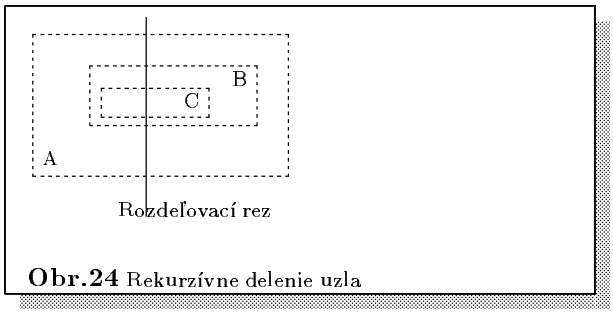



**Obr.23** $R^+$-strom, k danému príkladu

Zaujímavý je hyperobdĺžnik $G$, ktorý by bol rozdelený na tri časti, keby nebol na úrovni listov. Dotazovanou oblasťou W z obr.21 sa v tomto prípade prehľadáva iba syn $P$, lebo iba $W \cap P \neq \emptyset$. (A tu sa demonštruje potreba $G$ zahrnúť v každom obale.)

Odstránenie prekryvu je čiastočne zaplatené zväčšením počtu synov a tým i nepriamo výšky stromu. Zmena výšky je však nepatrná v porovnaní s tým, koľko ušetríme času pri vyhľadávaní.

V prípade R-stromu sa rozdelenie uzla šíri rekurzívne iba smerom ku koreňu, zatiaľ čo v prípade $R^+$-stromu sa šíri rekurzívne i smerom k listom. Obrázok obr.24 ukazuje prípad, keď treba rozdeliť oblasť $A$. Rozdelíme ju, ale nemáme zaručené či neobsahuje takú podoblasť $B$ ako na obrázku, ktorú treba rekurzívne rozdeliť. Pri najhoršom prípade rez rozdelí všetkých synov podoblasti $A$, ... Toto spomaľuje vkladanie nových hyperobdĺžnikov a tak sa $R^+$-strom častejšie používa na statické dáta. Autori v [21] predkladajú i algoritmus, ktorý predspracuje strom lepšie ako postupným vkladaním.




**Obr.24** Rekurzívne delenie uzla

## 4.7. R*-strom

Je miernou modifikáciou R-stromu. Je zmenený spôsob vkladania a rozdeľovania. Zavádza sa mechanizmus **znovuvloženia** vektora.

## 4.8. P-strom

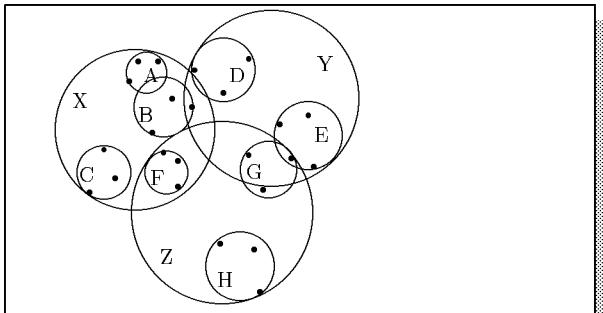
Táto štruktúra je priamym rozšírením R-stromu. Obal je tvorený prienikom viacerých hyperobdĺžnikov, ktoré majú rozličnú orientáciu. (Polyeder.)

## 4.9. K-D-B-strom

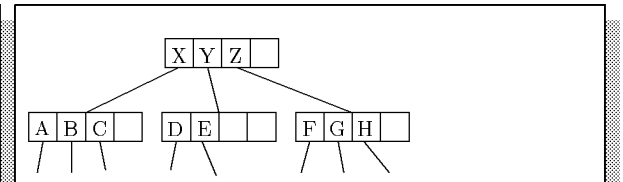
Vychádza z B-stromu. Priestor je rekurzívne delený hyperrovinami. Na každej úrovni stromu sú regióny disjunktné. T.j. vektorový dotaz bude vždy logaritmický. Dochádza však i k vytváraniu prázdnych oblastí, čo znižuje silu kns dotazov.

## 4.10. SS-strom

Listy sú tvorené vektormi a uzly obsahujú obaly-hypersféry. Pre hľadanie najbližších susedov sa tento spôsob a jeho kombinácie ukázal najvýhodnejší.




**Obr.25** SS-strom štruktúra




**Obr.26** SS-strom k danému príkladu

Obal je tvorený stredom, ktorý je vypočítaný ako centroid synov a polomer je určený ako najmenší obsahujúci vektory synov. Ak synovia nie sú listy, používajú sa ich centroidy. Nový vektor sa vkladá vždy do najbližšej hypersféry na danej úrovni stromu. Ak je list plný, je potrebné ho rozdeliť. Vypočíta sa disperzia v každej súradnici a rozdelí sa podľa súradnice s najväčšou disperziou. Výhodou oproti R-stromu je v polovičnej úspore pamäťového miesta (stred a polomer oproti dvom rohom hyperobdĺžnika).

## 4.11. VAMSplit R-strom

Verzia optimalizovaného R-stromu. Vektory sú rekurzívne rozdeľované hyperrovinou, ktorá je ortogonálna na dimenziu s najväčšou disperziou a je približným mediánom. VAMSplit R-strom vďaka predspracovaniu predčí R*-strom i SS-strom. Vhodný je na statické dáta.

## 4.12. TV-strom

Zvyšuje silu R*-stromu redukciou dimenzie a zapínaním/vypínaním dimenzií. (Redukcia je napríklad KL-rozvojom.) Dimenzie sú zoradené podľa významnosti. Ak majú všetky vektory v podstrome rovnakú najvýznamnejšiu zapnutú súradnicu, táto je vypnutá. A menej významná súradnica v poradí, je zapnutá. Podstrom sa vytvára iba podľa zapnutých dimenzií.

Tento prístup musí spĺňať nasledujúce podmienky, aby bol efektívny:

- dimenzie sa dajú zoradiť podľa významnosti
- existujú také vektory, ktoré umožnia prepínanie dimenzií.

Druhý bod nie je výhodný pre reálne dáta a sila prvého bodu sa dá využiť pri ľubovolnej inej štruktúre.

## 4.13. X-strom

Ďalšia varianta R*-stromu, ktorá používa bezprekryvové rozdelenia a špeciálne **superuzly**. Tento superuzol je uzol väčších rozmerov ako štandardný uzol za účelom obídenia prekryvu. Rozmer superuzla nie je obmedzený. Tam, kde by sa nedalo vyhnúť prekryvu, použije sa superuzol. Hlavná idea spočíva vtom, že radšej prehľadať lineárne ako umožniť prekryv, ktorý môže prehľadanie zhoršiť i oproti lineárnemu.

## 4.14. SR-strom

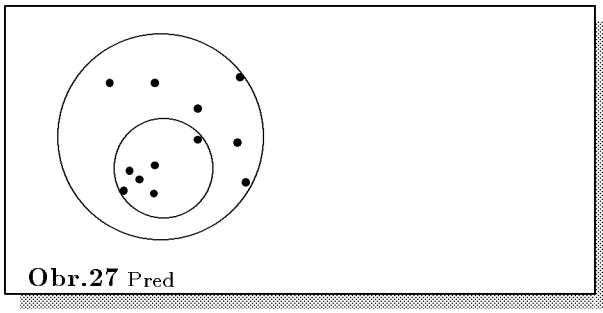
Kombinovaný variant SS-stromu a R-stromu. Obal je tvorený prienikom hypersféry a hyperobdĺžnika. Vychádaza sa z:

- SS-strom predčí R-strom v hľadaní kns.
- hypersféra má menší priemer oproti hyperobdĺžniku. Vďaka čomu je SS-strom efektívnejší.
- hyperobdĺžnik má menší hyperobjem ako hypersféra. Hypersféra narastá všetkými smermi oproti hyperobdĺžniku. Väčší hyperobjem dáva predpoklady väčšiemu prekryvu a tým dochádza k zhoršeniu efektívnosti.

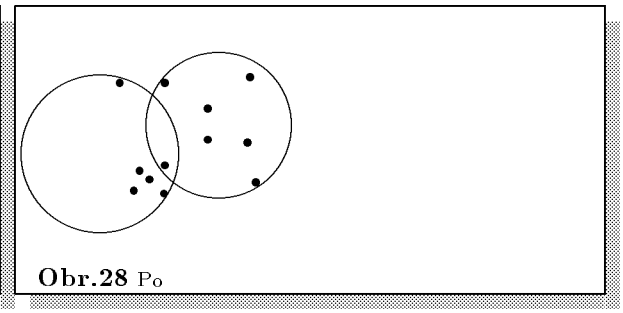
T.j. jedno i druhé má svoje výhody i nevýhody. Autori SR-stromu spojili výhody obidvoch.

## 4.15. SS$^+$-strom

Vylepšenie SS-stromu. Pri rozdeľovaní sa použije všeobecný algoritmus na hľadanie k oblastí (k-means clustering algorithm), ktorý je popísaný napríklad v [3]. Tým sa zmenší prekryv oproti spôsobu v SS-strome. Na obrázku obr.27 vidieť vnorenie. Po aplikovaní algoritmu je výsledok na obr.28.



**Obr.27** Pred



**Obr.28** Po

Ďalším vylepšením je hľadanie najmenšej hypersféry. Algoritmus na nájdenie najmenšej hypersféry je časovo náročný, tak sa autori v [20] snažia aspoň o priblíženie.

## 4.16. Implementácia

Za základ bol zvolený SS-strom. List obsahuje 10 až 20 vektorov.

$$L = (\vec{x}_1, \ldots, \vec{x}_i), \quad 10 \leq i \leq 20$$

Každý vektor je označený farbou oblasti, ktorá sa používa pri klasifikovaní. Uzol obsahuje 10 až 20 obalov–hypersfér.

$$U = ((S_1, w_1), \ldots, (S_i, w_i)), \quad 10 \leq i \leq 20$$

$S_i$ je hypersféra, $w_i$ určuje celkový počet vektorov v danom podstrome (váha). Celkový počet vektorov v strome som ohraničil. T.j. po dosiahnutí danej hranice, sa pri každom vložení, musí jeden vektor odobrať. Cieľom bolo, aby pri zmene rečníka došlo k postupnej obmene celého množstva vektorov za nové, ktoré vytvoria (možno iné) oblasti charakterizujúce nového rečníka. Z tohto hľadiska sa odoberá vždy najstarší vektor.

Program je písaný v ANSI C programovacom jazyku. Skúšaný pod operačným systémom Linux (RH 5.1). Cieľom nasledujúceho popisu nebolo do detailov rozoberať programové konštrukcie. Rovnako priložené fragmenty programu slúžia iba pre orientáciu. (Vytlačiť úplný program by bolo zbytočné a neekonomické.)

## 4.17. Vlož

vstup: $\vec{x}$

1.) postupuje sa od koreňa. Vyberie sa tá vetva, ktorej stred obalu je najbližšie ku vkladanému vektoru.
2.) Opakujeme 1.krok, pokým nie je list.
3.) vložíme vektor $\vec{x}$ do listu.
4.) ak je list plný, prerozdelí sa na dve časti. Rozdelenie sa šíri späť ku koreňu.
5.) prerozdelenie sa určí podľa súradnice s najväčšou disperziou. Obsah sa rozdelí na dve polovice usporiadaním podľa mediána (vypočítaného priemeru).

$$d_j = \sum_i (x_i(j) - x_{(}j))^2, \quad i \in L \text{ alebo } U \text{podľa prípadu}, \quad 1 \leq j \leq dimenzia,$$

$$\vec{x} = \frac{\sum_i \vec{x}_i}{|L| \text{ alebo } |U|},$$

kde $d_j$ je disperzia v danej súradnici (dimenzii), $\vec{x}_i$ resp. $x_i(j)$ je vektor (stred oblasti) resp. jeho $j$-ta zložka a $\vec{x}$ resp. $x(j)$ je priemer vektorov resp. jeho $j$-ta zložka. Ďalej vyberiame maximum

$$d_k = \max_{1 \leq j \leq dimenzia} d_j$$

a podľa $k$-tej súradnice (resp. jej priemernej hodnoty) prerozdelíme vektory (oblasti).
6.) nová hypresféra sa vypočíta

$$Sx(j) = \frac{\sum_{i=1}^{n} S_i x(j).w_i}{\sum_{i=1}^{n} w_i}, \quad 1 \leq j \leq dimenzia,$$

kde $Sx(j)$ a $S_i x(j)$ je $j$-ta súradnica stredu počítanej hypersféry a $i$-tej hypersféry. $Sr$ resp. $S_i r$ je polomer počítanej resp. $i$-tej hypersféry. Ešte treba vypočítať polomer

$$Sr = \max_{1 \leq i \leq n} (d(\vec{S_i x}, \vec{Sx}) + S_i r),$$

kde $n$ udáva počet synov ($|L|$ alebo $|U|$), ku ktorým hľadáme obal. Treba si uvedomiť, že v prípade listu sú polomery $S_i r$ nulové, $S_i x$ zodpovedá $x_i$ a postupuje sa rovnako.

Proces vkladania sa odvíja od funkcie TreeInsert(), ktorá využije funkciu InsertBod(). Možno pozrieť v priloženom fragmente programu. (Kompletné programy na priloženej diskete alebo u autora.)

## 4.18. Zruš

Najprv sa vymaže daný vektor z listu. Ak v liste ostalo málo vektorov, list je zrušený a uvoľnené vektory sa znovuvložia do stromu. Zrušenie sa šíri rekurzívne ku koreňu. Obaly na rozdiel od vektorov sa znovuvložia vo výške, v ktorej boli pôvodne.

Realizácia začína funkciou DeleteBod().

