Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Jiří Daniel

Zvukový analyzátor

Katedra aplikované matematiky

Vedoucí bakalářské práce: doc. RNDr. Jiří Fiala, Ph.D.

Studijní program: Informatika, programování

2010

Rád bych poděkoval všem, kteří mi zapůjčili potřebnou literaturu nebo mě jakkoli podpořili při psaní této bakalářské práce. Zejména děkuji mému vedoucímu doc. RNDr. Jiřímu Fialovi, který mi navrhnul téma analýzy zvuku ke zpracování, a s nímž jsem měl možnost podrobně prokonzultovat náplň práce.

Prohlašuji, že jsem svou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne Jiří Daniel

Název práce: Zvukový analyzátor
Autor: Jiří Daniel
Katedra: Katedra aplikované matematiky
Vedoucí bakalářské práce: doc. RNDr. Jiří Fiala, Ph.D., Katedra aplikované matematiky, Malostranské nám. 25, 118 00 Praha 1
e-mail vedoucího: fiala@kam.mff.cuni.cz
Abstrakt: Cílem práce je naprogramovat utilitu, která by prováděla spektrální analýzu zvukového signálu. Zvláštní zřetel bude kladen na to, aby analýza byla provedena v reálném čase (zde je nutné optimalizovat datové toky v závislosti na výpočetní síle stroje) a aby bylo možno uživatelsky specifikovat šířku a hustotu jednotlivých frekvenčních pásem. Práci je možné rozšířit i na přípravu spektrálních filtrů, které by prováděly navazující úpravu signálu.
Klíčová slova: spektrální analýza, Fourierova transformace, digitální pásmový filtr, zpětná vazba

Title: Sound analyzer
Author: Jiří Daniel
Department: Department of Applied Mathematics
Supervisor: doc. RNDr. Jiří Fiala, Ph.D., Department of Applied Mathematics, Malostranské nám. 25, 118 00 Praha 1
Supervisor's e-mail adress: fiala@kam.mff.cuni.cz
Abstract: The goal is to program a utility that would conduct the spectral analysis of the audio signal. Special attention will be given that the analysis was carried out in real time (here it is necessary to optimize data flows depending on the computational power of machines) and to allow the user to specify the width and the density of each frequency band. The work can be extended to the preparation of spectral filters, which would carry out the downstream signal conditioning.
Keywords: spectral analysis, Fourier transformation, digital band-pass filter, feedback

Obsah

1 Úvod 5
2 Zvuk a jeho digitální zpracování 6
2.1 Zvuk a lidské ucho 6
2.2 Zvuková pásma a oktávy 6
2.3 Zpětná vazba 7
2.4 Zvuk ve vzduchu a v počítači 8
3 Rychlá Fourierova transformace 9
3.1 Fourierova transformace 9
3.2 Diskrétní Fourierova transformace 10
3.3 Rychlá Fourierova transformace 12
3.4 Nevýhody Fourierovy transformace 14
4 Pásmový filtr 15
4.1 Elektronické pásmové filtry 15
4.2 Digitální pásmové filtry 16
4.3 Digitální bikvadratický filtr 16
4.4 Výhody a nevýhody pásmových filtrů 17
5 Uživatelská dokumentace 18
5.1 O programu 18
5.2 Ovládání 18
6 Programátorská dokumentace 20
6.1 Úvod 20
6.2 Práce se zvukovou kartou 20
6.3 Fft 21
6.4 BPF 23
7 Příloha 24
8 Zdroje 25
8.1 Literatura 25
8.2 Internetové zdroje 25
8.3 Podobné programy 25

1 Úvod

Cílem této práce je vytvořit program, který bude zobrazovat hlasitost jednotlivých frekvenčních pásem zvukového vstupu. Podnět k tomuto programu vychází z prostředí zvukařů. Při ozvučování koncertů totiž může dojít ke zpětné vazbě mikrofonu, známé jako nepříjemné pískání mikrofonu. Na trhu sice takové programy jsou, ale většina z nich je drahá a nabízí spoustu funkcí, které zvukaři při živém vystoupení nevyužijí, protože tyto programy jsou navrženy spíše pro práci v laboratoři. Navíc jsou tyto programy výrazně náročnější na výkon a kapacitu počítače.

Požadavky na program jsou následující. Přes zvukovou kartu dáme programu zvukový signál a program nám vykreslí graf hlasitosti jednotlivých frekvenčních pásem. Jelikož je program navrhován pro poslech lidským uchem, zajímají nás frekvence 16 Hz až 20 kHz, které člověk slyší. Toto rozmezí je rozděleno do pásem. Uživatel si může zvolit, jak jemné má zobrazení být. Možnosti jsou jedno, tři, šest nebo dvanáct sloupečků na oktávu. Dále je možno si zvolit jak často se mají zobrazovat změny, jinak řečeno, kolikrát za vteřinu se mají přepočítat hlasitosti jednotlivých pásem. Navíc je možné ještě nastavit, z jak dlouhého úseku dat se má výsledek počítat. Tyto všechny parametry závisí vždy na potřebách zvukaře a především na výkonu počítače. Čím přesnější chceme výsledky mít, tím jsou větší nároky na výkon a paměť počítače.

Při navrhování a řešení této práce jsem narazil na dva způsoby, pomocí nichž se dá tato problematika řešit. Prvním je použití rychlé Fourierovy transformace a druhým je použití pásmových filtrů. Obě řešení mají svá pro i proti, proto jsem volbu metody také nechal na uživateli. Pro srovnání je možné použít i obě metody najednou.

