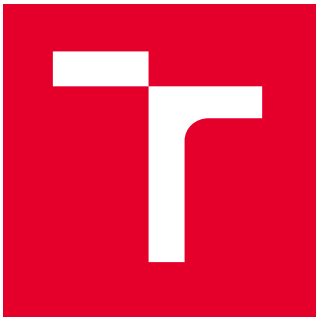


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

GIANT FFT PRO KONVERZI VZORKOVACÍHO KMITOČTU AUDIO SIGNÁLU

GIANT FFT FOR AUDIO SIGNAL RESAMPLING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jan Pavlačka**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Mgr. Pavel Rajmic, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Jan Pavlačka ***ID:*** 209453
Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Giant FFT pro konverzi vzorkovacího kmitočtu audio signálu

POKYNY PRO VYPRACOVÁNÍ:

Změna vzorkovacího kmitočtu audio signálu je často potřebnou operací. Tato funkce je implementována v každém DAW. Seznamte se s klasickými metodami pro změnu vzorkovacího kmitočtu [1]. Poté nastudujte nový přímočarý přístup přes FFT obřích rozměrů podle zdroje [2].

Implementujte oba přístupy v Matlabu. Porovnejte na zvukové databázi zahrnující různorodé signály, a to zejména pomocí objektivních kritérií. Neopomeňte srovnat také výpočetní náročnost a další faktory.

DOPORUČENÁ LITERATURA:

[1] Vaidyanathan, P. P. Multirate Systems and Filter Banks, Pearson College Div, 1992. ISBN 978-0136057185.
[2] Välimäki, V., Bilbao, S. Giant FFTs for Sample-Rate Conversion, J. Audio Eng. Soc., vol. 71, no. 3, 2023.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: prof. Mgr. Pavel Rajmic, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

Abstrakt

Tato bakalářská práce se věnuje konverzi vzorkovacího kmitočtu audio signálů pomocí různých metod konverze. Důraz je kladen na obecnou konverzi mezi jakýmikoliv dvěma vzorkovacími kmitočty běžně užívanými v audio aplikacích. V teoretické části práce je diskutována problematika klasického přístupu ke konverzi, číslicové filtry typu FIR a relativně nová metoda Giant FFT. V praktické části je popsána implementace jednotlivých metod. Na závěr jsou provedeny testy, porovnání a vyhodnocení, do kterých je zahrnut i software Reaper.

Klíčová slova

Konverze vzorkovacího kmitočtu, FIR filtr, Giant FFT, audio signály, Reaper, SDR

Abstract

This bachelor thesis is concerned with the conversion of the sampling frequency of audio signals using different conversion methods. The focus is on the general conversion between any two sample rates commonly used in audio applications. The theoretical part of the paper discusses the classical approach to conversion, FIR type digital filters and the relatively new Giant FFT method. The practical part describes the implementation of each method. Finally, tests, comparisons and evaluations are performed, which include the Reaper software.

Keywords

Sample rate conversion, FIR filter, Giant FFT, audio signals, Reaper, SDR

Bibliografická citace

PAVLAČKA, Jan. *Giant FFT pro konverzi vzorkovacího kmitočtu audio signálu* [online]. Brno, 2024 [cit. 2024-05-08]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/153582. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Pavel Rajmic.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Jan Pavlačka
VUT ID studenta:	209453
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Giant FFT pro konverzi vzorkovacího kmitočtu audio signálu

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.
V Brně dne: ----------

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce prof. Mgr. Pavlovi Rajmicovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

Jednou z velmi důležitých oblastí zpracování signálu, ve které se aplikuje konverze vzorkovacího kmitočtu, je digitální zpracování audio souborů [10]. Jedná se o proces, který je implementován v každém běžném DAW (digital audio workstation). Obvykle se používá mnoho různých vzorkovacích kmitočtů [9] a je potřeba, aby případná konverze mezi nimi co nejlépe zachovala původní audio signál. Konverze je například použita pro sjednocení vzorkovacích kmitočtů ve více různých souborech, pro dodržení standartních vzorkovacích kmitočtů pro různé nosiče (např. pro CD 44,1 kHz [8]), nebo dříve také pro redukci dat audio souboru. Alternativou k digitálním metodám konverze je proces, při kterém se digitální audio soubor převede pomocí A/D převodníku na analogový, poté se reprodukuje, opětovně nahraje a převede zpět na digitální soubor. Je prokázáno, že takový přístup je oproti digitální konverzi vzorkovacího kmitočtu nevýhodný [4].

Cílem této bakalářské práce je kvalitativní srovnání různých metod konverze audio signálů a následné vyhodnocení výsledků. Budou porovnány metody klasické postupné konverze, alternativní postupné konverze, metody Giant FFT a metody sinc interpolace, kterou běžně využívají různé DAW. Toho bude dosaženo výpočtem SDR, MSE a celkového času konverze. Všechny metody budou implementovány v prostředí softwaru Matlab s výjimkou metody sinc interpolace, která bude realizována pomocí softwaru Reaper.

První kapitola této práce se zabývá teorií základních operací v multitaktních systémech jako nadvzorkování, podvzorkování, interpolace a decimace, a s tím souvisejícími základními typy konverze vzorkovacího kmitočtu. Ve druhé kapitole je diskutována problematika návrhu číslicových filtrů typu FIR pomocí váhovacího okna. Třetí kapitola se zabývá metodou konverze zvanou Giant FFT, která ke konverzi vzorkovacího kmitočtu používá Fourierovy transformace s obrovskými počty vzorků v řádech milionů. Čtvrtá kapitola se zabývá problematikou implementace jednotlivých metod v prostředí softwaru Matlab. Pátá kapitola poskytuje výsledky porovnání jednotlivých metod. Testování je provedeno na umělých signálech vytvořených v programu Matlab a uložených jako soubory .mat a soubory .wav, a na čtyřech celých skladbách různých žánrů.

1. ZÁKLADNÍ OPERACE MULTITAKTNÍCH SYSTÉMŮ

V této kapitole bude vysvětlen princip základních operací, které se využívají pro konverzi vzorkovacího kmitočtu viz [1], [3], [4], [5].

Nejjednoduššími operacemi jsou snížení vzorkovacího kmitočtu nazývané podvzorkování signálu a zvýšení vzorkovacího kmitočtu nazývané nadvzorkování signálu. Doplněním těchto operací o číslicové filtry pak získáme operace decimace a interpolace.

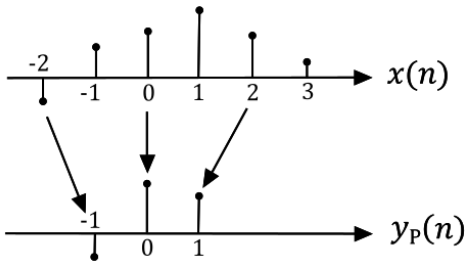
1.1 Podvzorkování signálu

Podvzorkováním diskrétního signálu rozumíme odebrání konečného počtu vzorků ze vstupního signálu.

Pro výsledný signál platí rovnice

$$y_{\mathrm{P}}(n) = x(Mn)\,, \tag{1.1}$$

kde M je celé číslo. Výsledný signál $y_{\mathrm{P}}(n)$ je tvořen výběrem každého M-tého vzorku ze vstupního signálu $x(n)$ [1]. Jedná se o destruktivní operaci, tj. vzorky, které nebyli vybrány jsou nevratně ztraceny. Tato operace se též chybně označuje jako decimace nebo také jako komprese vzorkovacího kmitočtu.



Obrázek 1.1 Demonstrace podvzorkování pro $M = 2$

Pomocí diskrétní Fourierovy transformace a transformace $\mathcal{Z}$ [4] lze odvodit výsledný vztah pro kmitočtové spektrum podvzorkovaného signálu

$$Y_{\mathrm{P}}\left(\mathrm{e}^{\mathrm{j}\omega}\right) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(\mathrm{e}^{\mathrm{j}\frac{\omega-2\pi k}{M}}\right), \tag{1.2}$$

kde $X(\mathrm{e}^{\mathrm{j}\omega})$ je kmitočtové spektrum původního signálu a proměnná ω je úhlový kmitočet.

Podvzorkování signálu způsobuje periodizaci kmitočtového spektra, tj. kmitočtové spektrum původního signálu je znovu zrcadleno na frekvencích rovných celočíselným násobkům vzorkovacího kmitočtu. Pokud se nově vzniklé složky spektra objeví na stejných kmitočtech jako složky shodné s původním spektrem, vzniká neodstranitelný aliasing.

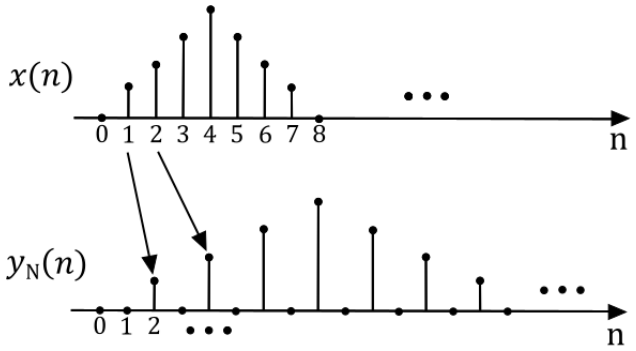
1.2 Nadvzorkování signálu

Nadvzorkování diskrétního signálu rozumíme přidání konečného počtu vzorků do vstupního signálu.

Pro výsledný signál platí rovnice

$$y_N(n) = \begin{cases} x\left(\frac{n}{L}\right), & \text{pro } n \text{ dělitelné } L \\ 0, & \text{jinak}, \end{cases} \tag{1.3}$$

kde L je celé číslo. Všechny vzorky vstupního signálu $x(n)$ jsou z původní pozice n posunuty na pozici Ln a doplněny o nulové vzorky [1]. Původní signál je možné zpětně rekonstruovat. Tato operace se též označuje jako expanze.



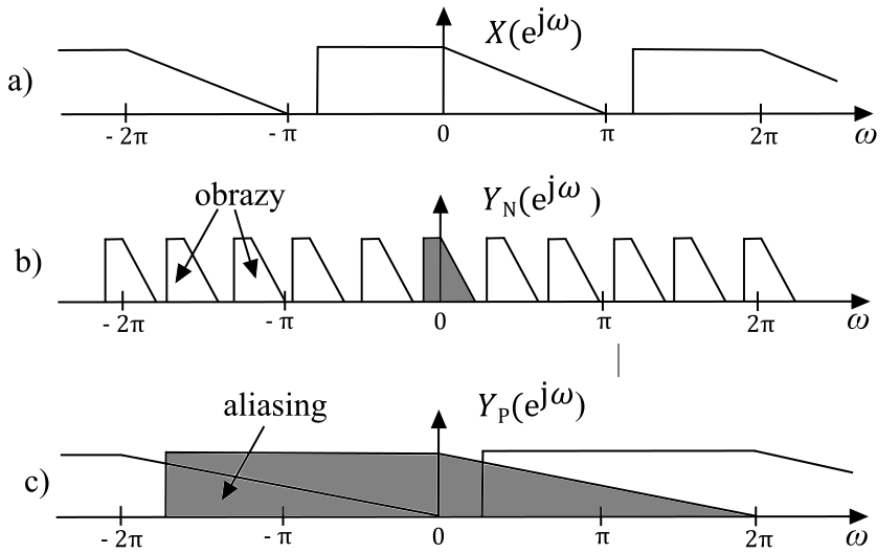
Obrázek 1.2 Demonstrace nadvzorkování pro $L = 2$

Obdobným způsobem jako v oddíle 1.1 můžeme získat vztah pro kmitočtové spektrum nadvzorkovaného signálu

$$Y_N(e^{j\omega}) = X(e^{j\omega L}), \tag{1.4}$$

kde $X(e^{j\omega L})$ je původní signál doplněný o nulové vzorky a proměnná ω je relativní úhlový kmitočet [4].

Nadvzorkování signálu způsobuje vznik obrazů spektra původního signálu, kvůli vloženým nulovým vzorkům. Vzniklé složky kmitočtového spektra, které se nacházejí v pásmu mezi polovinou původního vzorkovacího signálu a polovinou nového vzorkovacího kmitočtu, by se zde neměly nacházet, protože dále způsobují nežádané zkreslení nadvzorkovaného signálu.

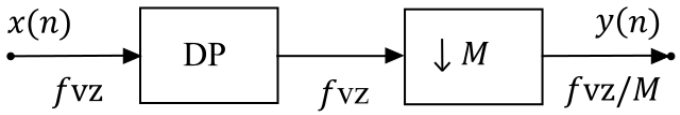


Obrázek 1.3 Transformace spektra původního signálu (a) ve frekvenční doméně. (b) Nadvzorkování pro $L = 5$. (c) Podvzorkování pro $M = 2$, kde lze pozorovat výrazný aliasing

1.3 Decimace signálu

Jak již bylo zmíněno v oddíle 1.1 podvzorkování signálu má za následek snížení vzorkovacího kmitočtu a může způsobit aliasing. Decimace signálu spočívá v zabránění vzniku aliasingu tím, že je před podvzorkováním aplikován antialiasingový filtr [1].

Antialiasingový filtr je typu dolní propust a je vybrán tak, aby byly propuštěny pouze kmitočty nižší než polovina nového vzorkovacího kmitočtu. Původní vzorkovací kmitočet je značen f_{vz}. Výběrem všech parametrů a konkrétním návrhem filtru se bude zabývat kapitola 2.

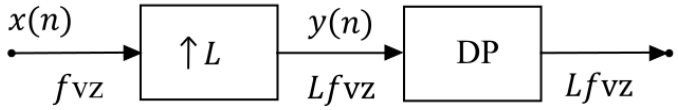


Obrázek 1.4 Blokové schéma decimace signálu.

1.4 Interpolace signálu

Při nadvzorkování signálu aliasing nehrozí, ale vzniká zde zkreslení – vznik nežádoucích obrazů signálu v kmitočtové doméně kolem násobků nové vzorkovací frekvence, což bylo vysvětleno v oddílu 1.2. Proto musí být po nadvzorkování aplikován rekonstrukční filtr. Tento proces se nazývá interpolace [1].