## 4.19. Nájdi kns

Hľadá sa kns, ktorí sú v danom polomere od vektora $\vec{x}$, t.j. vpadnú do $S(\vec{x}, r)$, kde $r$ udáva "štartovací" polomer prehľadávania. Postupné prehľadávanie. Na začiatku sú v agende obaly z koreňa stromu. Deti najbližšieho obalu z agendy sú pridané do agendy. Opakujeme, až kým nedospejeme k listu. Priebežne zmenšujeme i $r$. Skončíme, keď sa agenda vyprázdni. Tie obaly, ktoré nezasahujú do $S(\vec{x}, r)$ ostanú

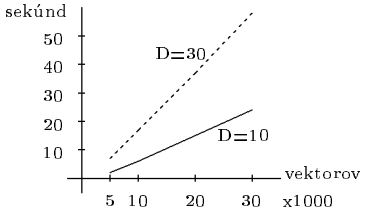
neprehľadané. Pre kvantitatívnu redukciu prehľadávaného množstva vektorov je dôležitá vhodná voľba počiatočného $r$.

V priloženom programe je to implementované funkciou TreeSearch(). Pre porovnanie je implementovaná aj lineárna verzia TreeSearchN().

## 4.20. Časové a iné výsledky

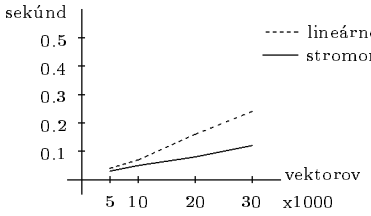
Testovanie prebiehalo na procesore Pentium$^R$ 100MHz, pod operačným systémom Linux. Boli zvolené náhodné vektory $\langle 0,1)^D$, kde $D$ je dimenzia vektorov. Vždy sa hľadalo 20 najbližších susedov k náhodne zvolenému vektoru.

Obrázok obr.29 poskytuje pohľad na graf času potrebného na vytvorenie stromu pre daný počet vektorov. Vplyv dimenzie $D$ je tiež nezanedbateľný.

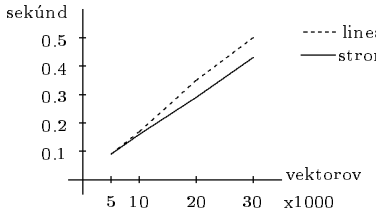



**Obr.29** Čas potrebný na vytvorenie stromu.

Časy vyhľadávania 20 najbližších susedov sú na obr.30 a obr.31. V porovnaní s lineárnym vyhľadávaním nastalo zrýchlenie. Keď sa však porovnajú obr.30 a obr.31 vzájomne, možno tiež sledovať nepriaznivý vplyv zväčšovania dimenzie.

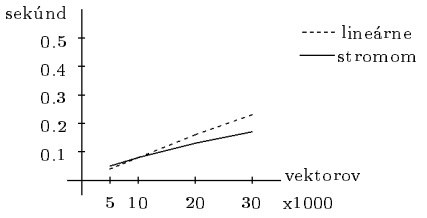



**Obr.30** Priemerný čas vyhľadania 20 najbližších susedov s polomerom r=0.4, D=10



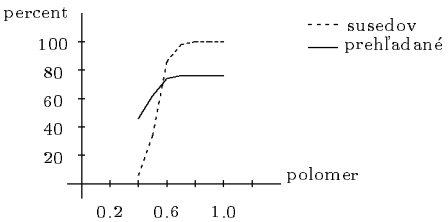

**Obr.31** Priemerný čas vyhľadania 20 najbližších susedov s polomerom r=0.4, D=30

Ako si možno na obr.32 všimnúť dôležitý je aj polomer vyhľadávania. Viď. tiež obr.30.




**Obr.32** Priemerný čas vyhľadania 20 najbližších susedov s polomerom r=20.0, D=10

Celkový pohľad na úspešnosť najdenia 20 najbližších susedov v závislosti na vyhľadávacom polomere je na obr.33. Rovnako tam možno vidieť i percentuálny počet prehľadaných vektorov. Pre lineárne prehľadávanie by to bolo 100%.




**Obr.33** Percentuálny počet najdených susedov a prehľadaných vektorov v závislosti od polomeru.

V najhoršom prípade by sa mali prezrieť všetky vektory. Tento prípad je zriedkavý. Vďaka zmenšovaniu polomeru $r$ v algoritme prehľadávania sa mu dá vyhnúť. (Hneď po 20-tich vyskúšaných vektoroch sa polomer upraví na vzdialenosť najvzdialenejšieho z nich. Vo väčšej vzdialenosti už nemá zmysel hľadať susedov.)

## 4.21. Zdrojový kód

Nasledujú fragmenty zdrojového súboru, ktoré sa podieľajú na udržiavaní a prehľadávaní stromovej štruktúry. Úplné kódy sú na priloženej diskete.

```
.
.
.
struct Sfera {
  double x[DIMENZIA];          /* Stred. */
  double r;                    /* Polomer. */
};
struct Kocka {
  double a[DIMENZIA];
  double b[DIMENZIA];          /* Pozicia ohranicujuceho hyperobdlznika. */
};
struct Bod {
  double x[DIMENZIA];          /* Suradnice bodu. */
  int oblast;                  /* Cislovanie oblasti. */
  struct Uzol *uzol;           /* Uzol(list) v ktorom sa nachadza. */
  int ktory;                   /* A jeho pozicia. */
};
struct Obal {
```

```
  struct Sfera sfera;           /* Ohranicujuca sfera. */
  struct Kocka kocka;           /* Ohranicujuci hyperobdlznik. */
  int pocet_bodov;              /* Pocet bodov v celom podstrome. */
  struct Uzol *syn;             /* Podstrom, ktoremu je obalom.  */
};
struct Uzol {
  int uroven;                   /* List => uroven=0. */
  int pocet;                    /* Pocet deti. */
  union {
    struct Obal *obal[DETI_MAX];/* uroven > 0, Vnutorny uzol */
    struct Bod  *bod[DETI_MAX]; /* uroven = 0, List */
  } obsah;
  struct Uzol *otec;            /* Ukazovatel na otca daneho uzla. */
  int ktory;                    /* A jeho pozicia. */
};
.
.
.
struct Bod body[POCET_BODOV];   /* Poradie pridavania bodov. */
int head=0;                     /* Toto je prvy bod t.j. najstarsi. */
int tail=0;                     /* Sem sa bude ukladat najnovsi bod. */
struct Uzol *koren;             /* Koren stromu. */
/* Prepocita novy obal pre zaveseneho syna. Vazeny priemer centier deti.*/
static void VypocitajObal(struct Obal *obal) {
  struct Obal *ob;
  struct Uzol *uzol;
  int i,j,sumw;
  unsigned int w[DETI_MAX];
  double *x[DETI_MAX],*r[DETI_MAX],*c,max,vzd,sumx;
  uzol=obal->syn;
  c=obal->sfera.x;
  if(uzol->uroven > 0) {
    for(i=0,sumw=0; i<uzol->pocet; i++) {
      ob=uzol->obsah.obal[i];
      sumw+=w[i]=ob->pocet_bodov;
      x[i]=ob->sfera.x; r[i]=&(ob->sfera.r);
    }
    for(j=0;j<DIMENZIA;j++) {
      for(i=0,sumx=0; i<uzol->pocet; i++) sumx+=x[i][j]*w[i];
      c[j]=sumx/sumw;
    }
    for(i=0,max=0; i<uzol->pocet; i++) {
      vzd=vzdialenost(c,x[i]) + *(r[i]);
      if(i==0 || vzd > max) max=vzd;
    }
  }
  else {
    sumw=uzol->pocet;
    for(i=0; i<uzol->pocet; i++) x[i]=uzol->obsah.bod[i]->x;
    for(j=0;j<DIMENZIA;j++) {
      for(i=0,sumx=0; i<uzol->pocet; i++) sumx+=x[i][j];
      c[j]=sumx/sumw;
    }
    for(i=0,max=0; i<uzol->pocet; i++) {
      vzd=vzdialenost(c,x[i]);
      if(i==0 || vzd > max) max=vzd;
    }
  }
  obal->sfera.r=max;
  obal->pocet_bodov=sumw;
}
/* Rovnomerne rozdeli obsah uzlov. Musia by rovnakej urovne. */
static void RozdelObsah(struct Uzol *uz1, struct Uzol *uz2) {
  void *pom[2*DETI_MAX];
  struct Obal *ob,**obal;
  struct Bod  *bo,**bod;
  double priem[DIMENZIA],disp[DIMENZIA];
  int i,j,poc,max;
  obal=(struct Obal **)pom;
  bod=(struct Bod **)pom;
```

```
  if(uz1->uroven > 0) {
    for(i=0;i<uz1->pocet;i++) obal[i]=uz1->obsah.obal[i];
    poc=i;
    for(i=0;i<uz2->pocet;i++,poc++) obal[poc]=uz2->obsah.obal[i];
    for(j=0;j<DIMENZIA;j++) {     /* Vypocet priemeru po suradniciach. */
      for(i=0;i<poc;i++) priem[j]+=obal[i]->sfera.x[j];
      priem[j]/=poc;
    }
    for(j=0;j<DIMENZIA;j++) {     /* Vypocet disperzie po suradniciach. */
      for(i=0;i<poc;i++) disp[j]+=moc(priem[j] - obal[i]->sfera.x[j]);
      disp[j]/=poc;
      if(j==0 || disp[j]>disp[max]) max=j;
    }                             /* Maximalna disperzia urci suradnicu. */
    i=0,j=poc-1;
    while(i<j) {
      while(obal[i]->sfera.x[max]<priem[max] && i<j) i++;
      while(obal[j]->sfera.x[max]>priem[max] && i<j) j--;
      ob=obal[i]; obal[i]=obal[j]; obal[j]=ob; i++; j--;
    }
    uz1->pocet=poc/2; uz2->pocet=poc-uz1->pocet;
    for(i=0;i<uz1->pocet;i++) {
      uz1->obsah.obal[i]=obal[i];
      obal[i]->syn->otec=uz1;
      obal[i]->syn->ktory=i;
    }
    j=i;
    for(i=0;i<uz2->pocet;i++,j++) {
      uz2->obsah.obal[i]=obal[j];
      obal[j]->syn->otec=uz2;
      obal[j]->syn->ktory=i;
    }
  }
  else {
    for(i=0;i<uz1->pocet;i++) bod[i]=uz1->obsah.bod[i];
    poc=i;
    for(i=0;i<uz2->pocet;i++,poc++) bod[poc]=uz2->obsah.bod[i];
    for(j=0;j<DIMENZIA;j++) {     /* Vypocet priemeru po suradniciach. */
      for(i=0;i<poc;i++) priem[j]+=bod[i]->x[j];
      priem[j]/=poc;
    }
    for(j=0;j<DIMENZIA;j++) {     /* Vypocet disperzie po suradniciach. */
      for(i=0;i<poc;i++) disp[j]+=moc(priem[j] - bod[i]->x[j]);
      disp[j]/=poc;
      if(j==0 || disp[j]>disp[max]) max=j;
    }                             /* Maximalna disperzia urci suradnicu. */
    i=0,j=poc-1;
    while(i<j) {
      while(bod[i]->x[max]<priem[max] && i<j) i++;
      while(bod[j]->x[max]>priem[max] && i<j) j--;
      bo=bod[i]; bod[i]=bod[j]; bod[j]=bo; i++; j--;
    }
    uz1->pocet=poc/2; uz2->pocet=poc-uz1->pocet;
    for(i=0;i<uz1->pocet;i++) {
      uz1->obsah.bod[i]=bod[i];
      bod[i]->uzol=uz1;
      bod[i]->ktory=i;
    }
    j=i;
    for(i=0;i<uz2->pocet;i++,j++) {
      uz2->obsah.bod[i]=bod[j];
      bod[j]->uzol=uz2;
      bod[j]->ktory=i;
    }
  }
}
/* Dolozi dany element do uzla. */
static void Insert1(struct Uzol *uzol, void *co) {
  if(uzol->uroven==0) {
    uzol->obsah.bod[uzol->pocet]=(struct Bod*)co;
    ((struct Bod*)co)->ktory=uzol->pocet;
```

```
    ((struct Bod*)co)->uzol=uzol;
    ++uzol->pocet;
  }
  else {
    uzol->obsah.obal[uzol->pocet]=(struct Obal*)co;
    ((struct Obal*)co)->syn->otec=uzol;
    ((struct Obal*)co)->syn->ktory=uzol->pocet;
    ++uzol->pocet;
  }
}
/* Rozdvoji uzol v strome. A dalej to propaguje vyssie. */
static void Insert(struct Uzol *uzol, void *co) {
  struct Obal *obal;
  struct Uzol *novyuzol;
  struct Obal *novyobal;
  Insert1(uzol,co);
  while(uzol->pocet == DETI_MAX) {
    if(uzol->otec == NULL) {
      koren=novyuzol=(struct Uzol *)EmMalloc(sizeof(struct Uzol));
      novyobal=(struct Obal *)EmMalloc(sizeof(struct Obal));
      novyobal->syn= uzol;
      novyuzol->obsah.obal[0]=novyobal;
      uzol->otec=novyuzol;
      uzol->ktory=0;
      novyuzol->uroven=uzol->uroven+1;
      novyuzol->pocet=1;
      novyuzol->otec=NULL;
    }
    obal=uzol->otec->obsah.obal[(uzol->ktory)];
    novyuzol=(struct Uzol *)EmMalloc(sizeof(struct Uzol));
    novyobal=(struct Obal *)EmMalloc(sizeof(struct Obal));
    novyuzol->uroven=uzol->uroven;
    novyuzol->pocet=0;
    novyuzol->otec=NULL;
    novyobal->pocet_bodov=0;
    novyobal->syn=novyuzol;
    RozdelObsah(uzol,novyuzol);   /* Rozdeli deti a aktualizuje pocet. */
    VypocitajObal(obal);          /* Aktualizuje obal. */
    VypocitajObal(novyobal);      /* Aktualizuje obal. */
    uzol=uzol->otec;
    Insert1(uzol,novyobal);
  }
  while(uzol->otec) {
    VypocitajObal(uzol->otec->obsah.obal[uzol->ktory]);
    uzol=uzol->otec;
  }
}
static void InsertBod(struct Bod *bod) {
  struct Uzol *uz;
  struct Obal *minobal,*obal;
  double mind,d;
  int i;
  uz=koren;                    /* Najdeme poziciu, kam vsunieme novy bod. */
  while(uz->uroven > 0) {       /* Jeden prvok musi urcite existovat. Akoby
                                   mohli existovat listy uroven==0 :-) */
    minobal=uz->obsah.obal[0];
    mind=vzdialenost(bod->x,minobal->sfera.x);
    for(i=1;i<uz->pocet;i++) {
      obal=uz->obsah.obal[i];
      d=vzdialenost(bod->x,obal->sfera.x);
      if(mind > d) {
        minobal=obal;
        mind=d;
      }
    }
    uz=minobal->syn;
  }
  Insert(uz,bod);               /* Bod vlozime a otestujeme preplnenost. */
}
static void InsertObal(struct Obal *ob) {
```

```
  struct Uzol *uz;
  struct Obal *minobal,*obal;
  double mind,d;
  int i;
  uz=koren;                     /* Najdeme poziciu, kam vsunieme obal. */
  while(uz->uroven > (ob->syn->uroven)+1) {
                                /* Jeden element musi urcite existovat. */
                                /* Kedze uroven > 0 existuju listy. :-) */
    minobal=uz->obsah.obal[0];
    mind=vzdialenost(ob->sfera.x,minobal->sfera.x);
    for(i=1;i<uz->pocet;i++) {
      obal=uz->obsah.obal[i];
      d=vzdialenost(ob->sfera.x,obal->sfera.x);
      if(mind > d) {
        minobal=obal;
        mind=d;
      }
    }
    uz=minobal->syn;
  }
  Insert(uz,ob);               /* Obal vlozime a otestujeme preplnenost. */
}
/* Vyhodi dany element z uzla. */
static void Delete1(struct Uzol *uzol, int poz) {
  struct Bod *pb;
  struct Obal *po;
  int k;
  if(poz==(k=--uzol->pocet)) k=0;      /* Vyhadzujeme posledny v rade? */
  if(uzol->uroven==0) {
    uzol->obsah.bod[poz]->uzol=NULL;
    if(k>0) {
      uzol->obsah.bod[poz]=pb=uzol->obsah.bod[k];
      pb->ktory=poz;
      uzol->obsah.bod[k]=NULL;
    }
    else uzol->obsah.bod[poz]=NULL;
  }
  else {
    uzol->obsah.obal[poz]->syn->otec=NULL;
    if(k>0) {
      uzol->obsah.obal[poz]=po=uzol->obsah.obal[k];
      po->syn->ktory=poz;
      uzol->obsah.obal[k]=NULL;
    }
    else uzol->obsah.obal[poz]=NULL;
  }
}
/* Vymazali sme bod alebo obal a treba obnovit konzistentnost stromu. */
static void DeleteBod(struct Bod *bod) {
  struct Uzol *otec=NULL,*uzol;
  struct Obal *obal;
  struct Bod *pb;
  int i;
  uzol=bod->uzol;
  Delete1(bod->uzol,bod->ktory);
  while(uzol->otec && uzol->pocet < DETI_MIN) {
    otec=uzol->otec;
    obal=otec->obsah.obal[uzol->ktory];
    Delete1(otec, uzol->ktory);
    EmFree(obal);
    if(otec->pocet==0) {   /* Toto moze byt iba koren. Treba znizit uroven. */
                           /* DETI_MIN > 1 t.j. nemoze taky existovat. */
      EmFree(otec);
      koren=uzol;
    }
    else {
      if(uzol->uroven==0)
        for(i=0; i<uzol->pocet; i++) InsertBod(uzol->obsah.bod[i]);
      else
        for(i=0; i<uzol->pocet; i++) InsertObal(uzol->obsah.obal[i]);
```