V tomto textu se pokusím vysvětlit nejdůležitější pojmy z této oblasti a algoritmy použité v programu.

2 Zvuk a jeho digitální zpracování

2.1 Zvuk a lidské ucho

Pod pojmem zvuk obecně chápeme mechanické vlnění v nějakém prostředí, které je schopen člověk vnímat uchem. Toto prostředí se nazývá vodič zvuku a je jím nejčastěji vzduch. Zvuk má čtyři základní vlastnosti: frekvenci, intenzitu, průběh kmitání a délku. Frekvence nebo také kmitočet udává výšku tónu. Intenzita udává jeho sílu, hlasitost. Průběh kmitání potom ovlivňuje zabarvení zvuku. Pro tuto práci jsou nejdůležitějšími vlastnostmi kmitočet a síla zvuku.

Lidské ucho je schopné vnímat zvuk o frekvenci v rozmezí 16 Hz až 20 kHz. Zvuk vzniká v přírodě naprosto přirozeně. Může vzniknout jednak od chvějícího se tělesa, ale zvuk může i naopak těleso rozkmitat. Toto je možné díky rezonanci. Některá tělesa jsou totiž schopná při určité frekvenci rezonovat. Proto je například možné pomocí určitých tónů rozkmitat tabulku skla tak, že praskne. Na tomto principu funguje i lidské ucho. Hlemýžď v uchu obsahuje vláskové buňky. Každá z nich rezonuje s jinou frekvencí a pomocí nervových zakončení mozek rozezná, které buňky jsou v neklidu. Nejcitlivější je lidské ucho v rozmezí 2 - 4 kHz. Na okrajích slyšitelného pásma je citlivost lidského ucha velmi malá.

2.2 Zvuková pásma a oktávy

Jak jsem již zmiňoval, program zobrazuje hlasitost frekvencí v určitých pásmech. Tato pásma odpovídají oktávám, popřípadě jejich dílům. Oktáva má v hudbě mírně jiný význam než zde. Nejedná se totiž o osm not, nýbrž o rozmezí frekvencí takových, že jsou v poměru 1:2. To ale platí i v hudbě. Například komorní A (a´) má frekvenci 440 Hz a A o oktávu výše (a´´) má frekvenci 880 Hz a A o oktávu níže má frekvenci 220 Hz. Slyšitelné pásmo (20 – 20 000 Hz) je rozděleno do oktáv tak, že 1 kHz je středem jednoho pásma na logaritmické křivce (při základu 2). Od toho se tedy odvíjí středy ostatních oktáv. Slyšitelné pásmo je tedy rozděleno do 10 oktáv se středy 31,5, 63, 125, 250, 500, 1k, 2k, 4k, 8k a 16k Hz. Pásma zobrazovaná programem potom odpovídají celé oktávě, třetině, šestině nebo dvanáctině oktávy. Toto záleží na volbě uživatele. Toto rozdělení je totiž běžně používáno u ekvalizérů, se kterými zvukaři pracují. Proto je důležité použít jej i při vykreslování výsledků.

Při vykreslování výsledků má osa frekvencí logaritmický průběh. Je důležité si uvědomit, že pásma na obrazovce mají sice stejnou šířku, ale ve skutečnosti obsahuje každé pásmo dvakrát více frekvencí než pásmo předchozí.

V programu jsem potřeboval počítat jak středy jednotlivých pásem (pásmové filtry) tak počítat rozmezí jednotlivých pásem (rychlá Fourierova transformace). K tomu slouží následující vzorečky.

Označme počet pásem v jedné oktávě p a posloupnost $s_1, s_2 \dots s_{10*p}$ označuje středy jednotlivých pásem. Potom platí:

$$s_1 = 1000 * 2^{-5,5+\frac{1}{2*p}} \ Hz$$

$$s_{n+1} = 2 * s_n \ Hz$$

Nechť p opět označuje počet pásem v oktávě a posloupnost $x_0, x_1 \dots x_{10*p}$ označuje hranice těchto pásem. Potom platí:

$$x_0 = 1000 * 2^{-5,5} \cong 22,1 \ Hz$$

$$x_{n+1} = x_n * 2^{\frac{1}{p}} \ Hz$$

Pomocí těchto vztahů lze tedy jednoduše vypočítat jak středy, tak i hranice jednotlivých pásem. Hranice oktáv slyšitelného rozsahu potom jsou přibližně $22;\ 44;\ 88;\ 177;\ 354;\ 707;\ 1,4k; 2,8k;\ 5,7k; 11,3k\ a\ 22,6k$ Hz.

2.3 Zpětná vazba

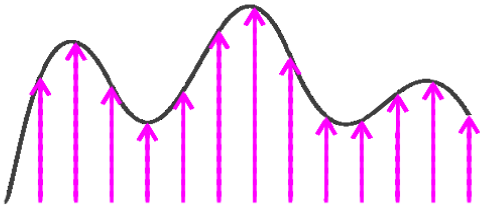
V elektroakustice je zpětná vazba v zásadě nežádoucí jev, který se projevuje nepříjemným pískáním. Zpětná vazba většinou vzniká tak, že v soustavě mikrofon – zesilovač – reproduktor dojde k přijetí zvuku z reproduktoru mikrofonem, zesílení do reproduktorů a opět přijetí mikrofonem. Takto vznikne smyčka, která pokračuje, dokud není dosažen výstupní výkon zesilovače. Frekvence zpětné vazby závisí na mnoha faktorech, jako například vlastnosti reproduktorů, frekvenční charakteristika mikrofonů, směrová charakteristika mikrofonů, vzdálenost mikrofonu od reproduktoru, tvar místnosti, fázový posun atd.