Rekonstrukční filtr bývá zpravidla typu dolní propust s mezním kmitočtem $f_{vzn}/2$, kde f_{vzn} je nový vzorkovací kmitočet nadvzorkovaného signálu. V časové doméně jsou nulové vzorky signálu vložené během nadvzorkování interpolovány na hodnoty odpovídající původnímu signálu. Díky tomu je kmitočtové spektrum interpolovaného signálu shodné s kmitočtovým spektrem původního signálu.

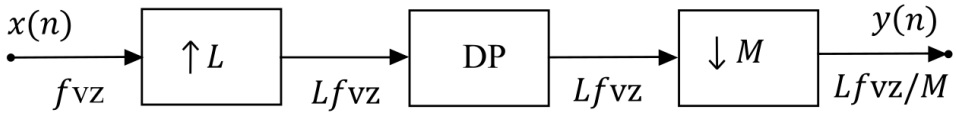


Obrázek 1.5 Blokové schéma interpolace signálu.

1.5 Převzorkování signálu v poměru racionálního čísla

Z definic v oddílech 1.1 a 1.2 je zřejmé, že doposud zmíněné operace se dají použít pouze pro převzorkování signálu v celočíselném poměru. Převzorkování v poměru racionálního čísla L/M je provedeno pomocí interpolace a následné decimace [3].

Takto zvolený postup má za následek použití antialiasingového a interpolačního filtru ve dvou krocích bezprostředně za sebou. Pokud jsou oba filtry typu dolní propust, je možné nahradit je pouze jedním z nich. Je zvolen ten filtr, který má nižší mezní kmitočet.



Obrázek 1.6 Blokové schéma převzorkování signálu v poměru racionálního čísla L/M, tj. interpolace a následná decimace

Je nutno podotknout, že je obecně vhodné, aby celá čísla L a M byla co nejmenší. S jejich růstem úměrně roste i náročnost na strmost použitých filtrů. Toho lze dosáhnout nalezením nejmenšího společného dělitele a následným podělením obou činitelů. Dále lze náročnost snížit postupnou konverzí. Tato metoda spočívá v rozložení faktoru L a M na prvočinitele a provedením několika interpolací a decimací po sobě. Obecně se jako první provádí interpolace. Pokud by byla prvně provedena decimace, pokles vzorkovací frekvence by mohl způsobit nedodržení Nyquistova teorému a nevratnou ztrátu dat.

1.5.1 Klasická postupná konverze

Dle klasické konvence je postupná konverze provedena následovně. Nejprve jsou oba faktory (L a M) rozloženy na prvočinitele. Následně se provede takový počet interpolací, jaký je počet prvočinitelů faktoru L a poté se provede takový počet decimací, jaký je počet prvočinitelů M. Jako faktory jednotlivých interpolací a decimací jsou použity odpovídající prvočinitele vypočtené z faktoru L a M.

Z předchozího textu plyne, že je možné vynechat jeden číslicový filtr v místě, kde za poslední interpolací následuje první decimace.

Tato metoda je jednoduchá na praktickou implementaci, avšak nese riziko vysoké výpočetní náročnosti, jelikož vektor nesoucí vzorky signálu může být po posledním nadvzorkování datově objemný. Tento jev se výrazně projeví u konverze audio souborů s velkými faktory L nebo M a může mít za následek nepřijatelně pomalé provedení samotné konverze.

1.5.2 Alternativní postupná konverze

Alternativní verze postupné konverze je v základu stejná jako klasická postupná konverze. Vyznačuje se jednou podstatnou změnou. Místo provedení všech interpolací před všemi decimacemi se interpolace a decimace střídají.

Opět je možné vynechat jeden číslicový filtr za každou interpolací, za kterou následuje decimace, dle dříve zmíněných pravidel.

Tato metoda na rozdíl od klasické metody efektivně šetří paměť, ale při jejím použití vzniká jiný problém. Je nutné správně zvolit pořadí toho, jak jsou všechny jednotlivé interpolace i decimace za sebou. Při nesprávně zvoleném pořadí může nastat situace, kdy nebude v daném kroku dodržen Nyquistův teorém a část spektra původního signálu bude nevratně ztracena. Tomu lze zamezit provedením decimací až ke konci algoritmu, nebo implementací dedikované funkce, která vypočítá vhodné pořadí. Je ale nutno podotknout, že taková funkce může zvýšit náročnost celkového výpočtu.

1.6 Zpoždění signálu

Vzhledem k vlastnostem antialiasingových a rekonstrukčních filtrů typu FIR (detailně se jimi bude zabývat oddíl 2), které budou použity pro realizaci konverze vzorkovacího kmitočtu dle teorie diskutované v oddílu 1.5, vzniká zpoždění výstupního (převzorkovaného) signálu vůči vstupnímu signálu.

Pro správné vyhodnocení chyby vzniklé převzorkováním viz oddíl 5, je nutné toto zpoždění umět vypočítat a kompenzovat. Nestačí však pouze vypočítat skupinové zpoždění dle rovnice (2.5), protože zpoždění způsobené konkrétním FIR filtrem je navíc ovlivněno změnami aktuální vzorkovací frekvence [11]. Zpoždění způsobené FIR filtrem během konverze vzorkovacího kmitočtu lze vypočítat pomocí rovnice

$$z = \tau \cdot \frac{f_{\mathrm{vzn}}}{f_{\mathrm{a}}}, \tag{1.5}$$

kde τ je skupinové zpoždění definované dle (2.5), f_{vzn} je nový vzorkovací kmitočet, tj. vzorkovací kmitočet výstupního signálu po všech interpolacích a decimacích a f_{a} je aktuální vzorkovací kmitočet, který se odvíjí od aktuální pozice filtru v řetězci zpracování.

Toto zpoždění je nutné vypočítat zvlášť pro každý filtr typu FIR, který byl během procesu konverze použit. Celkové zpoždění způsobené všemi FIR filtry během procesu konverze je definováno vztahem

$$z_c = \sum_{i=1}^{n} z_i, \tag{1.6}$$

kde n je celkový počet všech použitých filtrů.

Je zřejmé, že celkového zpoždění z_c, může nabývat libovolných reálných hodnot. Kompenzaci vzniklého zpoždění je ale možné provést pouze tehdy, jeli z_c celé číslo. Samotná kompenzace je realizována posunem výstupního signálu o takový počet vzorků, jaký odpovídá hodnotě z_c. Indexy jednotlivých vzorků jsou zákonitě pouze celá čísla. Tuto problematiku však nelze řešit pouhým zaokrouhlením z_c, jelikož takový postup by mohl vést k nepravdivým výsledkům během vyhodnocení chyby, vzniklé převzorkováním viz oddíl 5. Jedinou vyhovující variantou je tedy určení délky použitých číslicových filtrů typu FIR tak, aby celkové zpoždění z_c bylo vždy celé číslo. Problematikou implementace zmíněné varianty se bude dále zabývat kapitola 4.

2. FIR FILTRY

FIR filtry jsou číslicové filtry s konečnou impulsní charakteristikou (finite impulse response). Jde o nejvíce využívaný typ filtru při návrhu multitaktních systémů. V této kapitole budou uvedeny základní vlastnosti a vztahy podstatné pro tuto bakalářkou práci.

2.1 Vlastnosti filtrů typu FIR

Impulzní odezva FIR filtru má tvar

$$h(n) = \sum_{i=0}^{N-1} h_i \cdot \delta(n-i), \tag{2.1}$$

kde $i = 0, 1, 2, ..., N-1$ a $\delta(n)$ je Diracův impuls definovaný jako

$$\delta(n) = \begin{cases} 1, & \text{pro } n = 0 \\ 0, & \text{jinak} \end{cases}. \tag{2.2}$$

Odezvu FIR filtru získáme konvolucí vstupního signálu $x(n)$ s jeho impulsní odezvou $h(n)$ dle vztahu

$$y(n) = h(n) * x(n) = \sum_{i=0}^{N-1} h_i \cdot x(n-i). \tag{2.3}$$

Přenosovou funkci filtru typu FIR získáme v $\mathcal{Z}$ transformaci dle následujícího vztahu

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N-1} b_i z^i}{z^{N-1}}, \tag{2.4}$$

kde N je délka impulsní charakteristiky a b_i jsou koeficienty filtru.

Filtr typu FIR s přenosovou funkcí dle (2.4) je vždy kauzální, protože ve jmenovateli je z s exponentem $N-1$ a vždy stabilní [4]. Způsobuje skupinové zpoždění vyjádřené vztahem

$$\tau = \frac{N-1}{2}, \tag{2.5}$$

kde N je přirozené číslo udávající počet vzorků impulsní odezvy FIR filtru. Skupinové zpoždění τ je definované jako záporná derivace fázové kmitočtové charakteristiky podle kmitočtu. Toto skupinové zpoždění je konstantní a každý vzorek signálu je zpožděn o stejný interval. Pokud je požadována realizace přenosové funkce se strmým přechodovým pásmem, je nutné použít filtr FIR s velmi vysokým řádem $N-1$. Důsledkem je veliké zpoždění signálu a vyšší náročnost na místo v paměti [4].

2.2 Metoda váhovacích oken

Metoda váhovacích oken je rozšířená metoda pro návrh filtrů typu FIR, kterou ve svém výpočetním algoritmu používá funkce fir1 softwaru Matlab. Funkce fir1 nejprve vypočítá koeficienty filtru pomocí metody nejmenších čtverců a poté vyhladí jeho impulsní odezvu právě pomocí zvoleného váhovacího okna [6]. Tato funkce bude hojně využita v následné implementaci, a proto se teorii metody váhovacích oken bude věnovat tato kapitola. Pro její pochopení je nejprve nutné zmínit vlastnosti ideálních filtrů.

Kmitočtová charakteristika ideálního filtru obsahuje skokovou změnu z propustného do nepropustného pásma, tj. neexistuje žádný přechodný stav mezi těmito pásmy. Impulsní charakteristiku ideálního filtru je možné vypočítat pomocí Fourierovy řady, protože koeficienty Fourierovy řady jsou zároveň koeficienty hledané impulsní charakteristiky. Získaná impulsní charakteristika je nekonečná a nekauzální, což je v rozporu s vlastnostmi FIR filtrů. Abychom získali charakteristiky vhodné pro číslicový FIR filtr, je nutné zachovat pouze konečný počet vzorků. Pokud je vyžadováno porovnání signálu před a po filtraci pomocí filtru typu FIR v časové oblasti, je navíc potřeba vzorky zpozdit dle definice skupinového zpoždění.

Těmito zásahy výrazně ovlivníme jak fázi, tak modul kmitočtové charakteristiky, obzvláště pak přechodové pásmo. Zpoždění impulsní charakteristiky má za následek změnu fázové kmitočtové charakteristiky, která již není nulová, ale lineárně klesající. Ponechání konečného počtu vzorků má za následek změnu tvaru kmitočtové charakteristiky. Výpočtu impulsní charakteristiky číslicového filtru typu FIR je docíleno pomocí součinu ideální impulsní charakteristiky a vybraného váhovacího okna [4].

Váhovacích oken existuje mnoho typů a jejich výběr má zásadní vliv na výslednou kmitočtovou charakteristiku navrhovaného FIR filtru. V následujícím textu bude zmíněno několik základních typů:

Pravoúhlé okno je nejjednodušším typem váhovacího okna. Zachovává daný počet vzorků a zbylé vzorky vynuluje. Je definováno vztahem

$$w[nT] = \begin{cases} 1, & pro\ n\ \in\ \langle 0,\ \ N-1\rangle \\ 0, & \text{jinak} \end{cases}. \tag{2.6}$$

Hannovo okno

$$w[nT] = \begin{cases} 0{,}5\ \left[1-\cos\left(\dfrac{2\pi n}{N-1}\right)\right], & \text{pro}\ n\ \in\ \langle 0,\ \ N-1\rangle \\ 0, & \text{jinak} \end{cases}. \tag{2.7}$$

Hammingovo okno

$$w[nT] = \begin{cases} 0{,}54\ -0{,}46\cos\left(\dfrac{2\pi n}{N-1}\right), & \text{pro}\ n\ \in\ \langle 0,\ \ N-1\rangle \\ 0, & \text{jinak} \end{cases}. \tag{2.8}$$

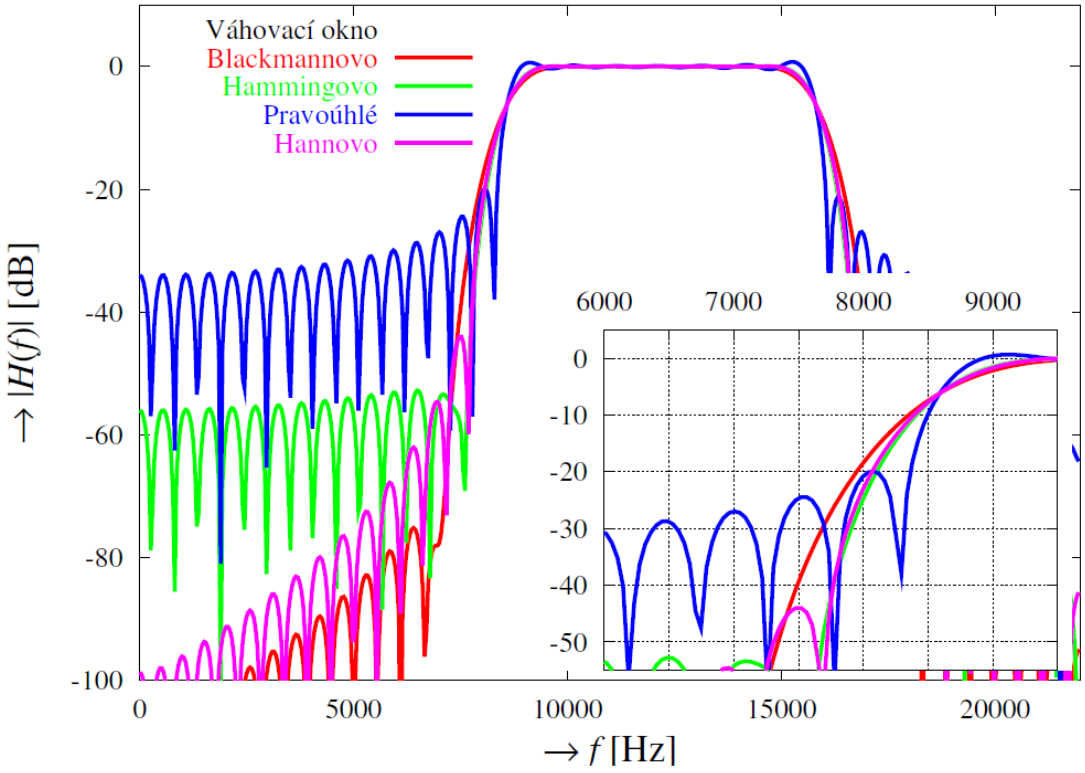
Blackmanovo okno

$$w[nT] = \begin{cases} 0{,}42 - 0{,}5\cos\left(\frac{2\pi n}{N-1}\right) + 0{,}08\cos\left(\frac{4\pi n}{N-1}\right), \text{pro } n \in \langle 0,\ N-1\rangle \\ 0, \qquad\qquad \text{jinak} \end{cases}. \tag{2.9}$$

Kaiserovo okno

$$w[nT] = \begin{cases} \dfrac{I_0\left\{\beta\sqrt{\left(\frac{N-1}{2}\right)^2 - \left(n - \frac{N-1}{2}\right)^2}\right\}}{I_0\left\{\beta\frac{N-1}{2}\right\}}, \quad \text{pro } n \in \langle 0,\ N-1\rangle \\ 0, \qquad\qquad\qquad\qquad\qquad\qquad \text{jinak} \end{cases}. \tag{2.10}$$

kde I_0 představuje Besselovu funkci nultého řádu prvního druhu a β je parametr určující vlastnosti okna. Správnou volbou tohoto parametru je možné Kaiserovým oknem simulovat jakýkoliv jiný typ váhovacího okna.