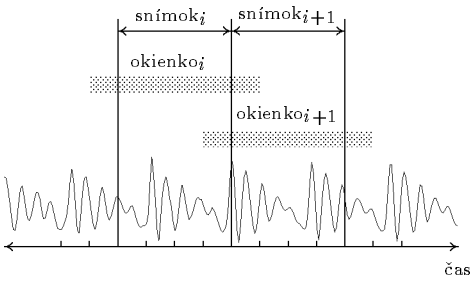
```
      EmFree(uzol);
      uzol=otec;
    }
  }
  while(uzol->otec) {
    VypocitajObal(uzol->otec->obsah.obal[uzol->ktory]);
    uzol=uzol->otec;
  }
}
void TreeInsert( double *x, int oblast ) {
  struct Uzol *uz;
  int i;
  if(head == (tail+1) % POCET_BODOV) {
    DeleteBod(&(body[head]));
    head=(head+1) % POCET_BODOV;
  }
  memcpy(body[tail].x,x,DIMENZIA*sizeof(double));
  body[tail].oblast=oblast;
  InsertBod( &(body[tail]));
  tail=(tail+1) % POCET_BODOV;
}
Queue adepts,knn;
int ltouch;
void TreeSearch(double *x, Queue *knn, double dist) {
  double max_dist,min_dist;
  Queue_elem elem,elem1;
  struct Uzol *uz;
  int i;
  ltouch=0;/*test*/
  adepts.pocet=0;                /* Vynulujeme bufer adeptov. */
  knn->pocet=0;                  /* Vynulujeme bufer susedov. */
  max_dist=dist;
  min_dist=0;
  uz=koren;
  while(min_dist<=max_dist) {
    if(uz->uroven) {             /* Deti su obaly. */
      for(i=0;i<uz->pocet;i++) {
        elem.key=vzdialenost(x,uz->obsah.obal[i]->sfera.x)
                 - uz->obsah.obal[i]->sfera.r;
        if(elem.key<0) elem.key=0;
        if(elem.key<max_dist) {
          elem.data=uz->obsah.obal[i]->syn;
          Queue_MinInsert(&adepts,&elem);
        }
      }
    }
    else {                       /* Deti su body. */
      for(i=0;i<uz->pocet;i++) {
        elem.key=vzdialenost(x,uz->obsah.bod[i]->x);
        if(elem.key<max_dist) {
          elem.data=uz->obsah.bod[i];
          if(knn->pocet==knn->size) Queue_MaxDeltop(knn,&elem1);
          Queue_MaxInsert(knn,&elem);
          if(knn->pocet==knn->size && knn->queue[1].key < max_dist)
            max_dist=knn->queue[1].key;
        }
      }
      ltouch+=uz->pocet;/*test*/
    }
    if(adepts.pocet==0) break;   /* Rad je prazdny. */
    Queue_MinDeltop(&adepts,&elem);
    min_dist=elem.key;
    uz=(struct Uzol *)elem.data;
  }
}
void TreeSearchN(double *x, Queue *knn, double dist) {
 int i;
 double max_dist,min_dist;
 Queue_elem elem,elem1;
 knn->pocet=0;
```

```
 max_dist=dist;
 i=head;
 while(i!=tail) {
   elem.key=vzdialenost(x,body[i].x);
   if(elem.key<max_dist) {
     elem.data=&body[i];
     if(knn->pocet==knn->size) Queue_MaxDeltop(knn,&elem1);
     Queue_MaxInsert(knn,&elem);
     if(knn->pocet==knn->size && knn->queue[1].key < max_dist)
       max_dist=knn->queue[1].key;
   }
   i=(i+1) % POCET_BODOV;
 }
}
.
.
.
```

# 5. Techniky spracovania signálu

## 5.1. Spracovanie signálu

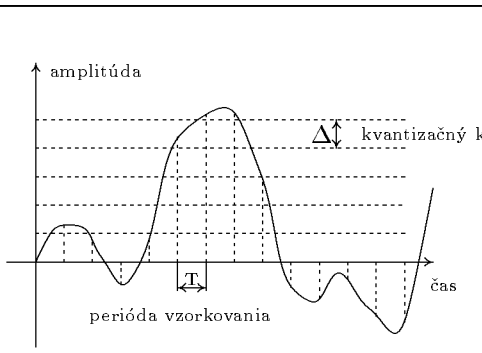
Signál z mikrofónu sa najprv filtruje analógovým filtrom, potom sa A/D prevodníkom prevedie na číselný (digitálny) signál, ktorý je ešte upravený digitálnym filtrom. V tejto fáze je potrebné odfiltrovať šum elektrickej siete $50 - 60Hz$, napájacieho zdroja, .... Rovnako treba odfitrovať frekvencie, ktorých veľkosť je väčšia ako polovica vzorkovacej frekvencie.




**Obr.34** Vzťah okienka a snímku

## 5.2. Kódovanie tvaru vlny

Spojitý signál zvukovej vlny je potrebné previesť do číselnej podoby. Najčastejšie sa používa PCM (pulzná kódová modulácia). Princíp vidieť na obr.35. Základnými údajmi sú **vzorkovacia frekvencia** $F_v$ a **kvantizačný krok** $\Delta$. Pri vzorkovacej frekvencii je treba dodržiavať Shannonovu vzorkovaciu teorému. T.j. keď je analógový signál zhora ohraničený (filtrom) frekvenciou $F_m$, tak pre vzorkovaciu frekvenciu musí platiť $F_v \geq 2F_m$. Pri nedodržaní dochádza ku skresleniu signálu.




**Obr.35**

Kvantizácia je vhodné priblíženie číselnej hodnoty z konečnej množiny hodnôt k analógovej hodnote signálu. Realizuje sa A/D prevodníkom. Dôležitý je počet úrovní kvantovania (na obr.35 prerušované vodorovné čiary) a kvantizačný krok. Počet úrovní sa volí v tvare $2^B$. Ak pre signál platí $|s(kT)| \leq S_{max}$,

potom $2S_{max} = \Delta 2^B$. Na obr.35 tiež vidieť, že pri zaokrúhľovaní na najbližšiu kvantizačnú úroveň dochádza ku **kvantizačnému skresleniu**.

Okrem PCM sa používajú i logPCM (logaritmické rozloženie kvantizačných úrovní), DPCM (kóduje sa iba diferencia medzi nasledujúcimi hodnotami), DM (delta modulácia), ADPCM (adaptívna DPCM), ADM (adaptívna DM), ... Tieto metódy sa však využívajú skôr pre prenos reči s menším prenosom informácií ako pre účely rozpoznávania.

## 5.3. Funkcia okienka

Niekedy je potrebné vstupné vzorky signálu ováhovať

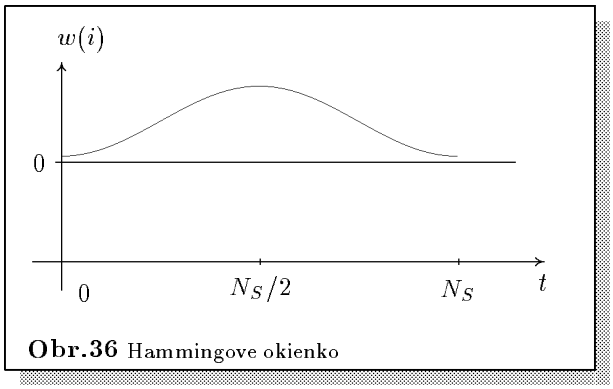
$$s_W(n+k) = s(n+k).w(N_S - k), \qquad 0 \leq k < N_S,$$

kde $n$ udáva pozíciu okienka v čase. Pre jednoduchosť sa veľmi často používa **pravouhlé** okienko t.j. žiadne.

$$w(k) = \begin{cases} 1 & \text{pre} \quad 0 \leq k < N_S, \\ 0 & \text{pre} \quad k \geq N_S. \end{cases}$$

Pravouhlé okienko má nepriaznivý vplyv na hraniciach, kedy dochádza ku skoku. Preto sa zavádza **Hammingovo** okienko (obr.36).

$$w(k) = \begin{cases} 0,54 - 0,46\cos\left(\frac{2\pi n}{N_S - 1}\right) & \text{pre} \quad 0 \leq k < N_S, \\ 0 & \text{pre} \quad k \geq N_S. \end{cases}$$




**Obr.36** Hammingove okienko

Jeho prechod ku hraniciam je plynulý, čo sa prejaví i na použitých výpočtoch.

## 5.4. Krátkodobá energia

- **krátkodobá energia**

$$E_n = \sum_{k=-\infty}^{\infty} \left(s(k).w(n-k)\right)^2,$$

kde $s(k)$ je vzorka signálu a $w(n)$ je príslušný typ okienka. Nedostatkom tejto charakteristiky signálu je veľká citlivosť na veľké zmeny urovne signálu.

- **krátkodobá intenzita**

$$M_n = \sum_{k=-\infty}^{\infty} |s(k)|.w(n-k).$$

Vhodné pre oddeľovanie segmentov ticha a segmentov reči. Prípadne oddeľovanie znelých a neznelých segmentov reči.

## 5.5. Krátkodobá funkcia stredného počtu prechodov signálu nulou

$$Z_n = \sum_{k=-\infty}^{\infty} |sgn(s(k)) - sgn(s(k-1))| w(n-k),$$

kde

$$sgn(s(k)) = \begin{cases} 1, & \text{pre } s(k) \geq 0, \\ -1, & \text{pre } s(k) < 0. \end{cases}$$

Vhodné pre určovanie hraníc zašumeného slova. Motiváciou je fakt, že pri sínusovom priebehu signálu o frekvencii $f$ je priemerný počet priechodov nulou $2f$. Pre zašumený úsek sa táto hodnota zvyšuje.

## 5.6. Krátkodobá autokorelačná funkcia

je definovaná nasledovne:

$$R_n(m) = \sum_{k=-\infty}^{\infty} s(k).w(n-k).s(k+m).w(n-k-m),$$

kde $w(n)$ je funkcia okienka. Ak má signál periódu $P$, potom $R_n(m)$ nadobúda maximá pre $m = 0, P, 2P, \ldots$.