Vedle rušivého efektu posluchačů je v krajním případě také možné, že reproduktory takový nápor nevydrží. Tomu se zvukaři snaží zabránit vhodným rozmístěním reproduktorů a mikrofonů či zeslabením nebo

potlačením příslušné frekvence na ekvalizéru. Právě k tomu potřebují znát hodnotu frekvence popřípadě frekvenčního pásma, které zpětnou vazbu způsobuje.

2.4 Zvuk ve vzduchu a v počítači

Pomocí mikrofonu lze převést akustický signál na signál elektrický. V běžných (dynamických) mikrofonech se k tomu používá membrána, která pohybuje cívkou v magnetickém poli, a díky zákonu magnetické indukce tak vzniká elektrický proud. Tento proud lze chápat jako analogový signál. Pokud chceme zvuk zpracovávat počítačem, potřebujeme ho v digitální podobě. Tento proces za nás dělá zvuková karta počítače. Z analogového signálu získáme vzorkováním vzorkovaný signál. Vzorkovaný signál je v podstatě pouze posloupnost hodnot v čase. Vzorkování funguje tak, že si v určitém intervalu pamatujeme sílu signálu (obr. 1).



Obr. 1: Vzorkování signálu

U vzorkování je velmi důležitý interval vzorkování, který nám udává vzorkovací kmitočet. Z Nyquistova-Shannonova teorému:

„Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byl vzorkován frekvencí alespoň dvakrát vyšší, než je maximální frekvence rekonstruovaného signálu.“

totiž plyne, že pokud se zajímáme o frekvence až do 20 kHz, musíme signál vzorkovat 40 000 krát za vteřinu. To nám zaručuje, že tón o libovolné slyšitelné frekvenci zachytíme minimálně dvakrát za jeho periodu. Pokud bychom vzorkovali signál například pouze 20 000 krát za vteřinu, jevil by se nám signál o frekvenci 20 kHz jako konstantní. Další vlastností vzorkování je, že vzorky mohou nabývat pouze omezeného počtu hodnot. Tedy například 256 nebo jiné mocniny 2.

Pro potřeby našeho programu tedy potřebujeme počítač se zvukovou kartou, která je schopna vzorkovat zvuk z mikrofonu vzorkovacím kmitočtem 40 kHz a s přesností 16 bitů (tedy 16 bitů na jeden vzorek).

3 Rychlá Fourierova transformace

3.1 Fourierova transformace

Fourierova transformace je operace, při níž je jedna komplexní funkce jedné reálné proměnné převedena na jinou komplexní funkci jedné reálné proměnné. V našem případě jde o převod z časové oblasti do oblasti frekvenční. Tedy reálnou proměnnou první funkce je čas a reálnou proměnnou druhé funkce je frekvence. Tento proces se také nazývá spektrální analýza signálu. Původní funkce, amplituda zvuku, je Fourierovou transformací převedena na jinou funkci, která vypovídá o amplitudě a fázi sinusových funkcí různých frekvencí v původním signálu.

Ve spojitém případě je Fourierova transformace definována pomocí integrálu. Tedy Fourierova transformace $S(\omega)$ funkce $s(t)$ je definována vztahem

$$S(\omega) = \int_{-\infty}^{\infty} s(t)e^{-i\omega t}dt$$

Fourierova transformace má také inverzi. Ta je definována vztahem

$$s(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\omega)e^{i\omega t}d\omega$$

Fourierova transformace vychází z Fourierovy analýzy. Jde o využití poznatku, že periodické funkce lze rozepsat pomocí součtu sinů a cosinů. My ale namísto sinů a cosinů používáme komplexní exponenciálu, která se ale dá rozepsat jako

$$e^{2\pi i\omega} = cos2\pi\omega + \ i\ sin2\pi\omega.$$

3.2 Diskrétní Fourierova transformace

Pro využití Fourierovy transformace v počítači je nutné přizpůsobit ji tomu, že máme k dispozici pouze vzorkovaný signál, tedy posloupnost čísel. Také si musíme uvědomit, že nemůžeme transformovat celou funkci najednou, jelikož je nekonečná.

K tomuto účelu slouží diskrétní Fourierova transformace. Její definice je velmi podobná té spojité. Jedná se o transformaci konečné posloupnosti N komplexních čísel na konečnou posloupnost N komplexních čísel. Pro výpočet diskrétní Fourierovy transformace S_k z posloupnosti s_k použijeme vztah

$$S_k = \sum_{n=0}^{N-1} s_n e^{-\frac{2\pi i}{N}kn} \quad , \quad k = 0, \dots, N-1$$

Pro výpočet inverzní transformace potom použijeme vztah

$$s_k = \frac{1}{N} \sum_{n=0}^{N-1} S_n e^{\frac{2\pi i}{N}kn} \quad , \quad k = 0, \dots, N-1$$

Posloupnost S_k potom popisuje amplitudu a fázi sinusových částí obsažených ve vstupní posloupnosti s_k. Hodnota $\frac{|S_k|}{N} = \frac{\sqrt{Re(S_k)^2 + Im(S_k)^2}}{N}$ udává velikost amplitudy funkce sin s frekvencí k period za posloupnost.