Obrázek 2.1 Modul přenosové funkce pro délku impulsní charakteristiky N = 40 při použití různých váhovacích oken. Převzato z [4]

3. METODA GIANT FFT

Giant FFT je vysoce kvalitní metoda konverze vzorkovacího kmitočtu ve frekvenční doméně, která je uskutečněna prostřednictvím jedné rychlé Fourierovy transformace s velmi velkým počtem vzorků a následné zpětné Fourierovy transformace. Tento úkon byl dříve pro počítače neproveditelný nebo velmi náročný a trval dlouhou dobu, avšak dnešní počítače jsou schopny poměrně rychle vypočítat FFT obsahující stovky milionů bodů a je tedy možné provést konverzi audia o délce cca jedné hodiny v řádu sekund [2].

Značnou výhodou této metody oproti klasické metodě využívající číslicové filtry typu FIR, je nulové zpoždění výstupního signálu, který tak setrvává v perfektní synchronizaci se vstupním signálem. Jedinou nevýhodou této metody je vznik umělé úzkopásmové složky signálu nazývané artefakt na frekvenci Nyquistova limitu. Ten však lze potlačit pomocí tapering funkce $W(k)$ nebo filtrem typu dolní propust, který ale vnese do systému zpoždění. Problematikou artefaktu a jeho potlačení se bude dále zabývat oddíl 3.4.

3.1 Obecné shrnutí algoritmu

Metoda Giant FFT lze jednoduše popsat pomocí následujícího algoritmu. Uvažujme, že máme vstupní signál x o vzorkovací frekvenci f_{vz} a požadujeme jeho konverzi na výstupní signál x' s novým vzorkovacím kmitočtem f_{vzn}.

1. Jsou vybrány vhodné délky N a N' viz oddíl 3.5
2. Vstupní signál o délce N_{in} je doplněn takovým počtem nulových vzorků, aby $N \geq N_{in}$
3. Je provedena FFT vstupního signálu x a tím je získáno jeho spektrum X o délce N vzorků
4. Pokud je to žádané, je proveden tapering ve frekvenční doméně pomocí funkce $W(k)$
5. V případě podvzorkování (viz rovnice (3.1)) jsou symetricky odebrány vzorky a v případě nadvzorkování (viz rovnice (3.2)) jsou symetricky přidány nulové vzorky tak, aby nové spektrum X' odpovídalo podmínce $N' = Nf_{vzn}/f_{vz}$
6. Je provedena IFFT spektra X' a tím je získán nový signál x' o délce vzorků N'
7. Nový signál x' je dán do správného měřítka pomocí faktoru L/M
8. Nadbytečné nuly jsou odstraněny a je získán výstupní signál x_{out} (viz rovnice (3.8))
9. Pokud je to žádané, je na Nyquistově limitu aplikován FIR filtr typu dolní propust

V následujících kapitolách budou podrobně vysvětleny důležité kroky a operace zmíněné ve výše uvedeném algoritmu.

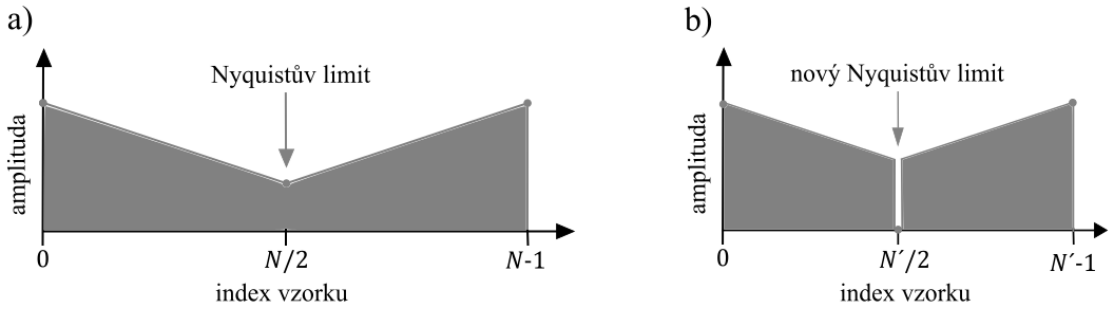
3.2 Podvzorkování zkrácením spektra

Jedná se o operaci snižující počet vzorků v signálu obdobně jako u klasické metody konverze, viz oddíl 1.1.

Na rozdíl od klasické metody, při které je nezbytný FIR filtr, u metody Giant FFT je podvzorkování uskutečněno pouze pomocí zkrácení signálu ve frekvenční doméně v poměru stejném, jako je původní a nová vzorkovací frekvence. Spektrum signálu tak není ovlivněno až do indexu nového Nyquistova limitu. Ekvivalentní operací by byla konvoluce signálu s pravoúhlým oknem, což vysvětluje vznik artefaktu, viz oddíl 3.4. Dále musí platit $N' < N$, kde N je délka původního signálu a N' je délka podvzorkovaného signálu [2]. Prakticky je podvzorkování uskutečněno následovně. Nejprve je nutné provést rychlou Fourierovu transformaci a poté ze získaného spektra odebrat vzorky s indexem vyšším, než je nový Nyquistův limit, který odpovídá nově zvolené vzorkovací frekvenci. Všechny zbylé vzorky zůstanou beze změny zachovány. Podvzorkovaný signál je konstruován dle následující rovnice

$$X'(k) = \begin{cases} X(k), & 0 < k < \frac{N'}{2} \\ 0, & k = \frac{N'}{2} \\ X(N - N' + k), & \frac{N'}{2} < k < N' - 1 \end{cases}, \tag{3.1}$$

kde X je spektrum původního signálu, X' je spektrum podvzorkovaného signálu a k je index spektrálního vzorku.



Obrázek 3.1 Podvzorkování ve frekvenční doméně. a) Spektrum původního signálu b) spektrum podvzorkovaného signálu

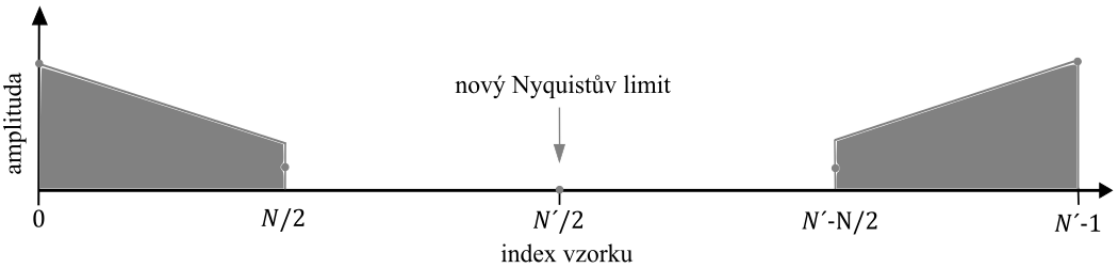
3.3 Nadvzorkování spektra

Jedná se o operaci zvyšující počet vzorků v signálu obdobně jako u klasické metody konverze, viz oddíl 1.2.

Na rozdíl od dříve zmíněné klasické metody je u metody Giant FFT nadvzorkování uskutečněno pouze pomocí prodloužení signálu ve frekvenční doméně v poměru stejném jako je původní a nová vzorkovací frekvence. Spektrum signálu tak není ovlivněno. Prakticky je nejprve nutné provést rychlou Fourierovu transformaci a poté do získaného spektra vložit nulové vzorky o počtu $N' - N$, kde N je délka původního signálu a N' je délka nadvzorkovaného signálu odpovídající nově zvolené vzorkovací frekvenci a platí, že $N' > N$ [2]. Nadvzorkovaný signál je konstruován dle následující rovnice

$$X'(k) = \begin{cases} X(k), & 0 < k < \frac{N}{2} \\ \frac{1}{2}X\left(\frac{N}{2}\right), & k = \frac{N}{2} \\ 0, & \frac{N}{2} < k < N' - \frac{N}{2} \\ \frac{1}{2}X\left(\frac{N}{2}\right), & k = N' - \frac{N}{2} \\ X(k - N' + N), & N' - \frac{N}{2} < k < N' \end{cases}, \tag{3.2}$$

kde X je spektrum původního signálu, X' je spektrum nadvzorkovaného signálu a k je index spektrálního vzorku. Hodnota vzorku na původním Nyquistově limitu je vydělena dvěma a ponechána na svém původním indexu a indexu symetrickém podle indexu nového Nyquistova limitu [7]. V teorii se jedná o přesnou operaci, uvažujeme-li časově a frekvenčně omezený signál. V praxi je ale možné dosáhnout pouze konečné přesnosti [2].



Obrázek 3.2 Nadvzorkování ve frekvenční doméně. Spektrum nadvzorkovaného signálu

3.4 Tapering

Tato část textu se bude zabývat metodou potlačení artefaktu, která je nazývána tapering. Tapering je termín z anglické odborné literatury [2], pro který neexistuje žádný český ekvivalent. Artefakt je nežádoucí uměle vzniklá úzkopásmová složka signálu ve frekvenční doméně, která vznikla následkem náhlého useknutí části spektra signálu na Nyquistově limitu. Toto useknutí ve spektru je ekvivalentní ke konvoluci signálu s pravoúhlým oknem, které je spojováno se vznikem Gibbsova jevu [3]. Popisovaný artefakt je obdobou Gibbsova jevu. Useknutí signálu je způsobeno jak rovnicí (3.1), tak rovnicí (3.2) a tudíž se mu nelze vyhnout. Tento problém vzniká pouze na frekvenci Nyquistova limitu.

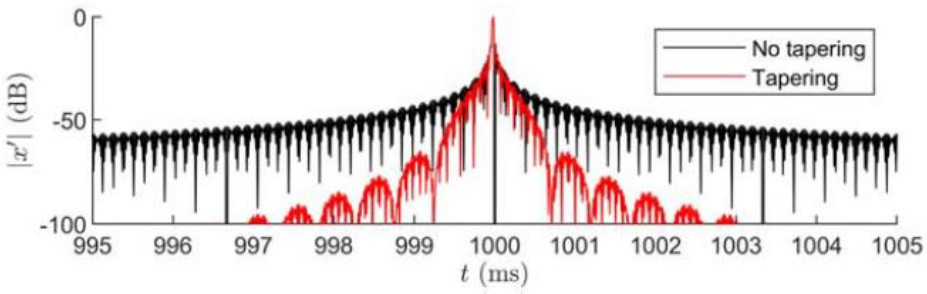
Vhodného taperingu lze dosáhnout použitím váhovací funkce $W(k)$, která obsahuje pouze nezáporné reálné hodnoty a je uskutečněna pomocí poloviny funkce cosinus. Pro spektrum obsahující M vzorků je definována následovně

$$W(k) = \begin{cases} 1, & 0 \le k, k_c \\ \dfrac{1 + \cos\left(\pi \dfrac{k - k_c + 1}{M/2}\right)}{2}, & k_c \le k \le M/2 \\ \dfrac{1 + \cos\left(\pi \dfrac{k_c - k + 1}{M/2}\right)}{2}, & M/2, k \le M/2 + k_c \\ 1, & M/2 + k_c, k, M \end{cases}, \tag{3.3}$$

kde k_c je index na kterém začne tapering. Funkce je navíc symetrická kolem její centrální hodnoty $W(M/2)$, která odpovídá hodnotě na Nyquistově limitu.

Při implementaci je nutné provést bodové komplexní násobení funkce tapering a spektra převzorkovaného signálu $W(k)X'(k)$. Jako vhodné řešení se nabízí násobit pouze ty vzorky, pro které se váhovací funkce nerovná jedné.

Pokud by tapering nebyl dostačující, je možné dodatečně použít FIR filtr typu dolní propust. Výstupní signál bude oproti vstupnímu signálu zpožděn, viz oddíl 2.1.