Autokorelačná funkcia je vhodná z hľadiska indikácie periodicity signálu. Používa sa na výpočet periódy základného hlasivkového tónu. Za týmto účelom musí byť okienko dostatočne dlhé, aby obsahovalo aspoň dve periódy signálu.

## 5.7. Z-transformácie

Z-transformácia diskrétneho signálu $x(n) \in \mathbb{R}$ je:

$$Z\{x(n)\} = X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}, \quad z \in \mathbb{C}$$

Zväčša sa používa jednostranná z-tranformácia:

$$Z\{x(n)\} = X(z) = \sum_{n=0}^{\infty} x(n) z^{-n}, \quad z \in \mathbb{C}$$

Pre jednotkový impulz $x_0 = 1, x_1 = 0, \ldots, x_n = 0$ platí

$$X(z) = 1z^0 = 1.$$

Pre trvajúci impulz t.j. $x_i = 1, \quad i = \overline{0,n}$ je

$$X(z) = \sum_{n=0}^{\infty} z^{-n} = \frac{(z^{-1})^{n+1} - 1}{z^{-1} - 1},$$

ak ďalej predpokladáme, že $|z^{-1}| < 1$, potom

$$X(z) = \sum_{n=0}^{\infty} z^{-n} = \frac{1}{1 - z^{-1}}, \quad |z^{-1}| < 1$$

Ďalšia zaujímavá vlastnosť ak $x(n) = a^n u(n)$ potom

$$X(z) = \sum_{n=0}^{\infty} a^n z^{-n}$$

$$X(z) = \sum_{n=0}^{\infty} a^n z^{-n} = \frac{1}{1 - az^{-1}}, \quad |z^{-1}| < 1$$

Stručná tabuľka niektorých z-transformácií:

| $x(n)$ | $X(z)$ |
|---|---|
| $\delta(n)$ | $1$ |
| $a^n u(n)$ | $\frac{1}{1-az^{-1}}$ |
| $a^n \cos(n\theta u(n))$ | $\frac{1-az^{-1}\cos(\theta)}{1-2az^{-1}\cos(\theta)+a^2z^{-2}}$ |
| $a^n \sin(n\theta u(n))$ | $\frac{az^{-1}\sin(\theta)}{1-2az^{-1}\cos(\theta)+a^2z^{-2}}$ |
| $na^n u(n)$ | $\frac{az^{-1}}{(1-az^{-1})^2}$ |
| $n(n-1)a^n u(n)$ | $\frac{2a^2z^{-2}}{(1-az^{-1})^3}$ |
| $a_1x_1(n) + a_2x_2(n)$ | $a_1X_1(z) + a_2X_2(z)$ |
| $x(n-m)$ | $z^{-m}X(z)$ |
| $x_1(n) * x_2(n)$ | $X_1(z)X_2(z)$ |
| $a^n x(n)$ | $X(a^{-1}z)$ |
| $nx(n)$ | $\frac{z^{-1}\delta X(z)}{\delta z^{-1}}$ |

Ak sa nulové body (korene) nachádzajú

- vo vnútri jednotkovej kružnice, hovoríme o signále (transformácii, filtri) s **minimálnou fázou**. (Napríklad $X_1(t)$ v ďalšom texte.) V tomto prípade existuje inverzné zobrazenie. $(X(z)X^{-1}(z) = 1)$
- mimo jednotkovej kružnice, hovoríme o signále (transformácii, filtri) s **maximálnou fázou**. (Napríklad $X_2(t)$ v časti o filtroch.)

## 5.8. Vlastnosti Z-transformácií

- linearita

$$Z\{a_1x_1(n) + a_2x_2(n)\} = \sum_{n=0}^{\infty}(a_1x_1(n) + a_2x_2(n))z^{-n} =$$

$$a_1\sum_{n=0}^{\infty} x_1(n)z^{-n} + a_2\sum_{n=0}^{\infty} x_2(n)z^{-n} = a_1Z\{x_1(n)\} + a_2Z\{x_2(n)\}$$

- posuv

$$Z\{x(n-m)\} = \sum_{n=0}^{\infty} x(n-m)z^{-n} = \sum_{n=-m}^{\infty} x(n)z^{-(n+m)} = z^{-m}\sum_{n=-m}^{\infty} x(n)z^{-n}$$

- konvolúcia
  Platí vzťah (Cauchy):

$$\left(\sum_{n=0}^{\infty} f(n)z^{-n}\right)\left(\sum_{n=0}^{\infty} g(n)z^{-n}\right) = \sum_{n=0}^{\infty}\left(\sum_{k=0}^{n} f(k)g(n-k)\right)z^{-n} = \sum_{n=0}^{\infty}\left(\sum_{k=0}^{n} f(n-k)g(k)\right)z^{-n}$$

Z toho

$$Z\{f(n) * g(n)\} = \sum_{n=0}^{\infty}\left(\sum_{k=0}^{n} f(n-k)g(k)\right)z^{-n} = \left(\sum_{n=0}^{\infty} f(n)z^{-n}\right)\left(\sum_{n=0}^{\infty} g(n)z^{-n}\right) = Z\{f(n)\}Z\{g(n)\}$$

$$Z\{a^n x(n)\} = \sum_{n=0}^{\infty} a^n x(n) z^{-n} = \sum_{n=0}^{\infty} x(n) \left(a^{-1} z\right)^{-n} = X(a^{-1} z)$$

- derivácia

$$z^{-1} \frac{\partial X}{\partial z^{-1}} = z^{-1} \sum_{n=0}^{\infty} n x(n) \left(z^{-1}\right)^{n-1} = \sum_{n=0}^{\infty} n x(n) z^{-n} = Z\{n x(n)\}$$

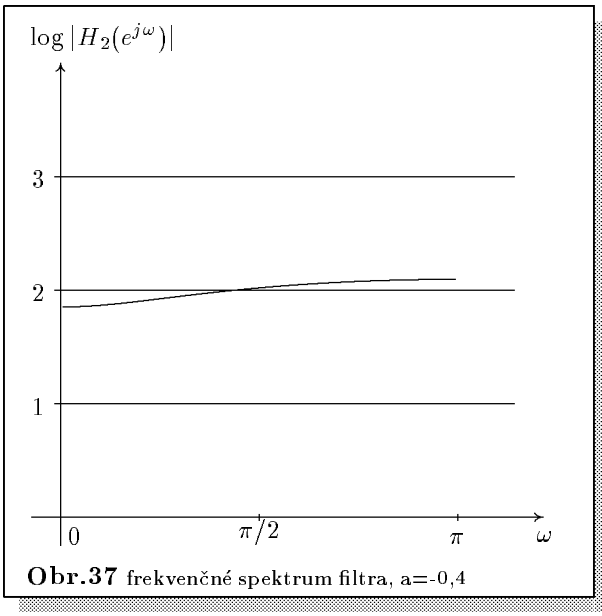
## 5.9. Filtre

Účelom filtrov je upraviť frekvenčné spektrum signálu podľa istých kritérií (váh). Možno sa na to dívať ako aplikácia okienkovej funkcie na spektrum. Pomocou z-transformácií úpravu zapisujeme nasledovne:

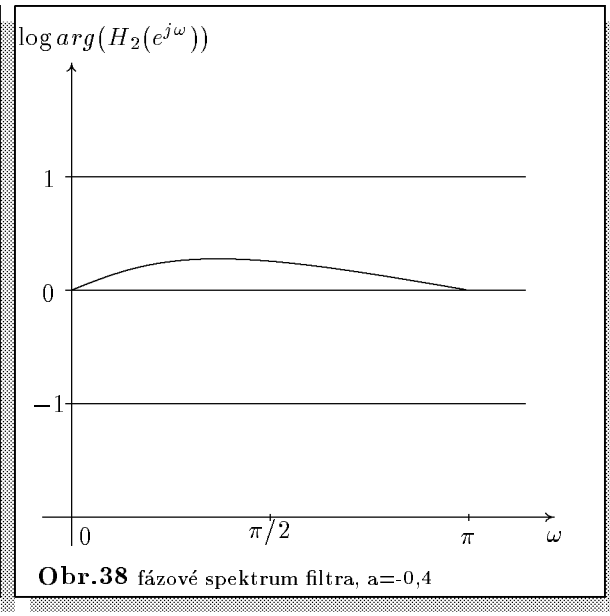
$$S_H(z) = S(z).H_i(z),$$

kde $S(z)$ je z-transformácia signálu, $H_i(z)$ je z-transformácia filtra a $S_H(z)$ je upravený signál (jeho z-transformácia). Najrozšírenejší filter je

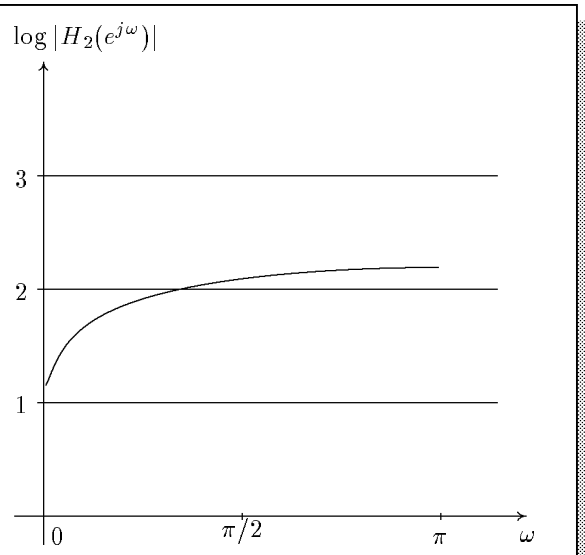
$$H_2(z) = 1 + a.z^{-1}, \qquad a \in \mathbb{R}$$



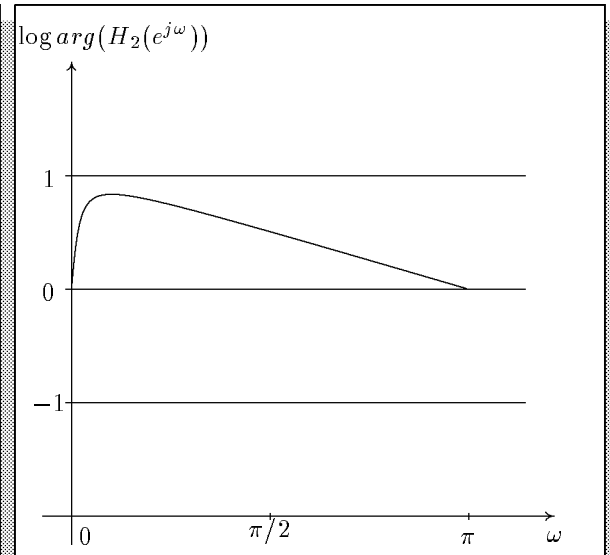

**Obr.37** frekvenčné spektrum filtra, a=-0,4



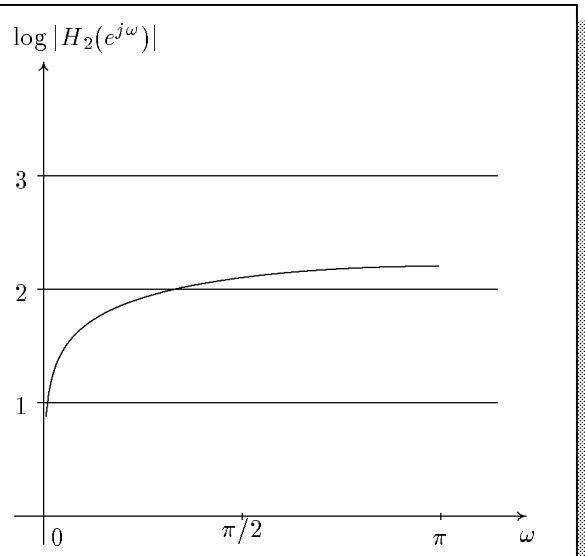

**Obr.38** fázové spektrum filtra, a=-0,4



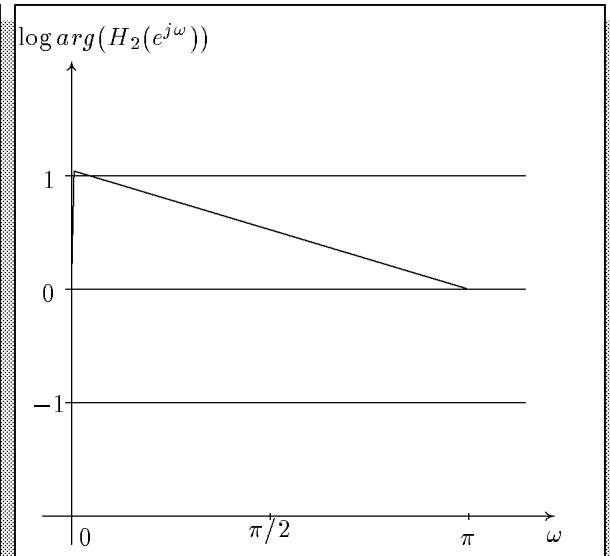

**Obr.39** frekvenčné spektrum filtra, a=-0,95




**Obr.40** fázové spektrum filtra, a=-0,95




**Obr.41** frekvenčné spektrum filtra, a=-1,0




**Obr.42** fázové spektrum filtra, a=-1,0

Pritom treba mať na pamäti okrem zmeny amplitúdy frekvenčného spektra i fázové posuny.