Funkce $e^{\frac{2\pi i}{N}kn}, k = 0, \dots, N-1$ jsou navzájem ortogonální, tedy pro $k \neq l$ platí

$$0 = \langle e^{\frac{2\pi i}{N}k}; e^{\frac{2\pi i}{N}l} \rangle \cong \sum_{n=0}^{N-1} e^{\frac{2\pi i}{N}kn} e^{\frac{2\pi i}{N}ln}$$

A pro $k = l$

$$1 = \langle e^{\frac{2\pi i}{N}k}; e^{\frac{2\pi i}{N}l} \rangle \cong \sum_{n=0}^{N-1} e^{\frac{2\pi i}{N}kn} e^{\frac{2\pi i}{N}ln}$$

Kde $\langle\,;\rangle$ značí skalární součin. Tím tvoří hodnoty funkcí $e^{\frac{2\pi i}{N}kn}, k = 0, \dots, N-1$ bázi vektorového prostoru N-dimenzionálních komplexních

vektorů. Diskrétní Fourierova transformace tedy není nic jiného než přechod mezi dvěma různými bázemi téhož vektorového prostoru.

Při výpočtu diskrétní Fourierovy transformace potom počítáme skalární součin původního signálu s bázovými funkcemi. Díky ortogonalitě bázových funkcí platí, že absolutní hodnota koeficientu S_k je rovna integrálu součinu bázové funkce $e^{\frac{2\pi i}{N}k}$ a křivky o stejné frekvenci obsažené v původním signálu. Abychom dostali veličinu nezávislou na délce vektoru, vydělíme hodnotu koeficientu číslem N. Takto dostáváme velikost amplitudy sinusoidy obsažené v původním signálu s frekvencí stejnou, jako má příslušná bázová funkce.

Bázové funkce mají tu vlastnost, že na intervalu dlouhém N čísel má funkce $e^{\frac{2\pi i}{N}k}$ přesně k period. Kdybychom tedy měli posloupnost komplexních čísel, která odpovídá akustickému signálu o délce 1 vteřiny, odpovídala by hodnota $\frac{|S_k|}{N}$ velikosti amplitudy s frekvencí k.

To by ale znamenalo, že bychom museli diskrétní Fourierovu transformaci provádět na vektory o délce 40 000. Proto si rozmyslíme jednoduchý přepočet. Máme-li vzorkovací frekvencí akustického signálu o frekvenci f_v a délku vektoru N, potom po transformaci dostaneme N koeficientů, kde $k - tý$ koeficient odpovídá frekvenci

$$k\frac{f_v}{N} \quad Hz$$

V tomto programu používáme vzorkovací frekvenci 44 100 Hz.

3.3 Rychlá Fourierova transformace

Diskrétní Fourierova transformace má výpočetní složitost $O(N^2)$. Existují ale implementace, které mají výpočetní složitost pouze $O(N\,logN)$. Ty se obecně nazývají rychlá Fourierova transformace. Ukažme si, jak tento algoritmus funguje. s_k označuje k-tý člen původního signálu a S_k k-tý člen diskrétní Fourierovy transformace.

Tedy:

$$S_k = \sum_{n=0}^{N-1} s_n\, e^{-\frac{2\pi i}{N}nk}\ , k = 0, \dots, N-1$$

Rozdělme si sumu na lichá a sudá

$$S_k = \sum_{n=0, n\ sudé}^{N-1} s_n\, e^{-\frac{2\pi i}{N}nk} + \sum_{n=0, n\ liché}^{N-1} s_n\, e^{-\frac{2\pi i}{N}nk} =$$

$$= \sum_{r=0}^{N/2-1} s_{2r}\, e^{-\frac{2\pi i}{N}2rk} + \sum_{R=0}^{N/2-1} s_{2r+1}\, e^{-\frac{2\pi i}{N}(2r+1)k} =$$

$$= \sum_{r=0}^{N/2-1} s_{2r}\, e^{-\frac{2\pi i}{N}2rk} + \sum_{R=0}^{N/2-1} s_{2r+1}\, e^{-\frac{2\pi i}{N}(2r)k}\, e^{-\frac{2\pi i}{N}k} =$$

$$= \sum_{r=0}^{N/2-1} s_{2r}\, e^{-\frac{2\pi i}{N}2rk} + e^{-\frac{2\pi i}{N}k} \sum_{r=0}^{N/2-1} s_{2r+1}\, e^{-\frac{2\pi i}{N}2rk} =$$

A když si uvědomíme, že

$$e^{-\frac{2\pi i}{N}2rk} = e^{-\frac{2\pi i}{N/2}rk}$$

potom

$$= \sum_{r=0}^{\mathrm{N}/2-1} s_{2r}\, e^{-\frac{2\pi i}{N/2}rk} + e^{-\frac{2\pi i}{N}k} \sum_{r=0}^{\mathrm{N}/2-1} s_{2r+1}\, e^{-\frac{2\pi i}{N/2}rk}$$

Nyní si můžeme všimnout, že obě sumy nejsou nic jiného než diskrétní Fourierova transformace velikosti N/2. Jediné co nám ještě vadí je fakt, že k jde od 0 do N-1. Tento problém však můžeme vyřešit rovnicí

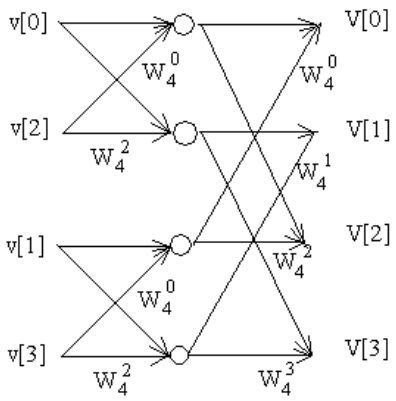
$$e^{-\frac{2\pi i}{N/2}(m+N/2)} = e^{-\frac{2\pi i}{N/2}m} e^{-\frac{2\pi i}{N/2}N/2} = e^{-\frac{2\pi i}{N/2}m} e^{-2\pi\iota} = e^{-\frac{2\pi i}{N/2}m}$$

protože

$$e^{-2\pi i} = \cos(-2\pi) + i\sin(-2\pi) = 1$$

Takto tedy můžeme redukovat výpočet N dimenzionální rychlé Fourierovy transformace na dvě N/2 dimenzionální. Pokud je navíc N mocninou dvojky, můžeme takto pokračovat až do dimenze velikost 2.