Obrázek 3.3 Impuls po konverzi vzorkovacího kmitočtu z 44,1 na 96 kHz bez taperingu a s taperingem. Převzato z [2]

3.5 Výběr vstupních a výstupních délek pro FFT

Velmi důležitou úlohu pro co nejefektivnější funkci popisované metody hrají vstupní a výstupní délky signálu, který je zpracováván pomocí FFT. Zvláště důležitá jsou délky vstupní, jelikož přímo ovlivňují náročnost a rychlost provedení samotné FFT. Nejvhodnější délky vstupního signálu jsou takové, které po rozkladu na prvočinitele obsahují malá prvočísla, například 2,3,5 a 7 [2].

Mějme vstupní signál o počtu vzorků N_{in}. K takovému signálu je nutné přidat vhodný počet nulových vzorků, čímž získáme signál o délce N, pro kterou platí $N > N_{\text{in}}$. Poté po provedení konverze vzorkovací frekvence získáme výstupní signál o nové délce N'. Signál o délce N a výstupní signál o délce N' musí být v poměru původní vzorkovací frekvence f_{vz} a nové vzorkovací frekvence f_{vzn} a pro tento poměr platí rovnice

$$\frac{f_{\text{vzn}}}{f_{\text{vz}}} = \frac{L}{M}, \tag{3.4}$$

kde L a M jsou celá čísla bez společného dělitele a platí pro ně

$$N = MP, \tag{3.5}$$

a

$$N' = LP, \tag{3.6}$$

kde P je celé číslo, které je vhodné zvolit tak, aby číslo N bylo co nejblíže číslu N_{in}. Z toho plyne následující rovnice

$$P \geq P_{\text{in}} = 2\left\lceil \frac{N_{\text{in}}}{2M} \right\rceil. \tag{3.7}$$

V případě, že vypočtené číslo P_{in} obsahuje po rozkladu na prvočinitele velké prvočíslo, je vhodnější použít P, které po rozkladu obsahuje pouze malá prvočísla a zároveň platí, že $P > P_{\text{in}}$. Problematikou toho, jak velké prvočíslo je považováno za nepřijatelné, se tento text nebude dále zabývat. Nutno však podotknout, že při velkém rozdílu mezi M a M_{in} vede k ještě většímu rozdílu mezi N a N_{in}, což může rovněž vést ke zvýšení náročnosti výpočtu FFT.

Na závěr mohou být přebytečné nulové vzorky signálu o délce N' odstraněny. Délka N_{out} odpovídající délce výstupního signálu je vypočítána dle rovnice

$$N_{\text{out}} = \left\lceil \frac{f_{\text{vzn}}}{f_{\text{vz}}} N_{\text{in}} \right\rceil, \tag{3.8}$$

z toho plyne, že bude odstraněno $N - N_{\text{out}}$ přebytečných nulových vzorků, které musí být odstraněny.

4. KONVERZE VZORKOVACÍHO KMITOČTU V PROSTŘEDÍ MATLAB

V následující kapitole bude popsána praktická implementace metod konverze vzorkovacího kmitočtu v prostředí softwaru Matlab. Zejména bude uplatněna teorie popsaná v oddílech 1.5.1, 1.5.2 a 3. Cílem implementace je tvorba funkčních kódu, které budou schopny zcela obecné konverze mezi různými vzorkovacími kmitočty. Praktické testování proběhne na vzorkovacích kmitočtech vhodných pro audio soubory [9].

Pro jednoduchou ukázku všech implementovaných metod konverze byl naprogramován skript `resample_demo`, ve kterém uživatel zvolí libovolný signál (obsažený v přiložené databázi testovacích signálu) ke konverzi. Jednotlivé testovací signály jsou uloženy buďto jako audio soubory .wav nebo jako datové soubory Matlabu .mat. Soubor je načten pomocí funkce `loadsignal` a uložen jako vektor `x` a jeho vzorkovací kmitočet je uložen jako hodnota `fs`. Dále je uživatelem zvolen nový vzorkovací kmitočet výstupního signálu `fsn`. Nakonec je vybrána jedna z šesti metod konverze, které budou popsány v následujících oddílech této kapitoly. Skript navíc umožňuje vyobrazení grafů vstupních a výstupních signálů jak v časové, tak ve frekvenční doméně pomocí funkce `dispgraphs`, ukládání výstupních signálů ve formátu .wav i .mat, a změnu délky FIR filtru u funkcích konverze, kde to má smysl.

4.1 Implementace klasické postupné metody konverze

Následující oddíl se bude věnovat praktické implementaci klasické postupné metody konverze dle oddílu 1.5.1. Metoda byla naprogramována ve formě dvou funkcí nazvaných `resample_classic_fixN` a `resample_classic_varN`. Hlavní rozdíl mezi nimi spočívá v tom, že u funkce `resample_classic_fixN` je předem specifikována délka všech FIR filtrů pomocí proměnné `N` (s možností volby hodnoty `N` uživatelem ve skriptu `resample_demo`) a u funkce `resample_classic_varN` je délka FIR filtrů `N` vypočítána uvnitř funkce tak, aby bylo dosaženo nulového zpoždění výstupního signálu oproti vstupnímu bez nutnosti zaokrouhlení viz oddíl 1.6. Tato kompenzace zpoždění je nutná pouze kvůli následnému kvalitativnímu vyhodnocení dané funkce konverze, viz oddíl 5.

Pro jednoduchost a přehlednost textu bude v následujících odstavcích popsána funkce `resample_classic_fixN` a následně budou vysvětleny rozdíly oproti funkci `resample_classic_varN`.

4.1.1 Resample_classic_fixN

V prvním kroku jsou vypočítány všechny důležité parametry. Vstupními proměnnými jsou vektor `x`, který reprezentuje všechny vzorky původního signálu, původní vzorkovací

kmitočet `fs` a nový vzorkovací kmitočet `fsn`. Nejprve je proveden výpočet nejmenšího společného dělitele původního a nového vzorkovacího kmitočtu pomocí funkce Matlabu `gcn` a získaná hodnota je uložena do proměnné `GCN`. Poté je proveden výpočet faktoru nadvzorkování `L` jako `fsn/GCN` a faktoru podvzorkování `M` jako `fs/GCN`. Dále jsou oba tyto faktory rozloženy na prvočinitele pomocí funkce Matlabu `factor` a jsou uloženy do proměnných `fL` a `fM`. Následně je alokována paměť pro vektor `fa`, který bude sloužit pro ukládání aktuálního vzorkovacího kmitočtu v každém kroku algoritmu, kdy je aplikován FIR filtr. Jako první je do něj uložena proměnná `fs`. Poté je vektor `x` uložen do proměnné `yvarL`, která slouží jako nosič signálu během části algoritmu pro nadvzorkování. Díky tomu je vektor `x` zachován ve své původní podobě a je možné ho použít pro uchování dat na výstupu této funkce. Na závěr je zkontrolováno, jestli uživatel ve skriptu `resample_demo` zadal proměnnou `N`. Pokud ne, je základně nastavena na hodnotu 99. Hodnota 99 byla zvolena na základě empirického měření rychlosti výpočtu a vzniklé chyby ve výpočtu pro různé hodnoty délky všech filtrů.

Následně je provedena hlavní část algoritmu, pomocí dvou cyklů. V prvním se provedou všechny interpolace, ve druhém se provedou všechny decimace. Před každým cyklem je zkontrolováno, jestli se náhodou proměnná `L` (`M`) nerovná 1. Pokud ano, k interpolaci (decimaci) nedochází, vektor `x` (`yvarM`) je uložen jako vektor `yvarM` (`y`) a celý cyklus je přeskočen.

```
if L == 1
    yvarM = x;
else
    for i = 1:length(fL)    % interpolation cycle
        yvar = upsample(yvarL,fL(i));
        aif = fir1(N,1/fL(i),"low");
        yvarL = filter(aif,1/fL(i),yvar);
        fa(i+1) = (fa(i)*fL(i));    % delay count
    end
    yvarM = yvarL;
end
if M == 1
    y = yvarM;
else
    if L == 1
        i = 0;
        faM = fs;
    else
        faM = fa(i);
    end
    for j = 1:length(fM)    % decimation cycle
        fa(i+j+1) = faM;    % delay count
        aaf = fir1(N,1/fM(j),"low");
        yvar = filter(aaf,1,yvarM);
        yvarM = downsample(yvar,fM(j));
        faM = (fa(i+j+1)/fM(j));
    end
    y = yvarM;
    fa = fa(2:end);
end
```

Uvnitř cyklů dále vystupuje proměnná `yvar`, sloužící k ukládání vektoru převzorkovávaného signálu v rámci jednoho cyklu, dříve popsaný vektor `fa` a pomocná proměnná `faM`, uplatněná pro přenos hodnoty fa mezi dvěma decimačními cykly.

Během interpolačního cyklu je prvně uložen nadvzorkovaný vektor podle aktuálního faktoru nadvzorkování `fL` do proměnné `yvar`. Nadvzorkování je uskutečněno pomocí jednoduché funkce `upsample`, která funguje na vektorové bázi podle rovnice (1.1). Dále je vygenerován vektor `aif` reprezentující rekonstrukční filtr (v anglické literatuře též anti-imagining filter [1]) dle části 1.4 a následně je jím nadvzorkovaný signál filtrován. Na závěr cyklu je ještě vypočítán aktuální vzorkovací kmitočet a uložen do vektoru `fa`. Cyklus se opakuje tak dlouho jaká je délka vektoru `fL`.

Během decimačního cyklu se prvně uloží aktuální vzorkovací kmitočet, který momentálně reprezentuje proměnná `faM` do vektoru `fa`. Poté je vygenerován vektor `aaf` reprezentující anti-aliasing filtr dle části 1.3 a následně je signál filtrován. Dále je signál podvzorkován pomocí jednoduché funkce downsample podle rovnice (1.3) a aktuálního faktoru podvzorkování `fM`. Na závěr je vypočítán nový aktuální vzorkovací kmitočet a je uložen do proměnné `faM`. Cyklus se opakuje tak dlouho, jaká je délka vektoru `fM`.

Na závěr hlavní části algoritmu je výsledný převzorkovaný vektor uložen do výstupní proměnné `y` a z vektoru `fa` je odebrána první hodnota, která reprezentuje původní vzorkovací kmitočet a je pro účely tohoto vektoru nadbytečná.

V poslední části funkce je vypočítáno zpoždění výstupního vektoru oproti vstupnímu pomocí vektoru `fa` a tato hodnota je uložena do proměnné `d`. Výstupní signál je posunut tak, aby výsledné zpoždění bylo nulové. Z podstaty věci může nastat situace, kdy proměnná `d` nebude celé číslo, to ale nepřipadá v úvahu, jelikož indexy jednotlivých vzorků signálu jsou vždy celočíselné. V tomto postupu je nutné proměnnou `d` vždy zaokrouhlit. Tím ale potenciálně vzniká chyba v nulování zpoždění a tato funkce tedy není vhodná pro následné porovnávání vstupního a výstupního signálu.

4.1.2 Resample_classic_varN

Jedná se v základu o stejnou funkci, jaká byla popsána v oddíle 4.1.1, pouze s tím rozdílem, že hodnota délky filtrů `N` není zadávána, ale je vypočítána tak, aby zpoždění výstupního signálu `d` bylo vždy celočíselné, viz oddíl 1.6.

Toho je dosaženo tak, že vektor `fa`, je vypočítán ještě před hlavním algoritmem. Následně je vypočtena suma obrácené hodnoty poměrů všech hodnot vektoru `fa` a nového vzorkovacího kmitočtu `fsn` dle rovnice (1.6). Poté je proveden cyklus, který nalezne všechny vhodné délky filtrů a jejich odpovídající zpoždění výstupního signálu v předem definovaném rozmezí proměnnou `n`.

Pokud se v daném rozmezí nenachází žádná vhodná délka filtrů `N`, je v příkazovém řádku zobrazena zpráva: filter delay is not integer! Pokud je vhodných `N` nalezeno více, je vybrána nejmenší hodnota `N` a příslušná nejmenší hodnota `d`.

Tato funkce je uzpůsobena následnému porovnávání vstupního a výstupního signálu.

4.2 Implementace alternativní postupné metody konverze

Následující oddíl se bude věnovat praktické implementaci alternativní postupné metody konverze dle oddílu 1.5.2. Metoda byla naprogramována ve formě dvou funkcí nazvaných `resample_alternating_fixN` a `resample_alternating_varN`. Hlavní rozdíl mezi nimi spočívá v tom, že u funkce `resample_alternating_fixN` je předem specifikována délka všech FIR filtrů pomocí proměnné `N` (s možností volby hodnoty `N` uživatelem ve skriptu `resample_demo`) a u funkce `resample_alternating_varN` je délka FIR filtrů `N` vypočítána uvnitř funkce tak, aby bylo dosaženo nulového zpoždění výstupního signálu oproti vstupnímu bez nutnosti zaokrouhlení, viz oddíl 1.6.

Pro jednoduchost a přehlednost textu bude v následujících odstavcích popsána funkce `resample_alternating_fixN` a následně budou vysvětleny rozdíly oproti funkci `resample_alternating_varN`.