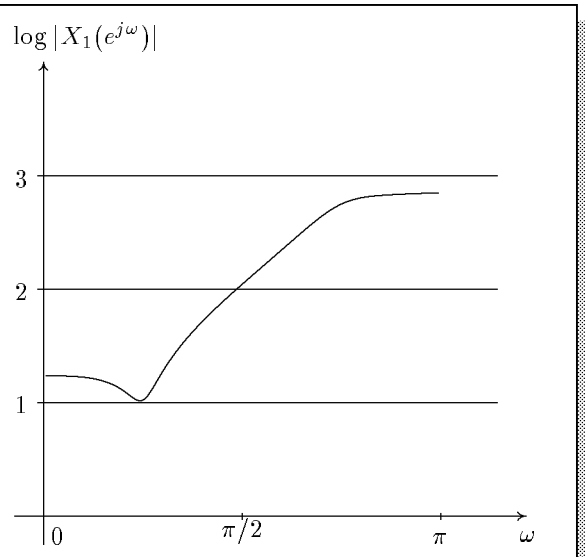
$$X_1(z) = \frac{(1-\varepsilon_1 z^{-1})(1-\varepsilon_1^* z^{-1})(1-\varepsilon_2 z^{-1})}{(1-\psi_1 z^{-1})(1-\psi_1^* z^{-1})(1-\psi_2 z^{-1})},$$

$$X_2(z) = \frac{(z^{-1}-\varepsilon_1)(z^{-1}-\varepsilon_1^*)(z^{-1}-\varepsilon_2)}{(1-\psi_1 z^{-1})(1-\psi_1^* z^{-1})(1-\psi_2 z^{-1})},$$

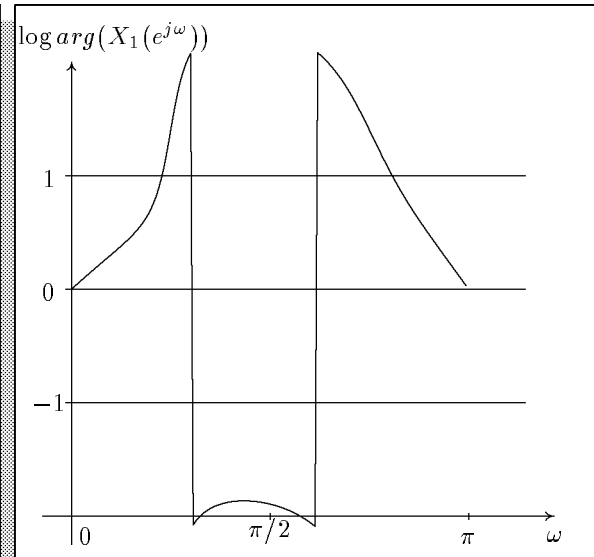
kde $\varepsilon_1 = 0.9 < 45^*, \varepsilon_2 = 0.5, \psi_1 = 0.7 < 135^*, \psi_2 = -0.5$ a $z \in \mathbb{C}$

$$|X_1(\omega)| = |X_2(\omega)|$$

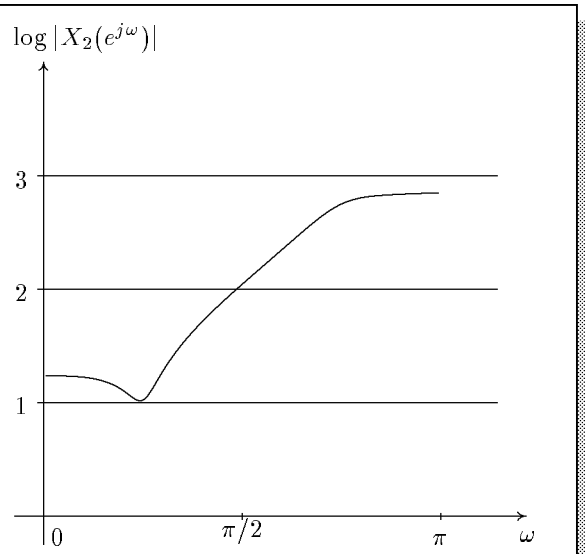
Transformácie $X_1(z)$ a $X_2(z)$ majú rovnaké frekvenčné spektrum, ale rozličné fázové spektrum. Rozdiel je aj v časovom vývoji ($X(z) = X(\omega t)$).



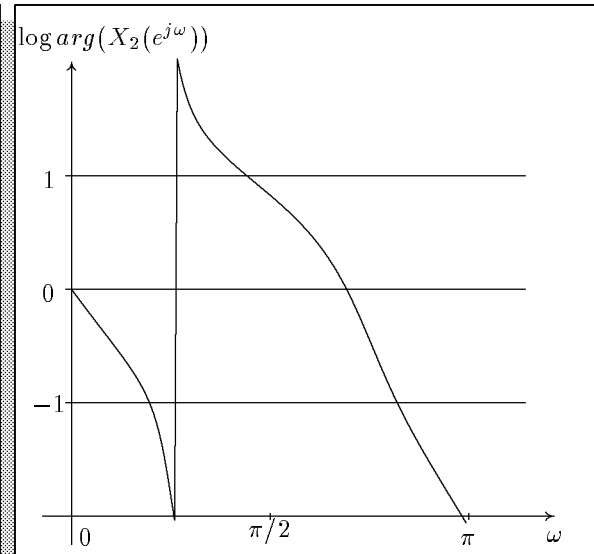

**Obr.43** frekvenčné spektrum filtra




**Obr.44** fázové spektrum filtra




**Obr.45** frekvenčné spektrum filtra




**Obr.46** fázové spektrum filtra

## 5.10. Diskrétna Furierova transformácia (DFT)

Nech $x_0, \ldots, x_{N-1}$ je konečná postupnosť hodnôt signálu, ktorý sa cyklí t.j. po hodnote $x_{N-1}$ nasleduje opäť hodnota $x_0$.

Diskrétna Furierova transformácia postupnosti $x_0, \ldots, x_{N-1}$ je nasledovná:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x(n) e^{-j\omega n}$$

Inverzná transformácia je nasledovná:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn} = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\omega k}$$

FT je vlastne špeciálnym prípadom z-transformácie pre $z = e^{j\omega}$.

## 5.11. Vlastnosti DFT

- linearita

  Nech $X(k)$ je DFT postupnosti $x(n)$ a $Y(k)$ je DFT postupnosti $y(n)$, potom DFT postupnosti $ax(n) + by(n)$ je $aX(k) + bY(k)$, kde $a$, $b$ sú ľubovoľné konštanty.
- posuv

  Nech $X(k)$ je DFT postupnosti $x(n)$, a nech $y(n)$ vznikne posunom postupnosti $x(n)$ o $n_0$ vzorkov. Potom DFT $y(n)$ je $Y(k) = X(k)e^{j\frac{2\pi}{N}kn_0}$. Podobne to platí i pre inverznú IDFT: Nech $X(k)$, $Y(k)$ sú DFT postupností $x(n)$ a $y(n)$ a naviac platí $Y(k) = X(k - k_0)$, potom $y(n) = x(n)e^{-j\frac{2\pi}{N}k_0 n}$
- kruhová konvolúcia

  Nech

$$u(n) = \sum_{i=0}^{N-1} x(i)y(n-i), \qquad n = 0, \ldots, N-1$$

potom

$$DFT\{u(n)\} = U(k) = X(k)Y(k)$$

Ďalej nech

$$u(n) = x(n)y(n)$$

potom

$$U(k) = \frac{1}{N}\sum_{i=0}^{N-1} X(i)Y(k-i), \qquad k = 0, \ldots, N-1$$

# 6. Lineárna prediktívna analýza

## 6.1. Lineárne prediktívne (LP) koeficienty

Táto metóda sa datuje už od 70 rokov. Je to časovo nenáročná a obľúbená metóda či už pre rozpoznávanie, kompresiu a verifikáciu. Taktiež je použiteľná i pre ekonómiu, fyziku, ...

Princíp spočíva na odhade $s(n)$ vzorku signálu pomocou lineárnej kombinácie $N_{LP}$ predchádzajúcich vzoriek.

$$s(n) = -\sum_{i=1}^{N_{LP}} a_{LP}(i)s(n-i) + e(n) \tag{1}$$

kde $e(n)$ je chyba predikcie (rozdiel medzi predikovanou a nameranou hodnotou) a $N_{LP}$ je rád modelu. Postupnosť $\{a_{LP}\}$ je definovaná ako **lineárne prediktívne koeficienty**. O čo menšia chyba, o to lepší model signálu. Neskôr sa ukáže ako LP model modeluje vyhladené spektrum signálu. Použitím z-transformácie možno (1) zapísať nasledovne

$$E(z) = H_{LP}(z).S(z),$$

kde $E(z)$ a $S(z)$ sú z-transformácie chybového a rečového signálu. $H_{LP}(z)$ je LP inverzný filter v tvare

$$H_{LP}(z) = 1 + \sum_{i=1}^{N_{LP}} a_{LP}(i)z^{-i}, \qquad \text{alebo}$$

$$H_{LP}(z) = \sum_{i=0}^{N_{LP}} a_{LP}(i)z^{-i}, \qquad \text{kde} \quad a_{LP}(0) = 1.$$

Krátkodobú energiu chybovej funkcie $e(n)$, ktorá je vyjadrená nasledovne

$$E = \sum_n e(n)^2 = \sum_n \left( s(n) + \sum_{i=1}^{N_{LP}} a_{LP}(i).s(n-i) \right)^2,$$

sa snažíme minimalizovať t.j.

$$\frac{\partial E}{\partial a_{LP}(j)} = 0, \qquad \text{pre} \quad 1 \le j \le N_{LP} \tag{2}$$

Zanietený čitateľ si isto všimne súvis s učením sa adaptívneho lineárneho neurónu, kedy však váhy upravujeme podľa znamienka derivácie postupne v čase.
Úpravou (2) dostaneme

$$\sum_{i=1}^{N_{LP}} a_{LP}(i) \sum_n s(n-j).s(n-i) = -\sum_n s(n-j).s(n), \qquad 1 \le j \le N_{LP},$$

a ďalej položme

$$\phi(j,i) = \sum_n s(n-j).s(n-i), \tag{3}$$

t.j.

$$\sum_{i=1}^{N_{LP}} a_{LP}(i).\phi(j,i) = -\phi(j,0), \quad 1 \le j \le N_{LP}. \tag{4}$$

Ďalšie zjednodušenie spočíva v ohraničení priebehu nekonečnej sumy v (3)

$$\phi_n(j,k) = \sum_{m=0}^{N_S-1} s(n-j+m).s(n-k+m) \tag{5}$$

Vzťah (4) možno prepísať do reči matíc

$$\begin{pmatrix} \phi_n(1,1) & \phi_n(1,2) & \dots & \phi_n(1,N_{LP}) \\ \phi_n(2,1) & \phi_n(2,2) & \dots & \phi_n(2,N_{LP}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_n(N_{LP},1) & \phi_n(N_{LP},2) & \dots & \phi_n(N_{LP},N_{LP}) \end{pmatrix} \begin{pmatrix} a_{LP}(1) \\ a_{LP}(2) \\ \vdots \\ a_{LP}(N_{LP}) \end{pmatrix} = \begin{pmatrix} -\phi_n(1,0) \\ -\phi_n(2,0) \\ \vdots \\ -\phi_n(N_{LP},0) \end{pmatrix} \tag{6}$$

a z toho vyjadriť koeficienty $\{a_{LP}\}$

$$\begin{pmatrix} a_{LP}(1) \\ a_{LP}(2) \\ \vdots \\ a_{LP}(N_{LP}) \end{pmatrix} = \begin{pmatrix} \phi_n(1,1) & \phi_n(1,2) & \dots & \phi_n(1,N_{LP}) \\ \phi_n(2,1) & \phi_n(2,2) & \dots & \phi_n(2,N_{LP}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_n(N_{LP},1) & \phi_n(N_{LP},2) & \dots & \phi_n(N_{LP},N_{LP}) \end{pmatrix}^{-1} \begin{pmatrix} -\phi_n(1,0) \\ -\phi_n(2,0) \\ \vdots \\ -\phi_n(N_{LP},0) \end{pmatrix} \tag{7}$$

Matica vystupujúca v (6) sa nazýva **kovariančná**. Ďalšia úprava spočíva v stanovení $s(n) = 0$ mimo daného snímku. V tomto prípade je potrebné si uvedomiť dôležitosť okienka, ktoré zabráni skoku k 0 na hraniciach snímku. Najčastejšie sa používa Hammingovo okienko. Upravujeme teda hranice v (3)

$$\phi_n(j,k) = \sum_{m=0}^{N_S-1-(j-k)} s(n+m).s(n+m+j-k) \tag{8}$$

Je zrejmé, že (8) je autokorelačná funkcia, pričom platí

$$\phi_n(j,k) = R_n(j-k),$$
$$R_n(k) = \sum_{m=0}^{N_S-1-k} s(n+m).s(n+m+k)$$

A keďže $R_n(k)$ je párna funkcia platí

$$\phi_n(j,k) = R_n(|j-k|),$$

a potom (4) môžeme prepísať

$$\sum_{i=1}^{N_{LP}} a_{LP}(i).R_n(|j-i|) = -R_n(|j|), \quad 1 \le j \le N_{LP}. \tag{9}$$

V reči matíc to vyzerá nasledovne

$$\begin{pmatrix} R_n(0) & R_n(1) & \dots & R_n(N_{LP}-1) \\ R_n(1) & R_n(0) & \dots & R_n(N_{LP}-2) \\ \dots & \dots & \ddots & \vdots \\ R_n(N_{LP}-1) & R_n(N_{LP}-2) & \dots & R_n(0) \end{pmatrix} \begin{pmatrix} a_{LP}(1) \\ a_{LP}(2) \\ \vdots \\ a_{LP}(N_{LP}) \end{pmatrix} = \begin{pmatrix} -R_n(1) \\ -R_n(0) \\ \vdots \\ -R_n(N_{LP}) \end{pmatrix}$$

Pre chybu $E$ po aplikovaní podobných predpokladov (signál mimo snímku je nulový) platí

$$\begin{aligned}
E &= \sum_n \left( s(n) + \sum_{i=1}^{N_{LP}} a_{LP}(i).s(n-i) \right)^2 \\
&= \sum_n s(n)^2 + \sum_n 2.s(n) \sum_{i=1}^{N_{LP}} a_{LP}(i).s(n-i) + \sum_n \sum_{i,j} a_{LP}(i).a_{LP}(j).s(n-i).s(n-j) \\
&= \phi(0,0) + 2 \sum_{i=1}^{N_{LP}} a_{LP}(i).\phi(i,0) + \sum_{i=1}^{N_{LP}} a_{LP}(i) \sum_{j=1}^{N_{LP}} a_{LP}(j)\phi(i,j) \\
E_n &= R_n(0) + 2 \sum_{i=1}^{N_{LP}} a_{LP}(i).R_n(i) - \sum_{i=1}^{N_{LP}} a_{LP}(i) R_n(i) \\
&= R_n(0) + \sum_{i=1}^{N_{LP}} a_{LP}(i).R_n(i)
\end{aligned}$$

Autokorelačná metóda vždy produkuje LP filter, ktorého nulové body ležia vo vnútri jednotkovej kružnice z-roviny (filter minimálnej fázy).