Na obrázku (Obr. 2) je zobrazen diagram výpočtu pro N=4, kde $W_N^k = e^{-\frac{2\pi i}{N}k}$.



Obr. 2: Diagram výpočtu pro N=4

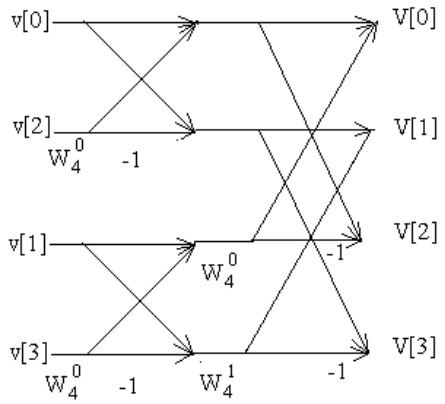
Pokud si navíc ještě všimneme, že

$$e^{-\frac{2\pi i}{\mathrm{N}}(s+N/2)} = e^{-\frac{2\pi i}{\mathrm{N}}s} e^{-\frac{2\pi i}{\mathrm{N}}N/2} = -e^{-\frac{2\pi i}{\mathrm{N}}s}$$

protože

$$e^{-\frac{2\pi i}{\mathrm{N}}N/2} = e^{-\pi i} = \cos(-\pi) + i\sin(-\pi) = -1$$

potom dostáváme diagram zobrazený na obrázku 3.



Obr. 3: Upravený diagram výpočtu pro N=4

Tímto způsobem můžeme poměrně rychle a efektivně spočítat Fourierovu transformaci o velikosti N takové, že N je mocninou dvojky.

3.4 Nevýhody Fourierovy transformace

Problém při použití Fourierovy transformace nastává, když chceme počítat hlasitosti frekvencí hodně přesně a hodně často. Jak vidíme, přesnost výsledků závisí na délce vzorku. To znamená, že pokud máme vzorek signálu o délce 256 při vzorkovací frekvenci 44 100 Hz, známe hlasitost frekvencí 172, 344, 517, 689, 861 Hz atd. Proto musíme volit délku vzorku podle přesnosti, kterou od výpočtu vyžadujeme.

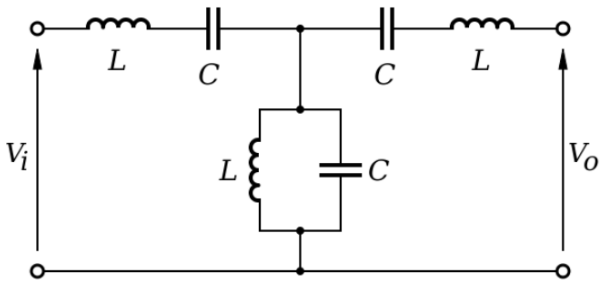
S tím ale souvisí další problém. Chceme-li počítat přesně, musíme brát delší vzorky, tím pádem máme výsledky také méně často. Když bychom například chtěli znát hlasitost každé frekvence samostatně, museli bychom dělat výpočet na vzorek o délce 44 100 prvků. To by ale znamenalo, že výsledek budeme znát pouze jednou za vteřinu.

Tento problém se dá obejít takzvaným okénkem. Nebudeme čekat na všechna nová data, ale budeme si pamatovat některá stará. Kupříkladu když budeme chtít znát výsledek čtyřikrát za vteřinu s přesností po frekvenci, počkáme vteřinu, uděláme výpočet a z předchozích dat si budeme pamatovat poslední ¾ dat. ¼ dat zahodíme a počkáme čtvrt vteřiny. Nová data potom přilepíme nakonec dat starých. Nyní můžeme opět udělat výpočet. Toto řešení má však také svá úskalí, a sice ve výkonu počítače. Čím častěji a čím přesněji chceme výsledek znát, tím výkonnější musíme mít počítač.

4 Pásmový filtr

4.1 Elektronické pásmové filtry

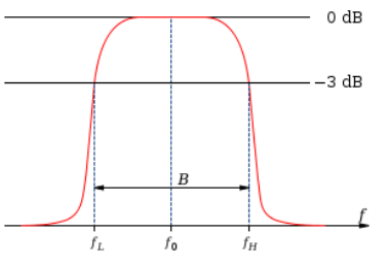
Pásmový filtr (nebo také pásmová propusť) funguje tak, že ze vstupního signálu propustí na výstup pouze určité pásmo frekvencí. Má tedy dvě pevně nastavené mezní frekvence. Na výstup potom propouští pouze tóny mezi těmito frekvencemi. Takové filtry se dají jednoduše sestrojit pomocí cívky, kondenzátoru a odporu. Hodnoty daných součástek potom rozhodují o vlastnostech dané propusti. Zapojení takového filtru je zobrazeno na obrázku.