4.2.1 Resample_alternating_fixN

Na začátku jsou vypočítány všechny důležité parametry, které jsou stejné jako parametry zavedené na začátku oddílu 4.1.1 s tím rozdílem, že je navíc vytvořen vektor `fLM`, ve kterém jsou uloženy všechny faktory nadvzorkování i podvzorkování. Ty jsou seřazeny tak, aby po faktoru nadvzorkování následoval faktor podvzorkování, pokud je to možné. A navíc, aby faktory podvzorkování měly co nejvyšší indexy. Tím je účinně zamezeno ztrátě dat v důsledku příliš nízké aktuální vzorkovací frekvence pro všechny běžné konverze vzorkovacího kmitočtu audio signálů. Dále je definován vektor `ud`, ve kterém se nachází 1 pokud se ve vektoru `fLM` na stejném indexu nachází faktor nadvzorkování a 0, pokud se ve vektoru `fLM` na stejném indexu nachází faktor podvzorkování. Dále jsou s obou vektorů odstraněny všechny nadbytečné hodnoty, tj. hodnoty 0 a 1 ve vektoru `fLM` a hodnoty na odpovídajících indexech ve vektoru `ud`. Poté je vektor `x` uložen do proměnné `yvarLM`, která slouží jako nosič signálu během hlavní části algoritmu. Díky tomu je vektor `x` zachován ve své původní podobě a je možné ho použít pro uchování dat na výstupu této funkce. Na závěr je zkontrolováno, jestli uživatel v demo skriptu `resample_demo` zadal proměnou `N`. Pokud ne, je základně nastavena na hodnotu 99.

```
for i = 1:length(fLM)    % resampling cycle
    if ud(i) == 1
        yvar = upsample(yvarLM,fLM(i)); % interpolation
        aif = fir1(N,1/fLM(i),"low");
        yvarLM = filter(aif,1/fLM(i),yvar);
        fa(i+1) = (fa2*fLM(i)); % delay count
        fa2 = fa(i+1);
    else
        fa(i+1) = fa2;  % delay count
        aaf = fir1(N,1/fLM(i),"low");   % decimation
        yvar = filter(aaf,1,yvarLM);
        yvarLM = downsample(yvar,fLM(i));
        fa2 = (fa(i)/fLM(i));
    end
end

y = yvarLM;
fa = fa(2:end); % removing unwanted values from fa array
```

V hlavní části algoritmu je proveden jeden cyklus tolikrát, kolik je délka vektoru `fLM`. V každém kroku cyklu je prvně ověřeno, jakému číslu se rovná hodnota na aktuálním indexu pole `ud`. Pokud se rovná 1, je provedena interpolace, pokud ne je provedena decimace. Interpolace i decimace je provedena obdobně jako v oddílu 4.1.1

Na závěr algoritmu je proveden výpočet zpoždění signálu `d` a jeho kompenzace stejně jako v oddílu 4.1.1

4.2.2 resample_alternating_varN

Jedná se o podobnou funkci, jaká byla popsána v oddíle 4.2.1, ale s následujícími rozdíly. Hodnota délky filtrů `N` není zadávána, ale je vypočítána tak, aby zpoždění výstupního signálu `d` bylo vždy celočíselné, viz oddíl 1.6 a vektor `fLM` je seřazen tak, aby žádná hodnota vektoru `fa/2` nebyla nižší než hodnota Nyquistova limitu, který je určen jako menší z obou vzorkovacích kmitočtů `fs`, `fsn`.

Toho je dosaženo pomocí cyklu `while`, který na začátku každého kroku vypočítá hodnotu `fa` za využití vektoru `ud`. Pokud je `ud` 1, původní kmitočet je vynásoben hodnotou na aktuálním indexu vektoru `fLM` pokud ne, tak je původní kmitočet vydělen hodnotou na aktuálním indexu vektoru `fLM`. Dále je ověřena podmínka, jestli je aktuální hodnota `fa/2` větší než minimum z hodnot `fs` a `fsn`. Pokud ano, přejde se na další krok cyklu. Pokud ne a `ud` je 1, je aktuální hodnota `fLM` a `ud` prohozena s předešlou hodnotou těchto vektorů a cyklus je o jeden krok vrácen. Pokud ne a `ud` je 0, je aktuální hodnota `fLM` a `ud` prohozena s následující hodnotou těchto vektorů a aktuální krok cyklu je opakován. To se děje, dokud není vektor `fLM` správně seřazen. Navíc je v průběhu vypočítán vektor `fa`, který slouží k výpočtu délky filtrů `N` a zpoždění výstupního signálu `d`, stejně jako v oddíle 4.1.2.

Tato funkce je vhodná pro následné porovnávání vstupního a výstupního signálu.

4.3 Implementace metody Giant FFT

Následující oddíl se bude věnovat praktické implementaci metody konverze nazývané Giant FFT dle oddílu 3. Metoda byla naprogramována ve formě funkce nazvané `resample_GFFT`. Z teorie je zřejmé, že funkce se velmi liší od výše zmíněných funkcí postupné konverze popsaných v oddílech 4.1.1, 4.1.2, 4.2.1, 4.2.2. Proto bude od začátku do konce podrobně popsána.

4.3.1 resample_GFFT

V prvním kroku jsou vypočítány všechny důležité parametry. Vstupními proměnnými jsou vektor `x`, který reprezentuje všechny vzorky původního signálu, původní vzorkovací kmitočet `fs` a nový vzorkovací kmitočet `fsn`. Nejprve je proveden výpočet nejmenšího společného dělitele původního a nového vzorkovacího kmitočtu pomocí funkce Matlabu `gcn` a získaná hodnota je uložena do proměnné `GCN` stejně jako v oddíle 4.1.1. Poté je proveden výpočet faktoru nadvzorkování `P` jako `fsn/GCN` a faktoru podvzorkování `Q` jako `fs/GCN`.

Dále je stanovena délka vektoru `x` pomocí funkce matlabu `length` a uložena do proměnné `Nin`. Poté je vypočítán parametr `Mmin` dle rovnice (3.7) a parametr `NMmin` podle rovnice (3.5) a jeho hodnota je rozložena na prvočinitele. Následně je ověřeno, jestli je některý prvočinitel větší než 9. Hodnota 9 byla zvolena na základě empirického měření rychlosti výpočtu pro různé maximální povolené hodnoty prvočinitelů. Pokud ne, hodnota `Mmin` se uloží jako proměnná `M`. Pokud ano, je nalezena nová hodnota `M` jako mocnina čísla 2. Mocnina je určena pomocí funkce Matlabu `nextpow2`, která vrátí nejmenší exponent mocniny dvou, při kterém je výsledné číslo větší než hodnota `Mmin`. Tím je dosaženo, rychlejšího výpočtu během následujících kroků, a navíc je zaručeno, že délka `N` vypočítána podle rovnice (3.5) bude vhodná pro funkci matlabu `fft` a délka `Nn` vypočítána dle (3.4) bude vhodná pro funkci matlabu `ifft`.

Ve druhém kroku je vstupní signál `x` doplněn takovým počtem nul, aby jeho výsledná délka odpovídala hodnotě proměnné `N`. Toho je dosaženo pomocí funkce `zt`, která je vypočítána jako rozdíl mezi hodnotami `Nin` a `N` a takový počet nul je vložen na konec vektoru `x`. Následně je provedena rychlá Fourierova transformace vektoru `x` pomocí funkce Matlabu `fft`, navíc je výsledná hodnota ještě podělena odmocninou z `N` (Z důvodu zachování energie signálu. Což není nativně implementováno ve funkci Matlabu `fft`) a uložena do proměnné `X`.

Ve třetím kroku je konstruován taperingový vektor `W` podle rovnice (3.3). Proměnná `kc` je definována jako 5 % z délky `Nn`. Tím je dosaženo toho, že tapering bude aplikován na prostředních 10 % spektra výstupního vektoru `Y`.

Ve čtvrtém kroku proběhne nadvzorkování nebo podvzorkování. Prvně je vypočítána proměnná `zf` jako rozdíl `Nn` a `N` a vektor `azf`, které má délku o velikosti `zf` a obsahuje

pouze 0. Pokud je proměnná `zf` větší než 0, tj. kladná, provede se nadvzorkování dle rovnice (3.2). Vektor `X` je rozpůlen podle proměnné `nyql`, která je vypočítána jako `N/2`, doprostřed je vložen vektor `azf` a výsledný vektor je uložen jako `Y`. Pokud je proměnná `zf` menší než 0, tj. záporná, je provedeno podvzorkování dle rovnice (3.1). Vektor `X` je rozdělen podle proměnné `nyqln`, která je vypočítána jako `Nn/2`, doprostřed je vložena jedna 0 a výsledný vektor je uložen jako `Y`. V tomto případě je část hodnot vektoru `X` nevratně ztracena. Nakonec je proveden tapering vynásobením hodnot vektoru `Y` a vektoru `W` na indexech odpovídajících prostředním 10 % délky vektoru `Y` značené `Nn`.

```
% upsampling or downsampling
zf = Nn - N; % number of zeros that must be inserted
azf = zeros(1,zf);
nyqln = Nn/2; % index of middle value of Y
if zf > 0 % upsampling
    nyql = N/2; % index of middle value of X
    Y = [X(1:end-nyql),azf,X(1+nyql:end)]; % Adding Nn - N zeroes in the middle
of signal
    Y(nyql) = X(end-nyql)./2; % dividing value at old nyquist limit by 2
    Y(end-nyql) = X(1+nyql)./2;
else % downsampling
    Y = [X(1:nyqln-1),0,X(end-nyqln+1:end)]; % inserting 0 at new nyquist limit
and truncating the signal
end
Y(nyqln+1-kc:nyqln+kc) = W.*Y(nyqln+1-kc:nyqln+kc); % tapering
```

V pátém kroku je vypočítám výstupní vektor `y` pomocí funkce Matlabu `ifft`, navíc vynásobené odmocninou z `N`. Z vektoru jsou odebrány všechny komplexní složky, aby obsahoval pouze reálná čísla, a nakonec je dán do správného měřítka vydělením všech jeho hodnot proměnou `Q` a vynásobením proměnou `P`.

V posledním kroku jsou odebrány všechny přebytečné nuly nacházející se na konci vektoru `y`. Hledaná délka vektoru `y` po odstranění přebytečných 0 je vypočítána podle rovnice (3.8) a uložena jako proměnná `Nout`.

5. POROVNÁNÍ A VYHODNOCENÍ

Tato kapitola se bude zabývat kvalitativním vyhodnocením a následným porovnáním chyb, vzniklých v důsledku převzorkování signálu. Testování proběhne na signálech převzorkovaných pomocí programů `resample_classic_varN`, `resample_alternating_varN` a `resample_GiantFFT`. Jejich implementace byla diskutována v oddíle 4. Navíc proběhne testování také na signálech převzorkovaných pomocí hudebního softwaru Reaper v6.78. Převzorkování bylo provedeno pomocí funkce renderování, kde byla jako metoda převzorkování vždy zvolena metoda sinc interpolation 192 pt a bitová hloubka 32 bit. Zmíněná metoda sinc interpolace v tomto textu doposud nebyla diskutována. V této metodě je každý bod signálu proložen funkcí sinc. Vzniknou tak nové pole, která jsou poté sečtena. Funkce sinc je teoreticky definována v nekonečném rozsahu, pro účely aplikace je však omezena na konečný počet vzorků, zde 192.

K testování bude využita sada testovacích signálů, která byla vygenerována prostřednictvím programu Matlab. Jedná se o následujících pět signálů: sinus, směs sinusovek, směs sinusovek s amplitudovou modulací, impulz a chirp (frequency sweep od 20 Hz do 20 kHz). Každý tento testovací signál byl vygenerován se vzorkovacím kmitočtem 8 kHz, 16 kHz, 32 kHz, 44,1 kHz, 48 kHz, 96 kHz a 192 kHz. Jedná se o vzorkovací kmitočty běžně používané v audio aplikacích [9]. Aliasing vzniklý u signálu chirp nebyl nijak kompenzován, protože to pro účely této práce není důležité. Všechny signály byly vytvořeny o délce 60 s a následně byly uloženy jako soubory .mat a jako soubory .wav s bitovou hloubkou 32 bit ve formátu single.

K vyhodnocení vzniklé chyby budou použity dvě veličiny. První z nich je veličina MSE – střední kvadratická chyba (mean squared error), která je definována vztahem

$$MSE = \frac{\|x - y\|}{n}, \tag{5.1}$$

kde x je vstupní signál, y je výstupní signál a n je počet vzorků signálu. Oba signály musí obsahovat stejný počet vzorků.

Druhou veličinou je veličina SDR – poměr signálu a zkreslení (signal to distortion ratio) definována vztahem

$$SDR = 20 \cdot \log_{10} \frac{\|x\|}{\|x - y\|} \text{ [dB]}, \tag{5.2}$$

kde x je vstupní signál a y je výstupní signál. Opět platí, že oba signály musí obsahovat stejný počet vzorků. Tento vztah může mít mnoho podob. Zde je použita logaritmická verze tohoto vztahu, protože výsledky budou mít velký dynamický rozsah. Vhodným aspektem je také udávání výsledků v decibelech, jelikož rovnou zobrazují, o kolik dB je zkreslení převzorkovaného signálu tišší oproti signálu původnímu.

Dále bude zohledněna výpočetní náročnost jednotlivých programů pomocí určení délky trvání jednotlivých výpočtů. K tomuto účelu budou v programech využity funkce `tic` a `toc`. Avšak nutno podotknout, že rozdíly mezi měřenými časy mohou být ovlivněny vytížením procesoru jinými souběžně probíhajícími procesy. V softwaru Reaper bude k tomuto účelu využit údaj o rychlosti renderování, který je vždy zobrazen.

Samotný výpočet výše zmíněných veličin bude proveden v prostředí programu Matlab. Všechny veličiny jsou z důvodu vysoké časové náročnosti měřeny pouze jednou (nejedná se o průměrované měření). K tomuto účelu byl implementován skript `compare_artificial`, pomocí kterého jsou porovnány datové soubory Matlabu .mat a skript `compare_audio`, který slouží k porovnání .wav audio souborů. Tento postup byl zvolen z důvodu různých nároků na porovnávání různých typů datových souborů.

5.1 Porovnání a vyhodnocení datových souborů Matlabu

Výhodou testování na datových souborech matlabu .mat je jejich přesnost. Proto je možné mezi sebou porovnávat signál převzorkovaný na konkrétní nový vzorkovací kmitočet f_{vzn} se signálem vygenerovaným na stejném vzorkovacím kmitočtu. Na tomto principu byl naprogramován skript `compare_artificialmat`, který bude popsán v tomto oddíle spolu s dosaženými výsledky jednotlivých programů pro konverzi vzorkovacího kmitočtu. Je zřejmé, že výsledky pro Reaper zde nebudou zahrnuty.