Výpočet koeficientov $\{a_{LP}\}$ autokorelačnou metódou (Levinson-Durbin), vyjadrený rekurzívne pre $i = 1, 2, \ldots, N_{LP}$

$$\begin{aligned}
E_n^{(0)} &= R_n(0), \\
k_i &= -\frac{\left( R_n(i) + \sum_{j=1}^{i-1} a_{LP}^{(i-1)}(j).R_n(i-j) \right)}{E_n^{(i-1)}}, \\
a_{LP}^{(i)}(i) &= k_i, \\
a_{LP}^{(i)}(j) &= a_{LP}^{(i-1)}(j) + k_i.a_{LP}^{(i-1)}(i-j), \qquad 1 \le j \le i-1, \\
E_n^{(i)} &= (1-k_i^2) E_n^{(i-1)}
\end{aligned}$$

Za predpokladu, že budiaca funkcia má tvar jednotkového impulzu (znelé segmenty reči) alebo bieleho šumu (neznelé segmenty reči), možno odvodiť vzťah pre výpočet zosilenia $G_{LP}$

$$G_{LP}{}^2 = R_n(0) + \sum_{i=1}^{N_{LP}} a_{LP}(i) R_n(i) = E_n = E_n^{(N_{LP})}$$

Pomocou koeficientov $\{a_{LP}\}$ je možné vypočítať spektrum signálu, ktoré má charakter vyhladenej obálky skutočného spektra

$$S_{LP}(z) = \frac{G_{LP}}{H_{LP}(z)}$$

$$S_{LP}(j\omega) = \frac{G_{LP}}{1 + \sum_{i=1}^{N_{LP}} a_{LP}(i).e^{-j\omega i}}$$

Za povšimnutie stojí zosilovací koeficient $G_{LP}$, ktorý prispôsobí LP spektrum k originálnemu spektru. Čo sa týka počtu $N_{LP}$ väčšinou sa volí pevne. Čím je rád vyšší, o to lepšie je modelovaný signál. Pozri obrázky obr.47 až obr.68, kde veľkosť okienka $N_S = 256$. Podobnosť obrázkov obr.57 a obr.58 rovnako i obr.67 a obr.68 súvisí už s veľkosťou okienka a požadovaným počtom koeficientov. Z iteračného výpočtu $N_{LP}$ je zrejmý monotónny pokles chyby so zväčšovaním rádu. Nízky rád produkuje iba hrubú spektrálnu obálku. Táto hrubá spektrálna obálka sa používala na odhad **formantov**. Formanty sú maximá takejto obálky. Pozri obr.50 a obr.60. Tento odhad je však niekedy komplikovaný, pretože tieto formanty sa

občas priblížia k sebe až sa spoja. Okrem toho formanty sú vhodné iba na odhad samohlások. Zväčša sa používajú prvé dva formanty. Experimentálne namerané rozloženie pozícií prvých dvoch formantov možno vidieť na obr.69, obr.70, obr.71 a obr.72. Na obr.69 vidieť jemnú štruktúru pre samohlásky. Tu bol použitý iba jeden prechod t.j. každá samohláska bola vyslovená iba raz. Viac prechodov samohláskami bolo použitých na obr.70. Nepoužiteľnosť pre celú abecedu demonštrujú obr.71 a obr.72. Diagonálna čiara vznikla v prípade, že spektrum obsahovalo iba jedno maximum.

Bližšie sa možno s problematikou fonológie slovenského jazyka zoznámiť v [14].

Správnym nastavením LP koeficientov je možné rekonštruovať zvukovú nahrávku rečového signálu. V histórii sa LP koeficienty používali priamo na rozpoznávanie.

## 6.2. Banka filtrov odvodená LP

Na výpočet amplitúdy jednotlivých fitrov sa používa LP spektrum, ktoré je stabilnejšie a robustnejšie pre odhady parametrov spracovania. Používa sa nasledovný vzťah:

$$S_{LP}(f) = \frac{G_{LP}}{\sum_{i=0}^{N_{LP}} a_{LP}(i) e^{-j2\pi(f/f_S).i}},$$

kde $f_S$ reprezentuje vzorkovaciu frekvenciu.

## 6.3. Kepstrálne koeficienty odvodené LP

Najprv zlogaritmujeme $S_{LP}(z)$

$$\log S_{LP}(z) = \log \frac{G_{LP}}{H_{LP}(z)},$$

kedže platí $\lim_{z\to\infty} H_{LP}(z) = 1$ a $H_{LP}(z)$ je polynóm v premennej $z^{-1}$, ktorý má korene vo vnútri jednotkovej kružnice, potom pomocou Taylorovho rozvoja vyjadríme

$$\log \frac{G_{LP}}{H_{LP}(z)} = c_{LP}(0) + c_{LP}(1)z^{-1} + c_{LP}(2)z^{-2} + \cdots = \sum_{k=0}^{\infty} c_{LP}(k).z^{-k}, \tag{10}$$

kde $\{c_{LP}\}$ sú **kepstrálne koeficienty**. Zderivujeme obidve strany rovnice (10)

$$-\sum_{k=1}^{\infty} k.c_{LP}(k).z^{-k-1} = \frac{\sum_{i=1}^{N_{LP}} i.a_{LP}(i).z^{-i-1}}{\sum_{i=0}^{N_{LP}} a_{LP}(i).z^{-i}},$$

ďalej upravíme

$$\left(\sum_{i=0}^{N_{LP}} a_{LP}(i).z^{-i}\right)\left(\sum_{k=1}^{\infty} k.c_{LP}(k).z^{-k}\right) = -\sum_{i=1}^{N_{LP}} i.a_{LP}(i).z^{-i}.$$

Porovnaním koeficientov pri rovnakých mocninách $z$ dostaneme

$$z^{-i}\sum_{j=1}^{i} a_{LP}(i-j).j.c_{LP}(j) = \begin{cases} -z^{-i}.i.a_{LP}(i), & \text{pre} \quad 1 \le i \le N_{LP} \\ 0, & \text{pre} \quad i > N_{LP} \end{cases}.$$

Z toho a z poznatku $a_{LP}(0) = 1$ odvodíme

$$i.c_{LP}(i) + \sum_{j=1}^{i-1} a_{LP}(i-j).j.c_{LP}(j) = \begin{cases} -i.a_{LP}(i), & \text{pre} \quad 1 \leq i \leq N_{LP} \\ 0, & \text{pre} \quad i > N_{LP} \end{cases}.$$

Teraz uvedieme rekurzívny výpočet $\{c_{LP}\}$

$$\begin{aligned} c_{LP}(1) &= -a_{LP}(1) \\ c_{LP}(i) &= -a_{LP}(i) - \sum_{j=1}^{i-1} \left(\frac{j}{i}\right) a_{LP}(i-j).c_{LP}(j), \qquad \text{pre} \quad 2 \leq i \leq N_{LP} \\ c_{LP}(i) &= -\sum_{j=1}^{N_{LP}} \left(\frac{i-j}{i}\right) a_{LP}(j).c_{LP}(i-j), \qquad \text{pre} \quad i > N_{LP} \end{aligned} \tag{12}$$

Počet kepstrálnych koeficientov $N_{CLP}$ môže byť rôzny. Obyčajne sa volí v intervale $0,75.N_{LP} \leq N_{CLP} \leq 1,25.N_{LP}$. Z kepstrálnych koeficientov sa vypočíta vzdialenosť medzi testovaným a referenčným segmentom podľa vzťahu:

$$d_{\mathrm{CEP}}(t,r) = \sum_{i=1}^{N_{CLP}} \left(c_t(i) - c_r(i)\right)^2,$$

kde $t$ a $r$ je testovaný a referenčný prípad, $\{c_t\}$ a $\{c_r\}$ sú kepstrálne koeficienty odvodené LP príslušných prípadov.

$d_{\mathrm{CEP}}(t,r)$ spĺňa všetky vlastnosti vzdialenosti (symetrická, pozitívne definitná) a je vhodná pre budovanie stromovej štruktúry.

Zlepšenie kepstrálnej miery je dosiahnuté maticou váh $\mathbf{W}$:

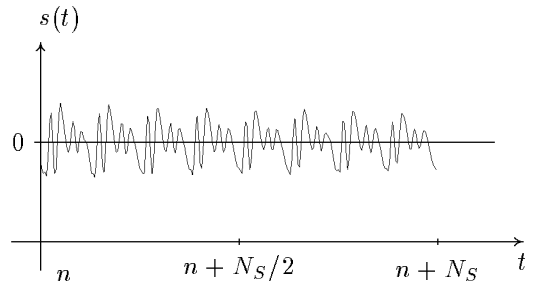
$$d_{\mathrm{MCEP}}(t,r) = (\vec{c}_t - \vec{c}_r)^T \mathbf{W} (\vec{c}_t - \vec{c}_r),$$

prípadne zjednodušená forma

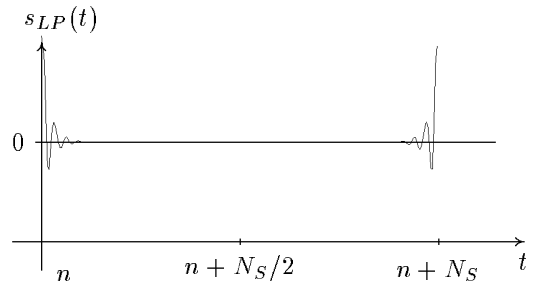
$$d_{\mathrm{WCEP}}(t,r) = \sum_{i=1}^{N_{CLP}} w(i) \left(c_t(i) - c_r(i)\right)^2$$

Pre LP koeficienty $\{a_{LP}\}$ sú vhodné i iné miery odlišnosti, ktoré však nie sú symetrické, napríklad Itakurova miera :

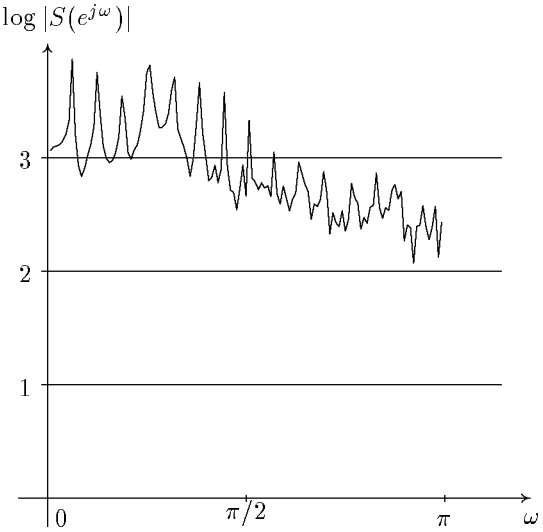
$$\begin{aligned} d_{\mathrm{LLR}}(t,r) &= \log\left(\frac{\vec{a}_r{}^T \mathbf{R}_t \vec{a}_r}{\vec{a}_t{}^T \mathbf{R}_t \vec{a}_t}\right) \\ \vec{a}_t{}^T &= (1, a_t(1), a_t(2), \ldots, a_t(N_{LP})) \\ \vec{a}_r{}^T &= (1, a_r(1), a_r(2), \ldots, a_r(N_{LP})) \\ \mathbf{R}_t(i,j) &= (R|i-j|), \qquad i,j = 0,1,2,\ldots,N_{LP} \\ R(i) &= \sum_{k=0}^{N-1-i} s(k).s(k+i), \qquad i = 0,1,2,\ldots,N_{LP} \end{aligned}$$