Obr. 4: Elektrický obvod zapojení pásmového filtru

Mimo pásmových filtrů existují také dolní a horní propusť, které mají pouze jednu mezní frekvenci. Dolní propusť propouští pouze frekvence nižší než je její mezní frekvence a horní propusť pouze frekvence vyšší než je její mezní. Dále existuje také pásmová zádrž, která funguje přesně naopak než pásmová propusť, tedy propouští na výstup všechny frekvence kromě vymezeného pásma.

Žádná z těchto propustí ale není ideální. Skutečnost totiž není taková, že by nežádané frekvence ze signálu úplně odstranila, ona je pouze zeslabí. Na obrázku je znázorněn graf, jak jsou frekvence zeslabovány. f_0 je frekvence středu filtru a f_L je dolní mezní frekvence a f_H je horní mezní frekvence.



Obr. 5: Graf frekvenční charakteristiky pásmového filtru

4.2 Digitální pásmové filtry

Jelikož je pásmový filtr v elektronice celkem jednoduchou a spolehlivou záležitostí, nabízí se, že digitální implementace bude také jednoduchá. Ukazuje se ale, že konečná implementace sice jednoduchá je, ale je velmi náročné se k ní propracovat.

Pásmový filtr je jeden typ lineárního filtru. Lineárním filtrem nazveme takový filtr, který vyhovuje principu superpozice. Princip superpozice se používá při analýze lineárních soustav, v nichž působí více zdrojových veličin. V elektronice se tak dají analyzovat elektrické obvody. Díky Kirchhoffovu zákonu platí, že součet proudů vedoucích do uzlu se musí rovnat součtu proudů vedoucích z uzlu a součet všech dílčích napětí v jedné větvi musí odpovídat celkovému napětí nad celou větví. Jinými slovy, odezva lineárního systému tvořeného tímto filtrem na součet dvou či více signálů musí být rovna součtu odezev tohoto systému na jednotlivé signály. Takovéto filtry se popisují pomocí diferenčních rovnic. Tyto rovnice se dají velmi jednoduše a rychle implementovat.

Jinými způsoby popisu lineárního filtru je přenosová funkce. Tu lze dostat z-transformací diferenční rovnice filtru a podílem jeho výstupu ke vstupu. Dalším způsobem popisu je rozložení jeho nulových bodů a pólů v komplexní rovině. Tato forma popisu je vhodná pro určení stability filtru.

4.3 Digitální bikvadratický filtr

Digitální bikvadratický filtr je lineární filtr druhého řádu, to znamená že přenosová funkce tohoto filtru je tvořená podílem dvou kvadratických funkcí. Obecně má tedy tato funkce tvar

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Označíme-li vstupní signál jako posloupnost $x[n]$ a výstupní signál jako posloupnost $y[n]$ potom dostáváme diferenční rovnici

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) - a_1 y(n-1) - a_2 y(n-2)$$

Hodnoty koeficientů rozhodují o mezních frekvencích, a dalších vlastnostech filtru. Klíč k nastavení koeficientů je následující:

f_v je frekvence vzorkování, tedy v našem případě $f_v = 44\,100$
f_0 udává frekvenci ve středu pásma, které chceme filtrovat
BW je šířka pásma v oktávách, tedy v našem případě 1, 3, 6 a 12

$$\omega = \frac{2\pi\, f_0}{f_v} \quad , \qquad \alpha = \sin(\omega) * \sinh\left(\frac{ln(2)}{2} * \; BW * \frac{\omega}{sin(\omega)}\right)$$

$$b_0 = \frac{\alpha}{1+\alpha} \quad , \qquad b_1 = 0 \quad , \qquad b_2 = -\frac{\alpha}{1+\alpha}$$

$$a_1 = \frac{-2 * \cos(\omega)}{1+\alpha} \quad , \qquad a_2 = \frac{1-\alpha}{1+\alpha}$$

Takto lze jednoduchým způsobem určit koeficienty digitálního pásmového filtru. Použití v programu je následující. Vyrobíme si tolik filtrů, kolik chceme mít pásem s příslušnými parametry a nainicializujeme jejich proměnné. Když chceme znát hlasitost určitého pásma, „prosypeme“ filtrem dosavadní vzorky. U toho si budeme pamatovat výstup z filtru. Lépe řečeno, budeme sčítat jeho absolutní hodnoty. Na konci tuto sumu vydělíme počtem vzorků. Tím dostaneme průměrnou amplitudu daného frekvenčního pásma. Tento proces uděláme pro všechny pásma.

4.4 Výhody a nevýhody pásmových filtrů

Obrovskou výhodou pásmových filtrů je jejich jednoduchá implementace, nenáročnost na výkon ani paměť počítače. Avšak cenou za tento komfort je jejich mírná nepřesnost. Je-li některá frekvence blízko mezní hranice našeho filtru, ovlivní jej to. To v tomto případě ale zase tolik nevadí, protože potřebujeme znát výsledky pouze přibližně.