Prvně jsou názvy všech testovacích signálů uložených v souborech .mat uloženy v jedné matici nazvané `filename`. Dále jsou všechny možné nové vzorkovací kmitočty (8000, 16000, 32000, 44100, 48000, 96000, 192000) uloženy jako vektor `fsn`. Následně jsou alokovány matice `sdr`, `mse` a `et`, kde budou uloženy výsledky příslušných veličin popsaných v oddíle 5. V matici `et` (elapsed time) jsou uloženy data o celkovém času výpočtu. Hlavní algoritmus tohoto skriptu se skládá ze tří `for` cyklů, pomocí kterých jsou načítány jednotlivé signály prostřednictvím funkce `loadsignal`. Dále je vybrána příslušná metoda konverze, a nakonec vypočítány hodnoty `sdr`, `mse` a `et` a uloženy na příslušné místo v maticích. Tyto matice byly následně překopírovány do programu Excel a upraveny do formy přehledných tabulek.

V následujících tabulkách jsou uvedeny hodnoty SDR, MSE a elapsed time pro konverze vzorkovacích kmitočtů z 48 kHz na všechny výše zmíněné nové vzorkovací kmitočty. Tento postup ukáže, jak kvalitně je provedeno převzorkování v širokém rozsahu nových vzorkovacích kmitočtů.

Tabulka 5.1 Tabulka hodnot *SDR* [dB] pro soubory .mat a fs = 48 kHz

	sinus						
fsn [kHz] =	8	16	32	44,1	48	96	192
klasická metoda	14,16	23,68	23,68	25,43		29,68	26,17
alternativní metoda	14,16	23,68	23,68	21,45		29,68	26,17
Giant FFT	216,75	214,57	213,16	208,58		210,24	211,15
	směs sinusovek						
klasická metoda	16,20	25,71	5,65	7,31		11,52	8,06
alternativní metoda	16,20	25,71	5,65	3,55		11,52	8,06
Giant FFT	201,58	199,68	194,80	190,51		192,21	193,01
	směs sinusovek + AM						
klasická metoda	7,59	16,88	5,54	7,07		11,27	7,82
alternativní metoda	7,59	16,88	5,54	3,33		11,27	7,82
Giant FFT	201,50	199,13	194,43	190,25		191,90	192,74
	Impulz						
klasická metoda	0,70	1,67	9,52	6,69		−3,69	−7,06
alternativní metoda	0,70	1,67	9,52	2,01		−3,69	−7,06
Giant FFT	0,16	0,78	4,66	15,77		−2,37	−6,43

	Chirp						
klasická metoda	8,49	17,94	17,94	19,69		23,94	20,43
alternativní metoda	8,49	17,94	17,94	15,70		23,94	20,43
Giant FFT	66,96	88,28	105,52	120,18		110,56	109,94

Tabulka 5.2 Tabulka hodnot *MSE* pro soubory .mat a fs = 48 kHz

	sinus						
fsn [kHz] =	8	16	32	44,1	48	96	192
klasická metoda	2,00E-04	4,73E−05	3,34E−05	2,33E−05		9,67E−06	1,02E−05
alternativní metoda	2,00E−04	4,73E−05	3,34E−05	3,68E−05		9,67E−06	1,02E−05
Giant FFT	1,48E−14	1,35E−14	1,12E−14	1,62E−14		9,06E−15	5,77E−15
	směs sinusovek						
klasická metoda	2,24E−04	5,29E−05	5,33E−04	3,75E−04		1,56E−04	1,65E−04
alternativní metoda	2,24E−04	5,29E−05	5,33E−04	5,78E−04		1,56E−04	1,65E−04
Giant FFT	1,20E−13	1,06E−13	1,86E−13	2,59E−13		1,44E−13	9,32E−14
	směs sinusovek + AM						
klasická metoda	4,26E−04	1,03E−04	3,82E−04	2,72E−04		1,14E−04	1,20E−04
alternativní metoda	4,26E−04	1,03E−04	3,82E−04	4,19E−04		1,14E−04	1,20E−04
Giant FFT	8,59E−14	7,98E−14	1,37E−13	1,89E−13		1,06E−13	6,79E−14
	impulz						
klasická metoda	1,92E−06	8,59E−07	1,74E−07	1,75E−07		2,66E−07	1,96E−07
alternativní metoda	1,92E−06	8,59E−07	1,74E−07	3,00E−07		2,66E−07	1,96E−07
Giant FFT	2,05E−06	9,52E−07	3,04E−07	6,15E−08		2,28E−07	1,82E−07
	Chirp						
klasická metoda	3,84E−04	9,15E−05	6,47E−05	4,51E−05		1,87E−05	1,98E−05
alternativní metoda	3,84E−04	9,15E−05	6,47E−05	7,13E−05		1,87E−05	1,98E−05
Giant FFT	4,58E−07	2,78E−08	2,70E−09	4,26E−10		8,73E−10	6,64E−10

Tabulka 5.3 Tabulka hodnot elapsed time [s] pro soubory .mat a fs = 48 kHz

	sinus						
fsn [kHz] =	8	16	32	44,1	48	96	192
klasická metoda	0,145	0,058	0,120	13,581		0,120	0,252
alternativní metoda	0,058	0,035	0,107	5,126		0,115	0,309
Giant FFT	0,096	0,066	0,128	0,198		0,330	0,568
	směs sinusovek						
klasická metoda	0,057	0,044	0,174	15,025		0,119	0,273
alternativní metoda	0,047	0,052	0,183	4,907		0,145	0,285
Giant FFT	0,050	0,073	0,127	0,156		0,278	0,630
	směs sinusovek + AM						
klasická metoda	0,057	0,048	0,123	15,639		0,113	0,267
alternativní metoda	0,051	0,036	0,172	4,952		0,120	0,334
Giant FFT	0,056	0,072	0,120	0,210		0,336	0,597
	impulz						
klasická metoda	0,054	0,043	0,166	15,840		0,173	0,294
alternativní metoda	0,063	0,042	0,167	4,433		0,135	0,294
Giant FFT	0,058	0,084	0,127	0,150		0,325	0,585
	chirp						
klasická metoda	0,044	0,035	0,110	12,680		0,093	0,283
alternativní metoda	0,039	0,046	0,115	4,839		0,105	0,290
Giant FFT	0,050	0,079	0,099	0,154		0,291	0,609

Již z prvního pohledu na tabulky 5.1 Tabulka a 5.2 Tabulka je zřejmé, že metoda konverze Giant FFT dosahuje u všech testovacích signálů lepší výsledky než klasická a alternativní metoda. Výsledky těchto dvou metod se od sebe liší pouze při převzorkování na nový vzorkovací kmitočet 44,1kHz. To se děje z toho důvodu, že všechny ostatní vyhodnocované konverze pracují s tak jednoduchými hodnotami faktorů nadvzorkování a podvzorkování, že jejich postup výpočtu je pravděpodobně zcela totožný. Při převzorkování na 44,1kHz dosahuje metoda klasické konverze lehce lepších výsledků jako alternativní metoda.

Dále je třeba zdůraznit, že signál impulz je jako vektor reprezentován jedním vzorkem o hodnotě jedna, který je doplněn potřebným počtem nulových vzorků. Kvůli převzorkování se takový signál značně zkreslí, ať už kvůli použitým FIR filtrům typu dolní propusti, nebo useknutím spektra na Nyquistově kmitočtu. Proto zde uvedené výsledky pro převzorkování impulzního signálu mají jen velmi malou vypovídající hodnotu.

U signálu chirp je u metody Giant FFT vidět značný pokles v hodnotách SDR oproti jiným testovacím signálům (vyjma signálu impulz). Pravděpodobně se tak děje kvůli neustále se měnící povaze signálu chirp. Rychlá Fourierova transformace je totiž

navržena pro práci s ustálenými systémy. V signálu chirp se v podstatě nenachází žádná ustálená sinusová vlna, jelikož se frekvence tohoto signálu neustále mění (roste). Proto se jedná o neustálený systém.

V tabulce 5.3 Tabulka lze pozorovat, že celkový čas výpočtu pomocí metody Giant FFT roste s rostoucí velikostí nového vzorkovacího kmitočtu, protože tím také značně narůstá objem dat, s kterými musí pracovat funkce ifft. U dvou zbylých metod celkový čas výpočtu roste s rostoucí velikostí faktorů podvzorkování a nadvzorkování a u konverze na 44,1kHz čas výpočtu nabývá extrémních hodnot v jednotkách sekund. Pokud je tento fakt opomenut metoda alternativní konverze nabývá nejlepších výsledků pro celkový čas výpočtu.

V následujících tabulkách jsou uvedeny hodnoty SDR, MSE a elapsed time pro konverze vzorkovacích kmitočtů z 44,1kHz na všechny výše zmíněné nové vzorkovací kmitočty. Tento postup ukáže, jak kvalitně je provedeno převzorkování s velkou hodnotou faktorů nadvzorkování a podvzorkování.

Tabulka 5.4 Tabulka hodnot SDR [dB] pro soubory .mat a fs = 44,1 kHz

	sinus		
fsn [kHz] =	32	44,1	48
klasická metoda	21,176		21,841
alternativní metoda	12,654		16,841
Giant FFT	211,487		208,541
	směs sinusovek		
klasická metoda	3,587		3,965
alternativní metoda	−2,800		−0,350
Giant FFT	193,613		190,435
	směs sinusovek + AM		
klasická metoda	3,620		3,746
alternativní metoda	−2,824		−0,494
Giant FFT	193,346		190,176
	impulz		
klasická metoda	9,179		−0,788
alternativní metoda	−1,423		−3,508
Giant FFT	6,048		9,583
	chirp		
klasická metoda	15,42991		16,12911
alternativní metoda	7,027925		11,15908
Giant FFT	106,1955		119,3662

Tabulka 5.5 Tabulka hodnot *MSE* pro soubory .mat a fs = 44,1 kHz

	sinus		
fsn [kHz] =	32	44,1	48
klasická metoda	4,46E−05		3,37E−05
alternativní metoda	1,19E−04		5,99E−05
Giant FFT	1,36E−14		1,56E−14
	směs sinusovek		
klasická metoda	6,75E−04		5,28E−04
alternativní metoda	1,41E−03		8,68E−04
Giant FFT	2,13E−13		2,51E−13
	směs sinusovek + AM		
klasická metoda	4,76E−04		3,83E−04
alternativní metoda	9,99E−04		6,24E−04
Giant FFT	1,55E−13		1,83E−13
	impulz		
klasická metoda	1,81E−07		3,80E−07
alternativní metoda	6,14E−07		5,20E−07
Giant FFT	2,60E−07		1,15E−07
	chirp		
klasická metoda	8,64E−05		6,51E−05
alternativní metoda	2,27E−04		1,15E−04
Giant FFT	2,50E−09		4,48E−10

Tabulka 5.6 Tabulka hodnot elapsed time [s] pro soubory .mat a fs = 44,1 kHz

	sinus		
fsn [kHz] =	32	44,1	48
klasická metoda	28,590		47,516
alternativní metoda	6,909		9,192
Giant FFT	0,168		0,189
	směs sinusovek		
klasická metoda	32,953		49,523
alternativní metoda	7,279		9,906
Giant FFT	0,141		0,177
	směs sinusovek + AM		
klasická metoda	32,504		46,069
alternativní metoda	6,925		9,749
Giant FFT	0,141		0,166
	impulz		
klasická metoda	30,284		46,692
alternativní metoda	6,558		9,146
Giant FFT	0,138		0,171

	chirp		
klasická metoda	30,984		46,385
alternativní metoda	6,002		8,964
Giant FFT	0,144		0,183

Z tabulek 5.4 Tabulka 5.5 Tabulka a 5.6 Tabulka lze vyčíst, že klasická metoda konverze nabývá zcela nevhodných časů výpočtu, ale její výsledky pro *SDR* a *MSE* jsou o něco lepší než výsledky alternativní metody konverze. Obě tyto metody však nedosahují dostačujících výsledků pro zkoumané konverze. Oproti tomu metoda Giant FFT dosahuje nesrovnatelně lepších výsledků jak v kvalitě, tak v rychlosti převzorkování.

5.2 Porovnání a vyhodnocení umělých audio souborů

Testování na audio souborech .wav bylo provedeno obdobně jako v oddíle 5.1, protože tyto soubory byli uměle vygenerovány zároveň s testovacími soubory .mat. Je tedy možné mezi sebou porovnávat signál převzorkovaný na konkrétní nový vzorkovací kmitočet f_{vzn} se signálem vygenerovaným na stejném vzorkovacím kmitočtu. Na tomto principu byl naprogramován skript `compare_artificialwav`, který bude popsán v tomto oddíle spolu s dosaženými výsledky jednotlivých programů pro konverzi vzorkovacího kmitočtu. V tomto oddíle budou zahrnuty také výsledky dosažené softwarem Reaper.

Prvně jsou názvy všech testovacích signálů vygenerovaných pomocí programu Matlab a uložených v souborech .wav uloženy v jedné matici nazvané `filename`. Poté jsou názvy všech testovacích signálů převzorkovaných pomocí softwaru Reaper a uložených v souborech .wav (konverze v Reaperu byla provedena předem) uloženy v jedné matici nazvané `filenamer`. Dále jsou všechny možné vzorkovací kmitočty (8000, 16000, 32000, 44100, 48000, 96000, 192000) uloženy jako vektor `fs`. Následně jsou alokovány matice `sdr`, `mse` a `et`, kde budou uloženy výsledky příslušných veličin popsaných v oddíle 5. V matici `et` (elapsed time) jsou uloženy data o celkovém času výpočtu. Hlavní algoritmus tohoto skriptu se skládá ze tří `for` cyklů, pomocí kterých jsou načítány jednotlivé signály prostřednictvím funkce `loadsignal`. Dále je vybrána a provedena příslušná metoda konverze. Pokud je v příslušném kroku cyklu zvolen software Reaper, neprovádí se žádná metoda konverze a místo toho je pouze načten příslušný signál, který již byl převzorkován pomocí softwaru Reaper. Konkrétním způsobem užití softwaru Reaper se zabývá oddíl 5.3. Nakonec jsou vypočítány hodnoty `sdr`, `mse` a `et` a následně jsou uloženy na příslušné místo v maticích. Tyto matice byly následně překopírovány do programu Excel a upraveny do formy přehledných tabulek.