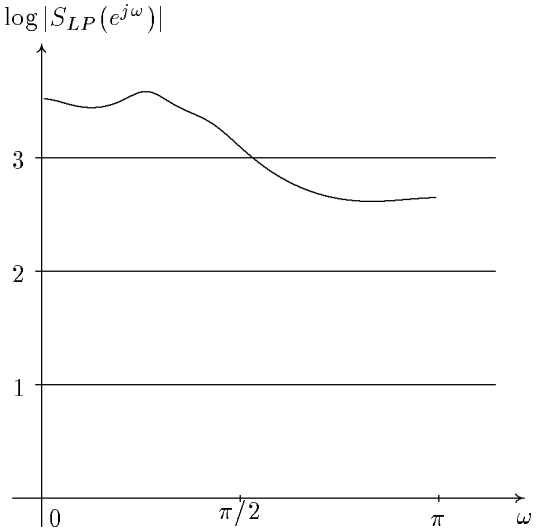

**Obr.47** pôvodný signál



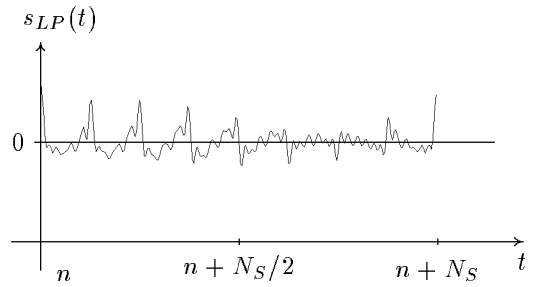

**Obr.48** modelovaný signál LP rádu 7, bez okienka



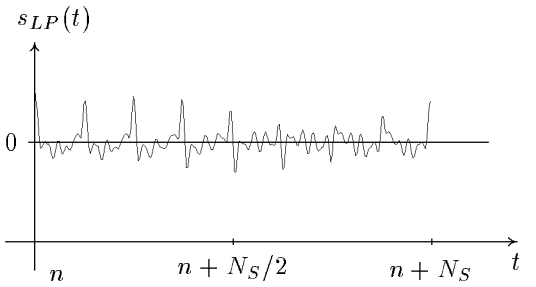

**Obr.49** frekvenčné spektrum signálu FT



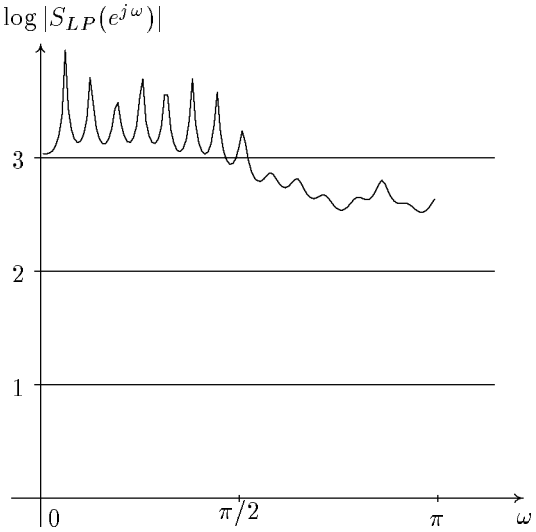

**Obr.50** frekvenčné spektrum LP rádu 7, bez okienka



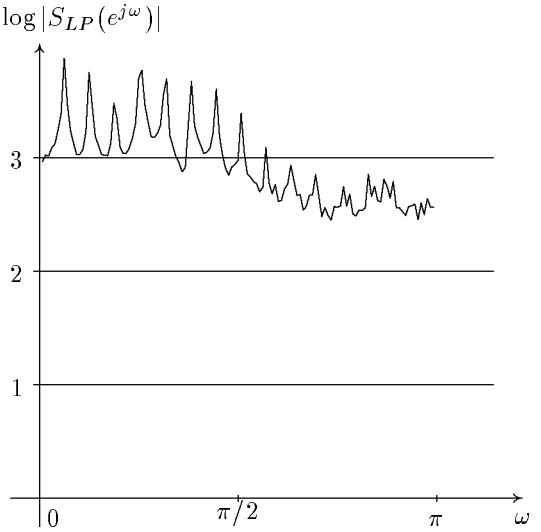

**Obr.51** modelovaný signál LP rádu 30, bez okienka



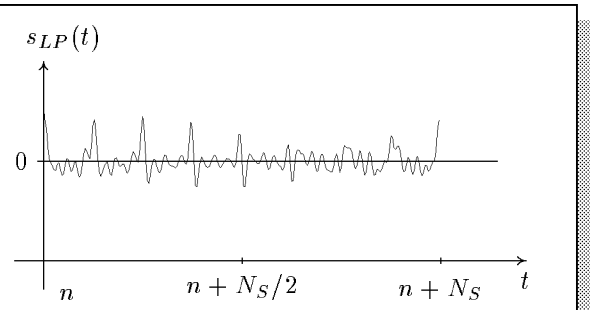

**Obr.52** modelovaný signál LP rádu 150, bez okienka



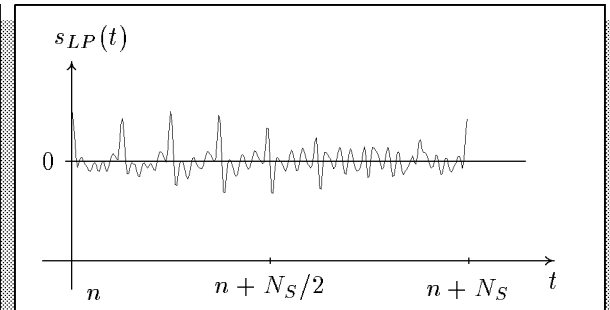

**Obr.53** frekvenčné spektrum LP rádu 30, bez okienka



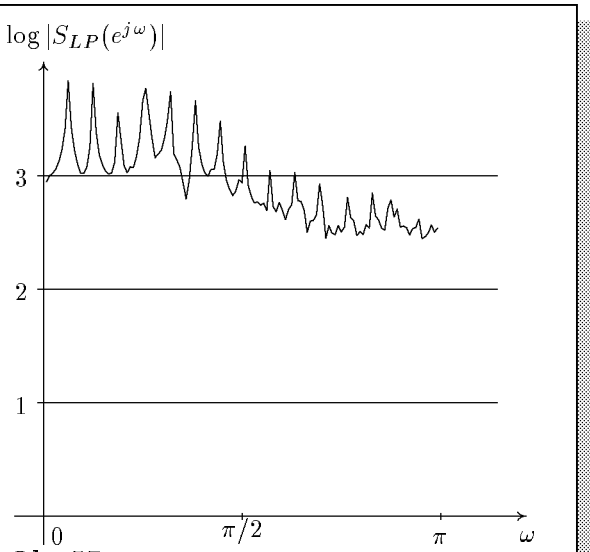

**Obr.54** frekvenčné spektrum LP rádu 150, bez okienka



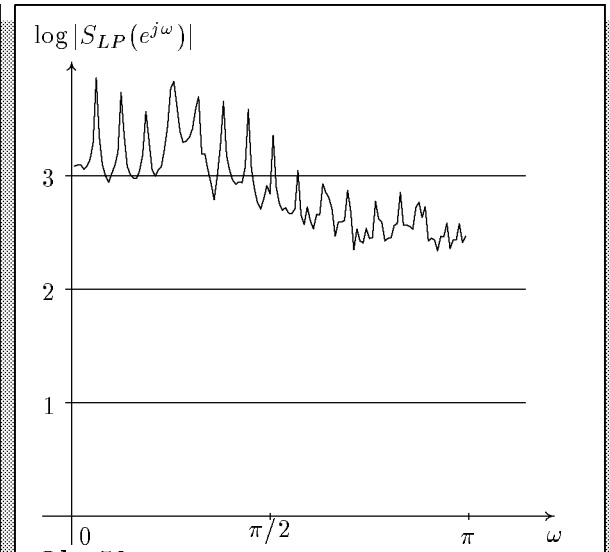

**Obr.55** modelovaný signál LP rádu 256, bez okienka



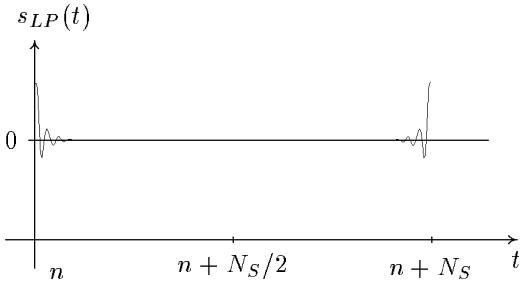

**Obr.56** modelovaný signál LP rádu 400, bez okienka



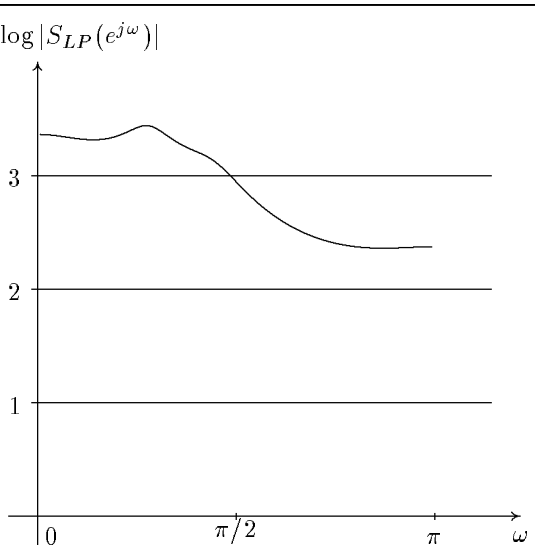

**Obr.57** frekvenčné spektrum LP rádu 256, bez okienka



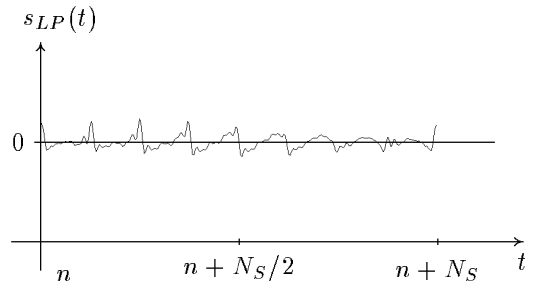

**Obr.58** frekvenčné spektrum LP rádu 400, bez okienka



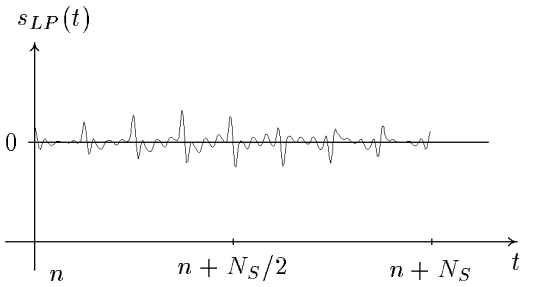

**Obr.59** modelovaný signál LP rádu 7, s okienkom



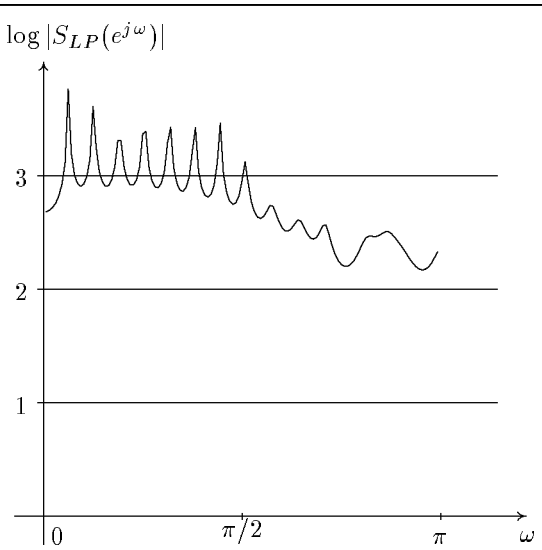

**Obr.60** frekvenčné spektrum LP rádu 7, s okienkom



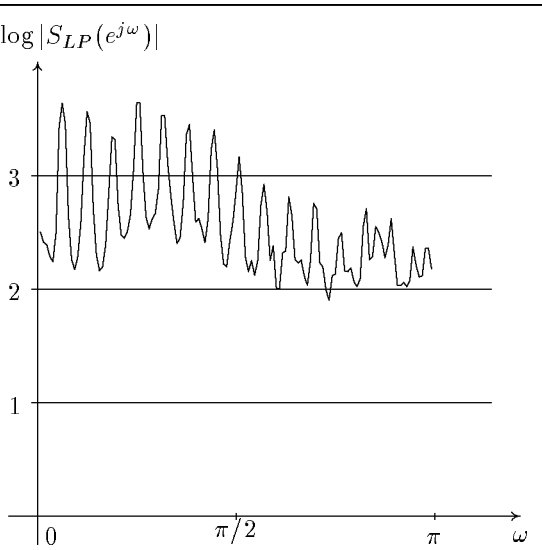

**Obr.61** modelovaný signál LP rádu 30, s okienkom



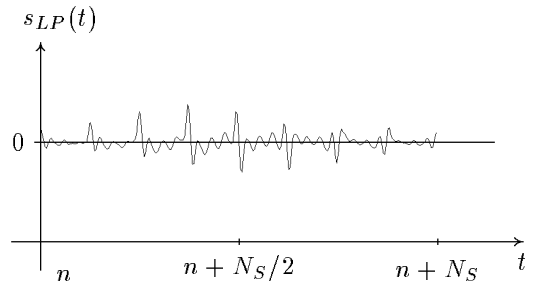

**Obr.62** modelovaný signál LP rádu 150, s okienkom



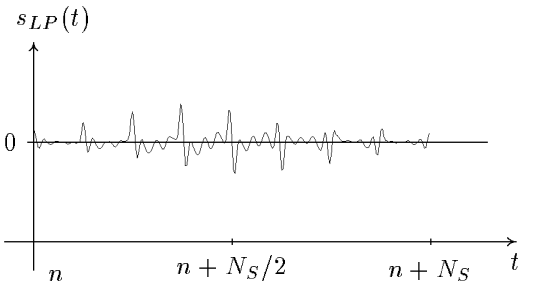

**Obr.63** frekvenčné spektrum LP rádu 30, s okienkom



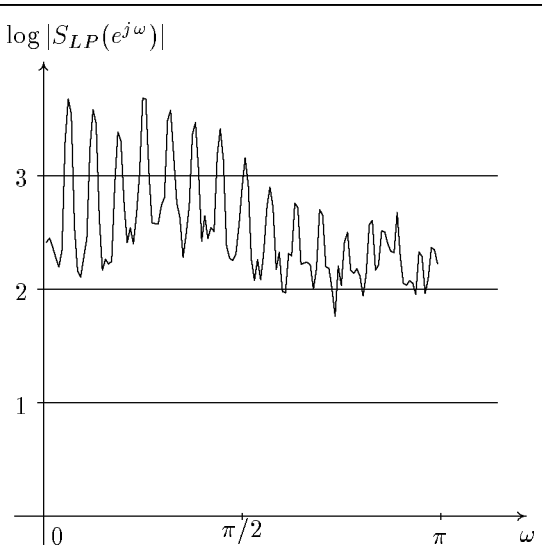

**Obr.64** frekvenčné spektrum LP rádu 150, s okienkom



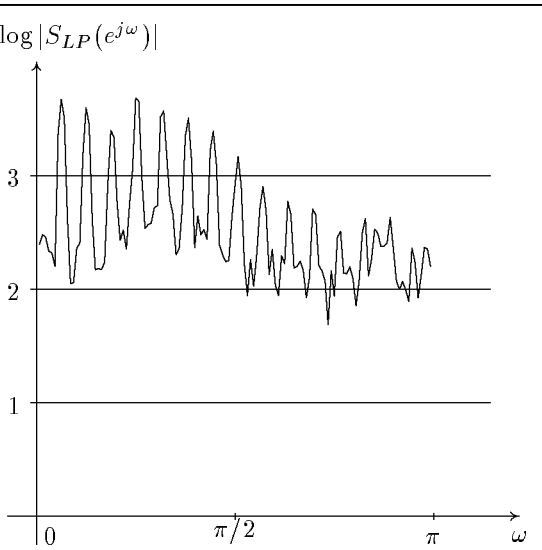

**Obr.65** modelovaný signál LP rádu 256, s okienkom



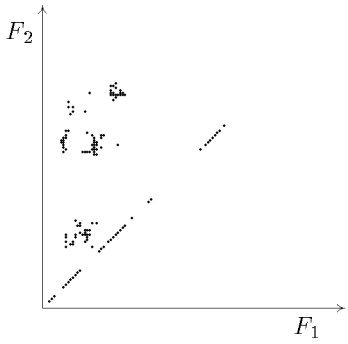

**Obr.66** modelovaný signál LP rádu 400, s okienkom



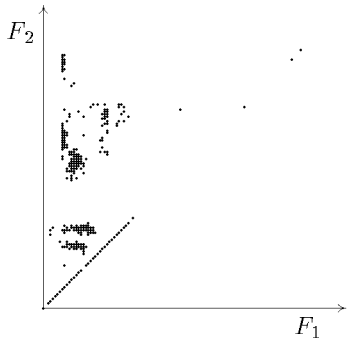

**Obr.67** frekvenčné spektrum LP rádu 256, s okienkom



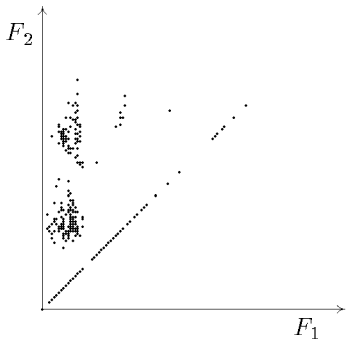

**Obr.68** frekvenčné spektrum LP rádu 400, s okienkom



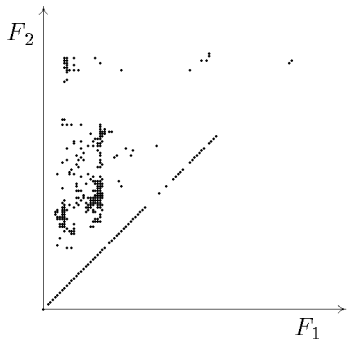

**Obr.69** pozície formantov pre hlásky a,e,i,o,u (žena)




**Obr.70** pozície formantov pre hlásky a,e,i,o,u (muž)




**Obr.71** pozície výskytu formantov pre celú abecedu (muž)




**Obr.72** pozície výskytu formantov pre celú abecedu (dieta)

# 7. Implementácia

Program bol vyvíjaný pod operačným systémom Linux (RH 5.1.). Ako vstup bolo použité štandardné zariadenie `\dev\audio` , ktoré produkuje vzorky signálu frekvenciou $f_S = 8kHz$ a kvalitou $8bit$. Je to kódované v tvare u-low t.j. kvázy logaritmicky. Po prevedení na lineárne číslo je to cca. $10bit$. Ďalej počítam s $16bit$-ovým znamienkovým integerom. Takže by nemal byť problém neskôr to prerobiť na takýto vstup (prípadne iný).

Snímky používam veľkosti 256 a okienko je tiež veľkosti $N_S = 256$. Pre potrebu rýchlosti sa okienka neprekrývajú, iba na seba nadväzujú. Pre dané okienko sa vypočítajú kepstrálne koeficienty LP (10) podľa rekurentného vzťahu (12). Kepstrálne koeficienty tvoria vektor nového prípadu. Nájdem k nemu 20 najbližších susedov v polomere $< 0,4$. Používam metriku $d_{CEP}$. Každý vektor má číselko oblasti, z ktorej pochádza. Pre nový vektor zvolíme oblasť, ktorá je v najväčšej početnosti medzi susedmi. Zistíme lokálne centrum tejto oblasti na základe susedov danej oblasti. Ak sa nový prípad nachádza dostatočne blízko ($< 0,2$) tohto centra, nebude zaradený do stromu. Nie je to potrebné, lebo oblasť nám neobohacuje. Zároveň tým predídeme zahlteniu stromu napr. vektormi ticha. Ak už bol strom zaplnený, najstarší vektor sa vyhodí. Výhodné pri zmene rečníka, čím je zabezpečená adaptácia.

Pre vektor sa vybraná oblasť konvertuje cez ručne editovateľnú tabuľku na hlásku, prípadne sa vypíše číslo oblasti (nenašla sa v tabuľke).

Ako prípadový vektor boli skúšané i formanty resp. LP koeficienty so štandardnou euklidovskou metrikou. Výsledky však boli neuspokojivé. A to vďaka zisťovaniu formantov (lokálne maximá LP obálky) resp. nevhodne zvolenej metrike.

Koeficienty FT neboli skúšané vďaka vysokej časovej náročnosti. Pre bádacie účely bola použitá FT počítaná pomocou FHT. Knižnica je chránená dvomi patentmi (FHT,rýchle goniometrické prepočty)

# 8. Záver

Moja práca odzrkadľuje vývin stavu v rozpoznávaní reči. Začínalo sa s bankami fitrov, neskôr bola rozpracovaná teória lineárnej prediktívnej analýzy, Furierovej transformácie. Práca s prípadmi, vektorová kvantizácia, najnovšie použitie skrytých markovych modelov a neurónových sietí.

Postupne som prešiel historickými fázami. Najprv samotný signál, FT, obálka, priechody nulou, energia, filtre, LP koeficienty, LP kepstrum. Snažil som sa odhadnúť z vizuálnej reprezentácie, čo bude najvhodnejšie pre príznakový vektor.

Na prácu s veľkým množstvom vektorov som musel použiť nejakú prehľadávaciu štruktúru, ktorá urýchlila hľadanie oblastí.

Počas celej práce som mal na pamäti hlavný cieľ, ktorým bola i práca v reálnom čase. Hlavne táto podmienka mi viazala ruky v použití komplikovanejších matematických techník.