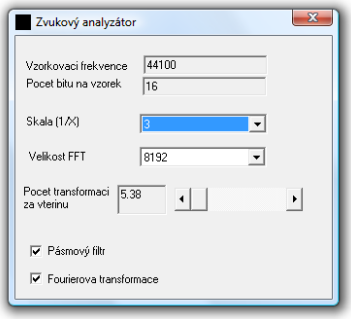
5 Uživatelská dokumentace

5.1 O programu

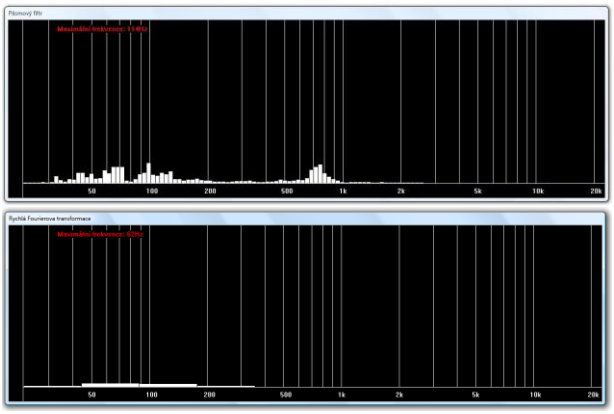
Tento program slouží k analýze zvukového signálu. Zobrazuje hlasitost jednotlivých pásem frekvencí slyšitelného spektra. Je určen pro zvukaře k detekci zpětné vazby, ale jeho jiné využití není nikterak vyloučeno. Instalace programu není nutná, celý program je obsažen v jednom spustitelném souboru. Pro běh programu je nutné mít v počítači zvukovou kartu s mikrofonem.

5.2 Ovládání

Po spuštění programu se uživateli zobrazí 3 okna. Hlavní okno obsahuje ovládací prvky.



Obr. 6: Hlavní okno programu



Obr. 7: Okna s vykreslenými výsledky

Skala znamená, na kolik sloupečků bude rozdělena 1 oktáva. Možnosti jsou 1, 3, 6 a 12. *Velikost FFT* ovlivňuje délku vzorku signálu. Tím pádem zároveň i přesnost výsledků počítaných Fourierovou transformací. Možnosti tohoto nastavení jsou 2048, 4096, 8192, 16 384, 32 768 a 65 536. *Pocet transformaci za vterinu* udává, kolikrát se při tomto nastavení provede výpočet hodnot. Pomocí posuvníku se dá počet změnit. Minimální počet je takový, aby byly počítány výsledky z celého signálu. Maximální počet je 43,06 výpočtů za vteřinu, což odpovídá výpočtu po 1024 vzorcích. Pomocí checkboxů *Pásmový filtr* a *Fourierova transformace* se zapíná a vypíná výpočet hodnot pomocí dané metody.

Vykreslování výsledků je pro každou metodu zvlášť. Okna jsou stejná. Osu x tvoří frekvence s logaritmickým měřítkem při základu 2. Mřížka v pozadí naznačuje hodnoty 10, 20, 30, ..., 100, 200, ...,1k, 2k, ...,10k a 20k Hz. Některé hodnoty jsou pod osou popsány popiskem.

V levém horním rohu je červeným písmem napsána frekvence středu pásma s nejvyšší hodnotou.

Program je napsán pro 32 bitové verze operačního systému Windows.

6 Programátorská dokumentace

6.1 Úvod

Program je psán v jazyce C++ v Microsoft Visual Studiu 2008 Prfessional Edition jako Win32 Project. V programu jsem používal pouze standardní prostředky jazyka C++.

Celý program je dělen do několika souborů. Složka `Audio` obsahuje především zdrojový kód týkající se obsluhy zvukové karty a také implementace transformačních algoritmů. Složka `Lib` obsahuje především kód použitých komponent a tříd, které obsluhují jejich události. V kořenovém adresáři je potom kód hlavního programu a vykreslování výsledků na obrazovku.

6.2 Práce se zvukovou kartou

Získávání dat ze zvukové karty je celkem jednoduché, pokud člověk umí pracovat s vícevláknovými aplikacemi. Celé to funguje tak, že si vytvoříme kruhový spojový seznam bufferů a separátní vlákno, které je probuzeno operačním systémem, když jsou připravena data ze zvukové karty. Toto vlákno potom dá zvukové kartě další buffer v pořadí a získaná data předá dál programu. Toto řešení samozřejmě vyžaduje ochranu zámky.

6.3 Fft

Třída Fft implementuje algortimus rychlé Fourierovy transformace. Zde je hlavička třídy. Popíšeme si nejdůležitější části.

```
typedef std::complex<double> Complex;

class Fft
{
public:
    Fft (unsigned points, unsigned sampleRate);
    void    Transform (std::vector<int> const & data);
    double  GetIntensity (unsigned i) const;
    unsigned Points () const { return _points; }
    double GetFrequency (unsigned point) const;
    int HzToPoint (int freq) const ;
    int MaxFreq() const { return _sampleRate; }
private:
    void DataIn (std::vector<int> const & data);
    void PutAt (unsigned i, double val);
    unsigned      _points;     //délka vzorku
    unsigned      _sampleRate; //vzorkovací frekvece
    unsigned      _logPoints;  //log(_points) při základu 2
    double        _sqrtPoints; //druhá odmicnina z _points
    std::vector<unsigned>  _aBitRev;//pole obsahující mapování
    std::vector<Complex>   _X;//pole vstupu a výstupu algoritmu
    std::vector<std::vector<Complex> > _W;//hodoty exponenciály
    std::vector<double>       _tape;      //vstupní signál
};
```

Vektor `_tape` slouží k uchovávání vstupního signálu. Funguje jako fronta. Při určitém nastavení programu totiž dochází k tomu, že dostaneme na vstupu méně vzorků, než potřebujeme k výpočtu nových hodnot. Proto vždy pouze umažeme staré hodnoty ze začátku a na konec je doplníme novými.

Ve vektoru `_W` jsou uložené spočítané hodnoty funkce $e^{\frac{2\pi\iota}{N}nk}$, kde platí $W[n][k] = e^{\frac{2\pi\iota}{N}nk}$.