V následujících tabulkách jsou uvedeny hodnoty SDR, MSE a elapsed time pro konverze vzorkovacích kmitočtů z 48 kHz na všechny výše zmíněné nové vzorkovací kmitočty. Tento postup ukáže, jak kvalitně je provedeno převzorkování v širokém rozsahu nových vzorkovacích kmitočtů.

Tabulka 5.7 Tabulka hodnot *SDR* [dB] pro soubory .wav a fs = 48 kHz

	sinus						
fsn [kHz] =	8	16	32	44,1	48	96	192
klasická metoda	14,16	23,68	23,68	25,43		29,68	26,17
alternativní metoda	14,16	23,68	23,68	21,45		29,68	26,17
Giant FFT	142,32	147,91	148,70	142,90		148,06	145,41
Reaper	39,48	39,48	39,48	39,48		39,48	39,48
	směs sinusovek						
klasická metoda	16,20	3,00	5,65	7,31		11,52	8,06
alternativní metoda	16,20	3,00	5,65	3,55		11,52	8,06
Giant FFT	152,95	3,01	154,34	148,78		152,51	151,27

Reaper	6,01	2,50	2,50	2,49		2,50	2,50
	směs sinusovek + AM						
klasická metoda	2,96	3,94	5,54	7,07		11,27	7,82
alternativní metoda	2,96	3,94	5,54	3,33		11,27	7,82
Giant FFT	3,68	3,98	154,45	149,95		154,78	151,85
Reaper	2,50	2,50	2,50	2,49		2,50	2,50
	impulz						
klasická metoda	0,70	1,67	9,52	6,69		−3,69	−7,06
alternativní metoda	0,70	1,67	9,52	2,01		−3,69	−7,06
Giant FFT	0,16	0,78	4,66	15,77		−2,37	−6,43
Reaper	17,47	17,60	16,81	16,42		2,92	1,22
	chirp						
klasická metoda	8,49	17,94	17,94	19,69		23,94	20,43
alternativní metoda	8,49	17,94	17,94	15,70		23,94	20,43
Giant FFT	66,96	88,28	105,52	120,18		110,56	109,94
Reaper	37,89	37,98	38,01	38,02		38,02	38,02

Tabulka 5.8 Tabulka hodnot *MSE* pro soubory .wav a fs = 48 kHz

	sinus						
fsn [kHz] =	8	16	32	44,1	48	96	192
klasická metoda	2,00E−04	4,73E−05	3,34E−05	2,33E−05		9,67E−06	1,02E−05
alternativní metoda	2,00E−04	4,73E−05	3,34E−05	3,68E−05		9,67E−06	1,02E−05
Giant FFT	7,81E−11	2,90E−11	1,87E−11	3,11E−11		1,16E−11	1,12E−11
Reaper	1,08E−05	7,66E−06	5,42E−06	4,62E−06		3,13E−06	2,21E−06
	směs sinusovek						
klasická metoda	2,24E−04	1,02E−03	5,33E−04	3,75E−04		1,56E−04	1,65E−04
alternativní metoda	2,24E−04	1,02E−03	5,33E−04	5,78E−04		1,56E−04	1,65E−04
Giant FFT	3,25E−11	1,02E−03	1,96E−11	3,16E−11		1,40E−11	1,14E−11
Reaper	7,23E−04	1,08E−03	7,66E−04	6,52E−04		4,42E−04	3,13E−04
	směs sinusovek + AM						
klasická metoda	1,36E−03	7,25E−04	3,82E−04	2,72E−04		1,14E−04	1,20E−04
alternativní metoda	1,36E−03	7,25E−04	3,82E−04	4,19E−04		1,14E−04	1,20E−04
Giant FFT	1,25E−03	7,22E−04	1,37E−11	1,96E−11		7,60E−12	7,53E−12
Reaper	1,43E−03	8,56E−04	5,41E−04	4,61E−04		3,13E−04	2,21E−04
	Impulz						
klasická metoda	1,92E−06	8,59E−07	1,74E−07	1,75E−07		2,66E−07	1,96E−07
alternativní metoda	1,92E−06	8,59E−07	1,74E−07	3,00E−07		2,66E−07	1,96E−07
Giant FFT	2,05E−06	9,52E−07	3,04E−07	6,15E−08		2,28E−07	1,82E−07
Reaper	2,79E−07	1,37E−07	7,52E−08	5,71E−08		1,24E−07	7,54E−08
	Chirp						
klasická metoda	3,84E−04	9,15E−05	6,47E−05	4,51E−05		1,87E−05	1,98E−05

alternativní metoda	3,84E−04	9,15E−05	6,47E−05	7,13E−05		1,87E−05	1,98E−05
Giant FFT	4,58E−07	2,78E−08	2,70E−09	4,26E−10		8,73E−10	6,64E−10
Reaper	1,30E−05	9,10E−06	6,42E−06	5,46E−06		3,70E−06	2,62E−06

Tabulka 5.9 Tabulka hodnot elapsed time [s] pro soubory .wav a fs = 48 kHz

	sinus						
fsn [kHz] =	8	16 kHz	32 kHz	44,1 kHz	48 kHz	96 kHz	192
klasická metoda	0,097	0,028	0,094	11,593		0,083	0,210
alternativní metoda	0,048	0,029	0,091	4,609		0,084	0,234
Giant FFT	0,077	0,075	0,110	0,177		0,283	0,569
Reaper	0,424	0,611	0,968	1,695		2,459	4,724
	směs sinusovek						
klasická metoda	0,044	0,041	0,135	15,120		0,080	0,231
alternativní metoda	0,047	0,031	0,120	4,747		0,092	0,223
Giant FFT	0,074	0,102	0,136	0,166		0,277	0,543
Reaper	0,424	0,611	0,968	1,695		2,459	4,724
	směs sinusovek + AM						
klasická metoda	0,048	0,046	0,117	14,731		0,117	0,250
alternativní metoda	0,042	0,029	0,135	5,308		0,117	0,235
Giant FFT	0,074	0,088	0,136	0,162		0,336	0,568
Reaper	0,424	0,611	0,968	1,695		2,459	4,724
	impulz						
klasická metoda	0,049	0,029	0,128	14,933		0,073	0,209
alternativní metoda	0,046	0,029	0,125	4,549		0,094	0,250
Giant FFT	0,073	0,090	0,135	0,199		0,276	0,573
Reaper	0,424	0,611	0,968	1,695		2,459	4,724
	chirp						
klasická metoda	0,065	0,032	0,109	15,151		0,076	0,251
alternativní metoda	0,043	0,030	0,155	4,652		0,077	0,257
Giant FFT	0,089	0,091	0,138	0,179		0,288	0,563
Reaper	0,424	0,611	0,968	1,695		2,459	4,724

V tabulkách 5.7 Tabulka a 5.8 Tabulka lze názorně vidět, že výsledky u prvních třech signálů pro všechny implementované metody konverze jsou velmi podobné výsledkům v kapitole 5.1 což se dá očekávat, jelikož se jedná o shodné signály. Největší rozdíl je zde jednoznačně u metody Giant FFT, která zde má viditelně horší výsledky. Navíc se v tabulce vyskytují očividné chyby (např. podvzorkování směsi sinusovek + AM metodou Giant FFT). Pro tyto chyby není známo žádné vysvětlení. V tabulce 5.9 Tabulka lze vidět, že celkový čas výpočtu je obecně o něco nižší než v tabulce 5.3 Tabulka

Nejzajímavějším předmětem k diskusi je zde jednoznačně software Reaper. Z tabulek 5.7 Tabulka a 5.8 Tabulka lze vyčíst, že Reaper dosahuje mnohem lepších výsledků při konverzi signálů sinus, impulz a chirp než klasická a alternativní metoda. Ale u signálů směs sinusovek a amplitudově modulovaná směs sinusovek dosahuje výsledků znatelně horších. Celkový čas výpočtu u softwaru Reaper závislý pouze na množství dat určených ke konverzi obdobně jako tomu je u metody Giant FFT. V tabulce 5.9 Tabulka lze ale vidět že provedení konverze je přibližně desetkrát až dvacetkrát pomalejší oproti metodě Giant FFT. Avšak tento výsledek není zcela směrodatný, jelikož není známo, co všechno je softwarem Reaper během exportování audio souboru počítáno. Ve zkratce jsou po renderování uživateli poskytnuty kromě výsledných audio souborů také grafy hlasitostí a různé další statistky. Navíc program Reaper umožňuje paralelní renderování více souborů současně, ale nebylo testováno, jaký má tato skutečnost vliv na celkový čas výpočtu. Po porovnání všech výsledků je metoda Giant FFT očekávaně nejlepší.

V následujících tabulkách jsou uvedeny hodnoty SDR, MSE a elapsed time pro konverze vzorkovacích kmitočtů z 44,1 kHz na všechny výše zmíněné nové vzorkovací kmitočty. Tento postup ukáže, jak kvalitně je provedeno převzorkování s velkou hodnotou faktorů nadvzorkování a podvzorkování.

Tabulka 5.10 Tabulka hodnot SDR [dB] pro soubory .wav a fs = 44,1 kHz

	sinus		
fsn [kHz] =	32	44,1	48
klasická metoda	21,868		23,476
alternativní metoda			18,866
Giant FFT	157,554		161,997
Reaper	38,497		38,497
	směs sinusovek		
klasická metoda	5,211		6,091
alternativní metoda			3,264
Giant FFT	140,041		141,103
Reaper	2,515		2,516
	směs sinusovek + AM		
klasická metoda	5,349		5,936
alternativní metoda			3,239
Giant FFT	145,251		138,815
Reaper	2,515		2,516
	Impulz		
klasická metoda	4,658		5,936
alternativní metoda			3,239
Giant FFT	16,089		10,815
Reaper	22,719		15,942

	chirp		
klasická metoda	16,17183		17,7635
alternativní metoda			13,22315
Giant FFT	124,2859		113,14
Reaper	44,030		44,037

Tabulka 5.11 Tabulka hodnot *MSE* pro soubory .wav a fs = 44,1 kHz

	sinus		
fsn [kHz] =	32	44,1	48
klasická metoda	4,12E−05		2,79E−05
alternativní metoda			4,75E−05
Giant FFT	6,76E−13		3,31E−12
Reaper	2,71E−06		2,21E−06
	směs sinusovek		
klasická metoda	5,60E−04		4,13E−04
alternativní metoda			5,72E−04
Giant FFT	3,21E−12		2,32E−12
Reaper	3,83E−04		3,13E−04
	směs sinusovek + AM		
klasická metoda	3,90E−04		2,98E−04
alternativní metoda			4,06E−04
Giant FFT	1,25E−12		2,14E−12
Reaper	2,71E−04		2,21E−04
	impulz		
klasická metoda	3,05E−07		2,20E−07
alternativní metoda			2,56E−07
Giant FFT	2,58E−10		4,95E−08
Reaper	3,81E−08	0,00	5,54E−08
	chirp		
klasická metoda	7,93E−05		5,39E−05
alternativní metoda			9,09E−05
Giant FFT	9,85E−13		9,18E−11
Reaper	3,81E−08		2,62E−06

Tabulka 5.12 Tabulka hodnot elapsed time [s] pro soubory .wav a fs = 44,1 kHz

	sinus		
fsn [kHz] =	32	44,1	48
klasická metoda	118,782		28,423
alternativní metoda			6,703
Giant FFT	0,180		0,222
Reaper	1,299		1,858
	směs sinusovek		
klasická metoda	118,615		30,508
alternativní metoda			6,862
Giant FFT	0,112		0,152
Reaper	1,299		1,858
	směs sinusovek + AM		
klasická metoda	118,987		30,272
alternativní metoda			6,662
Giant FFT	0,128		0,153
Reaper	1,299		1,858
	impulz		
klasická metoda	117,364		29,702
alternativní metoda			6,843
Giant FFT	0,107		0,151
Reaper	1,299		1,858
	chirp		
klasická metoda	118,700		29,900
alternativní metoda			6,639
Giant FFT	0,105		0,151
Reaper	1,299		1,858

Výsledky v tabulkách 5.10 Tabulka 5.11 Tabulka a 5.12 Tabulka dopadli dle očekávání. Nutno podotknout, že zde chybí hodnoty pro konverzi vzorkovacího kmitočtu z 32kHz na 44,1kHz a zpět u alternativní metody, jelikož tato metoda nebyla schopna nalézt vyhovující délky filtrů.

Software Reaper dosahuje stejně kvalitních výsledků i pro vysoké hodnoty faktorů podvzorkování a nadvzorkování.

5.3 Porovnávání a vyhodnocení audio nahrávek

Tento oddíl se zabývá testováním na celých skladbách uložených v audio souborech .wav. Pro porovnání byli vybrány následující skladby (v závorkách jsou uvedeny parametry podstatné pro tuto práci). Pavel Zlámal – Svatebního dne (2:55 min, 48 kHz, 24 bit), Plaster – Iperstatic (3:21 min, 44,1 kHz, 16 bit), TOPS – I Feel Alive (2:34 min, 44,1 kHz, 16 bit), sinks – the city (2:30 min, 44,1 kHz, 16 bit). Při výběru byl kladen důraz na žánrovou rozmanitost (jazz, elektronická hudba, pop, rock). Všechny skladby byly zakoupeny na webových stránkách Bandcamp. Z toho důvodu nebylo možné ovlivnit jejich vzorkovací kmitočet.

Testování bylo provedeno následovně. Signál s konkrétním vzorkovacím kmitočtem je prvně převzorkován na nový vzorkovací kmitočet a poté je znovu převzorkován zpět na původní vzorkovací kmitočet. Tím jsou získány dva signály, které je možné díky jejich stejné délce porovnávat pomocí veličin *SDR* a *MSE*. Na tomto principu byl naprogramován skript `compare_audio`, který bude popsán v tomto oddíle spolu s dosaženými výsledky jednotlivých programů pro konverzi vzorkovacího kmitočtu. V tomto oddíle budou zahrnuty také výsledky dosažené softwarem Reaper.