Podarilo sa mi dosiahnuť obstojné rozpoznávanie samohlások, ktoré sú príznačné svojou zvučnosťou. Horšie je to zo spoluhláskami, ktoré je zo spracovaného signálu obtiažne rozpoznať. Problém môže byť vo veľkosti okienka ($32ms$), ale i v počte kepstrálnych koeficientov. Odporúčam spustiť si program z diskety a sledovať čísla oblastí pri hovorení samohlások a spoluhlások. Počet oblastí pri experimente s mojím hlasom bol po hodine hovorenia (a,e,i,o,u) nasledovný: a 13, e 4, i 1, o 15, u 4. Po skúšaní samotných spoluhlások sa oblasti spoluhlások prelínali t.j. tá istá oblasť sa vyskytovala pri viacerých spoluhláskach. Ťažko bolo potom zvoliť sémantiku. Po vypustení spoločnej oblasti však nebolo možné spoluhlásku už vôbec rozpoznať, pretože jej časové trvanie sa zmestilo do jedného okienka. Podobný problém nastane pravdaže i pri krátko trvajúcich samohláskach. Pri spoluhláskach do pozornosti silno vstupuje i kontext, samohláska, ktorá nasleduje. Rovnako by sa zišlo zahrnúť do prípadového vektora i informácie o energii, priechodoch nulou. Treba však na to premyslieť vhodnú metriku.

Značne by tiež pomohla nejaká kontextová metóda (HMM, NN, slovník, pravidlá), ktorá by mohla byť ďalším pokračovaním mojej diplomovej práce v tejto oblasti.

Moje striktné odmietanie slovníka sa tiež ukázalo ako neopodstatnené. V inom poňatí, slovník možno chápať ako pravidlá v konkrétnej podobe. Tu je problém s ich veľkým množstvom. Zišlo by sa zo slovníka vyextrahovať všeobecnejšie pravidlá (a už sme pri HMM).

Ďalším zistením je potreba zaviesť i fázu učenia, ktorá je pri nezacvičenom rozpoznávači priam nevyhnutná. (Ručné editovanie tabuľky pravidiel nebolo bohvie čo.) Postupom času, keď rozpoznávač nadobudne prehľad o "stave sveta", sa mu ľahšie "domýšľajú" jemné odchýlky medzi rôznymi rečníkmi. O tom, že to nie je až také jednoduché, svedčí overený fakt veľkých rozdielov medzi rečovým signálom muža, ženy a dieťaťa.

Z naprogramovaného postupu a z experimentov cítim možnosť použitia tejto metódy, tak ako je, skôr len pre rozpoznávanie niekoľkých slov zo slovníka. Na vyššie ciele treba ešte pokračovať v bádaní a aplikovaní.

Neodporúčam bezhlavo sa spoliehať na knižnice z internetu. Buď sa zdá byť dobrá a funguje, kým sa neobjaví maličká chybička. Alebo je potreba knižnicu upraviť t.j. pozerať v zdrojových textoch, ktorých autor nechce aby sa to robilo, čo je veľmi namáhavé a zdĺhavé.

Pozor aj na texty. Radšej si niečo odvodiť, ako zobrať už upravený výsledok, v ktorom môžu byť preklepy.

# Literatúra

[1] Michal Stríženec: Psychológia a kybernetika, r1966, SAV.
[2] Kornej Čukovskij: Od dvoch do piatich, Mladé letá r1959, Bratislava.
[3] Josef Psutka: Komunikace s počítačem mluvenou rečí, Academia, r1995, Praha.
[4] Ivan Šípoš: Horizonty psychológie, r1987, Veda.
[5] František Kassay: Aj učiť sa treba učiť, Smena r1990, Bratislava.
[6] Kurt Tepperwein: Umenie ľahko sa učiť, FONTANA Kiadó r1992, Šamorín.
[7] Developmental Neurocognition: Speach and Face Processing in the First Year of Life. (edited by Bénédicte de Boysson-Bardies, Scania de Schonen, Peter Jusczyk, Peter McNeilage, John Morton), Academic Publishers, r1993, London.
[8] Eva Drlíková, Ladislav Ďurič, ...: Učiteľská psychológia, SPN, r1992, Bratislava.
[9] Tzay Y.Young, King-Sun Fu: Handbook of Pattern Recognition and Image Processing ACADEMIC PRESS, INC., r1986.
[10] Edited by: Pietro Laface, Renato De Mori: Speech Recognition and Understanding. Recent Advances, Trends and Applications, Springer-Verlag Berlin Heidelberg, NATO ASI Series, r1992.
[11] Peter H.Lindsay, Donald A.Norman: Human Information Processing: An Introduction to Psychology (second edition), Academic Press, Inc. r1977, London.
[12] Modisett Noah F., Luter James G.: Speaking clearly, r1979.
[13] Edited by: Antonio J.Rubio Ayuso, Juan M.López Soler: Speech Recognition and Coding. New Advances and Trends, Springer-Verlag Berlin, NATO ASI Series, r1995.
[14] Ábel Kráľ, Ján Sabol: Fonetika a Fonológia (prvé vydanie), SPN r1989, Bratislava.
[15] Edited by L.Auslander, T.Kailath, S.Mitter: Signal Processing Part I: Signal Processing Theory, Springer-Verlag r1990, New York.
[16] Edited by F.A.Grünbaum, J.W.Helton, P.Khargonekar: Signal Processing Part II: Control Theory and Applications, Springer-Verlag r1990, New York.
[17] Edited by G.Goos, J.Hartmanis, Leonard Bolc: Lecture Notes in Computer Science: Natural Language Communication with Computers, Springer-Verlag r1978, New York.
[18] Tomáš Pardel: Písaná reč jej vývin a poruchy u detí (prvé vydanie), SPN r1966, Bratislava.
[19] Norio Katayama, Shin'ichi Satoh: The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries, ACM SIGMOD r1997, Tucson, Arizona.
[20] R.Kurniawati, J.S.Jin, J.A.Shepherd: The SS+ -tree: An Improved Index Structure for Similarity Searches in a High-Dimensional Feature Space, Sydney 2052, Australia.
[21] Timos Sellis, Nick Roussopoulos, Christos Faloutsos: The R+ -tree: A Dynamic Index for Multi-Dimensional objects, r1987, Maryland.
[22] R.Kurniawati, J.S.Jin, J.A.Shepherd: An efficient nearest-neighbour search while varying euclidean metrics, ACM Multimedia r1998, Sydney 2052, Australia.
[23] David A.White, Ramesh Jain: Algorithms and Strategies for Similarity Retrieval, San Diego, California.
[24] Stefan Berchtold, Daniel A.Keim, Hans-Peter Kriegel: The X -tree: An Index Structure for High-Dimensional Data, VLDB r1996, Mumbai (Bombay), India.
[25] Edited by Vijay K.Madisetti, Douglas B.Williams: The digital signal processing handbook, CRC Press USA, r1998
[26] John R.Deller Jr., John G.Proakis, John H.L.Hansen: DISCRETE-TIME Processing of Speech Signals, Macmillan Publishing Company, r1993
[27] http://www.spd.eee.strath.ac.uk/~ interact/ztransform/

...