Do vektoru _X jsou uloženy hodnoty vstupního signálu podle určitého mapování. Při transformaci jsou hodnoty měněny přímo v tomto vektoru. Toto mapování odpovídá pořadí prvků při rychlé Fourierově transformaci. Tedy například pokud je počet prvků 4, jsou hodnoty do vektoru uloženy v pořadí 0-tý, 2-hý, 1-ní, 3-tí. Toto mapování je obsaženo ve vektoru `_aBitRev` tak, že i-tý prvek patří na pozici `_aBitRev[i]`.

Procedura PutAt (unsigned i, double val); vkládá hodnotu `val` i-tého prvku na pozici dle mapování do pole `_X`. Procedura `DataIn`

`(std::vector<int> const & data);` zkopíruje obsah vektoru data na konec vektoru `_tape`.

Funkce `double GetFrequency (unsigned point) const;` vrátí frekvenci, jejíž amplitudu popisuje `point`-tý prvek ve vektoru `_X`. Funkce `int HzToPoint (int freq) const;` vrátí pozici ve vektoru `_X`, která popisuje amplitudu frekvence `freq`. Funkce `double GetIntensity (unsigned i) const;` vrátí velikost amplitudy na `i`-té pozici v poli `_X`.

Konstruktor této třídy inicializuje hodnoty všech proměnných. Zde je kód inicializace mapování.

```
int rev = 0;
unsigned halfPoints = _points/2;
for (unsigned i = 0; i < _points - 1; i++)
{
    _aBitRev [i] = static_cast<unsigned> (rev);
    unsigned mask = halfPoints;
    while (rev >= static_cast<int> (mask))
    {
        rev -= mask;
        mask >>= 1;
    }
    rev += mask;
}
_aBitRev [_points-1] = _points-1;
```

Takto potom vypadá algoritmus výpočtu rychlé Fourierovy transformace

```
unsigned step = 1;
for (unsigned level = 1; level <= _logPoints; level++)
{
    unsigned increm = step * 2;
    for (unsigned j = 0; j < step; j++)
    {
        // U = exp ( - 2 PI j / 2 ^ level )
        Complex U = _W [level][j];
        for (unsigned i = j; i < _points; i += increm)
        {
            Complex T = U;
            T *= _X [i+step];
            _X [i+step] = _X[i];
            _X [i+step] -= T;
            _X [i] += T;
        }
    }
    step *= 2;
}
```

6.4 BPF

Nebo-li band-pass filter, tedy pásmový filtr. Tato třída implementuje řešení pomocí pásmových filtrů. Konstruktor nainicializuje všechny proměnné a vytvoří všechny filtry s potřebnými koeficienty. Procedura `Transform (std::vector<int> const & data);` předá vektor se vstupními daty všem filtrům a uloží výsledné hodnoty do vektoru `_X`. Funkce `GetIntensity` a `GetFrequency` vrátí velikost amplitudy popřípadě střední frekvenci `i`-tého filtru.

```
class Bpf
{
public:
      Bpf (unsigned scale, unsigned sampleRate);
      void Transform (std::vector<int> const & data);
      double  GetIntensity (unsigned i) const;
      double GetFrequency (unsigned point) const;
private:
      unsigned    _scale;     //počet sloupečků v oktávě
      unsigned    _sampleRate; //vzorkovací frekvence
      std::vector<double> _X; //hodnoty amplitud
      std::vector<Biquad> _filters; //vektor filtrů
};
```

Třída Biquad zastupuje samotný pásmový filtr. Konstruktor nainicializuje proměnné tak, aby měl dané vlastnosti. Fuknce `Transform` potom dělá vlastní práci filtru. Vrací průměr absolutních hodnot velikosti amplitudy.

```
class Biquad {
public:
      Biquad(double freq, double srate, double bandwidth);
      ~Biquad(){};
      double Transform(std::vector<int> const & data);
      double freq;
private:
    double a0, a1, a2, a3, a4;
    double x1, x2, y1, y2;
};
```

7 Příloha

Na přiloženém CD jsou soubory se zdrojovými kódy i soubory potřebné pro otevření projektu ve Visual Studiu a zkompilovaný program. Dále je na disku také přiložen tento text ve formátu pdf.

8 Zdroje

8.1 Literatura

Mark A. Pinsky (2009): *Introduction to Fourier analysis and wavelets*, American Mathematical Society

Donald B. Percival (1993): *Spectral analysis for physical applications: multitaper and conventional univariate techniques*, Cambridge University Press

Pete Becker (2007): *The C++ standard library extensions: a tutorial and reference*, Addison-Wesley

Ryan, Robert D. (1995): *Wavelets and multiscale signal processing*, London : Chapman and Hall

Kaiser, Gerald (1994): *A friendly guide to wavelets*, Boston : Birkhäuser

8.2 Internetové zdroje

http://cs.wikipedia.org/wiki/Zvuk

http://cs.wikipedia.org/wiki/Fourierova_transformace

http://en.wikipedia.org/wiki/Fourier_transform

http://de.wikipedia.org/wiki/Filter_%28Elektrotechnik%29

http://en.wikipedia.org/wiki/Digital_signal_processing

http://en.wikipedia.org/wiki/Digital_filter

http://msdn.microsoft.com/

http://www.musicdsp.org/

http://www.pdfgeni.com/book/Digital-Equalization-Methods-for-Audio-Systems-pdf.html

Podobné programy

TrueRTA - http://www.trueaudio.com/

Baudline - http://www.baudline.com/

SpectraPLUS - http://www.spectraplus.com/