Prvně jsou názvy všech skladeb uložených v souborech .wav uloženy v jedné matici nazvané `filename`. Poté jsou názvy všech těchto skladeb převzorkovaných pomocí softwaru Reaper (konverze v Reaperu byla provedena předem) a uložených v souborech .wav uloženy v jedné matici nazvané `filenamer`. Dále jsou všechny možné nové vzorkovací kmitočty (44100, 48000, 96000, 192000) uloženy jako vektor `fs`. Následně jsou alokovány matice `sdr`, `mse` a `et`, kde budou uloženy výsledky příslušných veličin popsaných v oddíle 5. V matici `et` (elapsed time) jsou uloženy data o celkovém času výpočtu. Hlavní algoritmus tohoto skriptu se skládá ze tří `for` cyklů, pomocí kterých jsou načítány jednotlivé signály prostřednictvím funkce `loadsignal`. Dále je vybrána příslušná metoda konverze, která je provedena dvakrát. Prvně na nový vzorkovací kmitočet uložený v proměnné `fsn` a poté zpět na původní vzorkovací kmitočet `fs`. Pokud je v příslušném kroku cyklu zvolen software Reaper neprovádí se žádná metoda konverze a místo toho je pouze načten příslušný signál, který již byl dvakrát převzorkován pomocí softwaru Reaper. Konverze byla provedena prostřednictvím exportu audio souboru .wav při kterém je také možné provést převzorkování zvolenou metodou. Bitová hloubka výstupního audio souboru byla vždy nastavena na stejnou hodnotu jako u vstupního audio souboru. Software Reaper vždy pracoval na stejné vzorkovací frekvenci jako vstupní audio signál, aby nedošlo k automatické konverzi před exportem. Nakonec jsou vypočítány hodnoty `sdr`, `mse` a `et`, které jsou navíc vhodně upraveny, aby chyba vzniklá převzorkováním odpovídala jednomu převzorkování a následně jsou uloženy na příslušné místo v maticích. Tyto matice byly následně překopírovány do programu Excel a upraveny do formy přehledných tabulek.

V následujících tabulkách jsou uvedeny hodnoty SDR, MSE a elapsed time pro konverze vzorkovacích kmitočtů skladeb na výše uvedené nové vzorkovací kmitočty a zpět. Tento postup ukáže, jak kvalitně je provedeno převzorkování reálných audio souborů jednotlivými metodami.

Tabulka 5.13 Tabulka hodnot *SDR* [dB] pro reálné audio soubory .wav

	Zlámal			
fsn [kHz] =	44,1	48	96	192
klasická metoda	22,384		28,280	24,876
alternativní metoda	18,502		28,280	24,876
Giant FFT	61,385		112,235	112,233
Reaper	3,206		2,848	4,123
	Plaster			
klasická metoda		32,889	12,162	
alternativní metoda		29,370	22,753	22,917
Giant FFT		105,035	105,472	104,977
Reaper		2,078	5,037	5,135
	TOPS			
klasická metoda		21,098	2,802	
alternativní metoda		16,855	6,674	6,387
Giant FFT		115,209	116,193	116,193
Reaper		2,078	5,037	5,135
	sinks			
klasická metoda		20,560	2,739	
alternativní metoda		16,518	8,674	8,494
Giant FFT		120,274	120,479	120,479
Reaper		2,078	5,037	5,135

Tabulka 5.14 Tabulka hodnot *MSE* pro reálné audio soubory .wav

	Zlámal			
fsn [kHz] =	44,1	48	96	192
klasická metoda	2,27E−06		1,15E−06	1,70E−06
alternativní metoda	3,54E−06		1,15E−06	1,70E−06
Giant FFT	2,54E−08		7,29E−11	7,29E−11
Reaper	2,06E−05		2,15E−05	1,86E−05
	Plaster			
klasická metoda		1,75E−06	1,90E−05	
alternativní metoda		2,62E−06	5,62E−06	5,51E−06
Giant FFT		4,32E−10	4,11E−10	4,35E−10
Reaper		6,07E−05	4,32E−05	4,27E−05

	TOPS			
klasická metoda		1,07E−05	8,79E−05	
alternativní metoda		1,74E−05	5,63E−05	5,82E−05
Giant FFT		2,11E−10	1,88E−10	1,88E−10
Reaper		6,49E−05	6,45E−05	6,41E−05
	sinks			
klasická metoda		1,08E−05	8,44E−05	
alternativní metoda		1,73E−05	4,26E−05	4,35E−05
Giant FFT		1,12E−10	1,09E−10	1,09E−10
Reaper		8,26E−05	8,32E−05	7,70E−05

Tabulka 5.15 Tabulka hodnot elapsed time [s] pro reálné audio soubory .wav

	Zlámal			
fsn [kHz] =	44,1	48	96	192
klasická metoda	107,861		0,300	0,759
alternativní metoda	23,496		0,330	0,785
Giant FFT	0,937		2,049	3,460
Reaper	2,971		4,397	10,870
	Plaster			
klasická metoda		145,986	528,0654	
alternativní metoda		25,991	35,772	90,801
Giant FFT		0,748	1,278	2,180
Reaper		3,673	7,319	14,126
	TOPS			
klasická metoda		86,553	287,946	
alternativní metoda		19,081	32,927	61,338
Giant FFT		0,733	1,267	2,000
Reaper		2,852	5,461	10,546
	sinks			
klasická metoda		85,033	285,692	
alternativní metoda		20,592	27,390	58,989
Giant FFT		0,709	1,309	2,332
Reaper		2,807	5,432	

Pro vstupní signál o vzorkovací frekvenci 44,1 kHz nebylo možné použít funkce pro konverzi vzorkovacího kmitočtu klasickou metodou `resample_classic_varN` a alternativní metodou `resample_alternating_varN`, protože nebylo možné zajistit celočíselné zpoždění i přes volbu velké délky FIR filtrů. Tento problém se vyskytnul pravděpodobně kvůli velkým faktorům nadvzorkování a podvzorkování v kombinaci se signály které v průběhu řetězce zpracování nabývali délek až ve stovkách milionů vzorků. Místo toho byly použity funkce `resample_classic_fixN`

a `resample_alternating_fixN`, které nesou riziko nesprávného určení celkového zpoždění výstupního signálu dle 4.1.1 a 4.2.1, ale jsou schopny dosáhnout výsledků. Jedinou výjimkou je konverze klasickou metodou ze 44,1 kHz na 192 kHz, kdy byla během výpočtu přesažena maximální paměťová kapacita programu Matlab (31 GB bylo přesaženo o více jak 10 GB z důvodu pole čítajícího více než 5 miliard prvků). Proto nejsou v tabulce pro tuto konverzi uvedeny žádné výsledky. Problém by se dal případně řešit provedením dvou konverzí po sobě. V takovém případě by nebyla překročena maximální paměťová kapacita Matlabu.

V tabulkách 5.13 Tabulka hodnot *SDR* [dB] pro reálné audio soubory .wav hodnot *MSE* pro reálné audio soubory .wav5.14 Tabulka pozorovat, že metoda Giant FFT má očekávaně nejlepší výsledky. Na druhou stranu software Reaper nabývá velmi špatných výsledků. To mohlo nastat ale i z důvodu nekompenzovaného zpoždění signálu vzniklého například kvůli výstupnímu vyhlazovacímu filtru, ale jedná se pouze o hypotézu. Bohužel není známo, zdali Reaper toto zpoždění na výstupu kompenzuje nebo ne. Dále lze pozorovat vliv konkrétních skladeb na *SDR* a *MSE*. Metoda Giant FFT má horší výsledky, čím větší má skladba dynamický rozsah (Zlámal, Plaster), doba trvání skladeb nemá velký vliv. Klasická metoda nabývá překvapivě dobrých výsledků, vyjma výsledků při konverzi s velkými faktory nadvzorkování a podvzorkování. To stejné platí i pro alternativní metodu, která má ale obecně horší výsledky. Obě tyto metody mají tím horší výsledky čím menší je dynamický rozsah skladeb (TOPS, sinks). Hodnoty elapsed time neboli doby trvání konverze lze nalézt v tabulce 5.15 Tabulka Výsledky vyšly dle očekávání. Klasická metoda má dobu konverze v řádech minut, metoda Giant FFT je naopak nejrychlejší. Velmi dobré výsledky má i software Reaper. Pro software Reaper stále platí stejné principy jako v oddíle 5.2.

6. ZÁVĚR

Tato práce přinesla výsledky v oblasti konverze vzorkovacího kmitočtu audio signálů. Byly vypočítány hodnoty *SDR*, *MSE* a doby výpočtu pro klasickou metodu konverze, alternativní metodu konverze, metodu Giant FFT a metodu sinc interpolace. Metoda sinc interpolace byla otestována prostřednictvím softwaru Reaper v kterém je nativně implementována. Zbylé tři metody byly implementovány v prostředí softwaru Matlab. Pro účely testování byly zvoleny konverze mezi typickými hodnotami vzorkovacích kmitočtů uplatňovaných v audio aplikacích [9].

Po porovnání mnoha dosažených výsledků se dá s jistotou říct, že metoda Giant FFT nabývá nejlepších hodnot *SDR* a *MSE* při konverzi umělých audio signálů i reálných skladeb. Navíc dosahuje velmi dobrých hodnot času výpočtu, které nepřímo ukazují, jak je daná konverze datově náročná. Dále je pak pro praktickou aplikaci této metody vhodné, že čas výpočtu závisí pouze na objemu dat výstupního signálu. Klasická a alternativní metoda konverze jsou v praxi téměř nepoužitelné pro jejich velké výkyvy v dosažených hodnotách *SDR*, *MSE* a také času výpočtu, které jsou obecně tím horší čím větší jsou faktory nadvzorkování a podvzorkování. Sinc interpolace v prostředí softwaru Reaper má rovněž velké výkyvy v hodnotách *SDR* a *MSE*. To však může být dáno i tím, že zde není kompenzováno případné zpoždění výstupního signálu oproti vstupnímu signálu. Výsledky pro čas výpočtu jsou u této metody stabilní, závislé pouze na datovém objemu výstupního signálu. Obecně je tato metoda několikanásobně pomalejší než metoda Giant FFT. Nutno ale brát v potaz, že se jedná o jedinou metodu, která provádí konverzi stereo signálů, a navíc v průběhu konverze počítá různé statistky.

Dále je také nutné zmínit problematiku zpoždění signálu, které musí být pro účely porovnávání původního a převzorkovaného signálu kompenzováno. Způsob, jakým je tato operace uskutečněna ve funkcích `resample_classic_varN` a `resample_alternating_varN` nese to riziko, že nebude nalezena vhodná délka filtrů pro celočíselné zpoždění. To platí zvláště pro konverze ze vzorkovacího kmitočtu 44,1 kHz.

Tuto práci je možné dále rozvést například v metodice porovnávání původního a převzorkovaného signálu. To lze uskutečnit nalezením lepších způsobů kompenzace zpoždění (např. volbou různých délek FIR filtrů v řetězci výpočtu u klasických metod). Dále je možné kontaktovat vývojáře softwaru Reaper a analyzovat samotný kód pro sinc interpolaci, která je hojně používána v tomto i v jiných DAW. Tento postup by vedl k lepší kvalitě a vypovídající hodnotě některých dosažených výsledků uvedených v této práci. Další možností pro navazující práci by bylo hlubší ponoření do implementace metody Giant FFT, která se jeví jako nejkvalitnější metoda pro konverzi vzorkovacího kmitočtu audio signálů. Například by šlo o rozšíření této metody na stereo signály, implementace ve vhodnějším programovacím jazyce pro praktickou aplikaci v prostředí DAW a následná spolupráce s vývojáři, například právě softwaru Reaper.

LITERATURA

[1] VAIDYANATHAN P. P. *Multirate Systems and Filter Banks*. Pearson College Div, 1992. ISBN 978-0136057185.

[2] VÄLIMÄKI V., BILBAO S. *Giant FFTs for Sample-Rate Conversion*. J. Audio Eng. Soc., vol. 71, no. 3, 2023.s

[3] FLIEGE N. J. *Multirate digital signal processing: multirate systems, filter banks, wavelets*. Chichester: John Wiley, 2000. ISBN 0471492043.

[4] SMÉKAL Z., SYSEL P. *Číslicové filtry*. Elektronická skripta FEKT VUT, Brno, 2012.

[5] VARGIČ R. *Wavelety a banky filtrov*. Elektronická skripta FEI STU, Bratislava, 2003. Dostupné na https://www.ktl.elf.stuba.sk/~vargic/wabf/skripta/ebook/skripta_wabf2004.pdf

[6] Digital Signal Processing Committee of the IEEE Acoustics, Speech, and Signal Processing Society, eds. *Programs for Digital Signal Processing*. New York: IEEE Press, 1979, Algorithm 5.2.

[7] ADAMS J. *A Subsequence Approach to Interpolation Using the FFT*. IEEE Trans. Circ.Syst., vol. 34, no. 5, pp. 568–570. 1987 May. https://doi.org/10.1109/TCS.1987.1086169.

[8] IMMINK K. A. *The Compact Disc Story*, J. AudioEng. Soc., vol. 46, no. 5, pp. 458–460, 462, 464, 465. 1998 May.

[9] AES. *AES Recommended Practice for Professional Digital Audio – Preferred Sampling Frequencies for Applications Employing Pulse-Code Modulation*. AES Standard AES5-2018. 2018 Dec.

[10] ZEINEDDINE A., NAFKHA A., PAQUELET S., MOY C., JEZEQUEL P. Y., *Comprehensive Survey of FIR-Based Sample Rate Conversion*. J. Signal Process. Syst., vol. 93, pp. 113–125. 2021 Jul. https://doi.org/10.1007/s11265-020-01575-6.

[11] KOZUMPLÍK J. *Multitaktní systémy*. Elektronická skripta FEKT VUT, Brno, 2005. Dostupné na https://www.vutbr.cz/www_base/priloha.php?dpid=23933.

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

FEKT	Fakulta elektrotechniky a komunikačních technologií
VUT	Vysoké učení technické v Brně
FIR	Finite impulse response
FFT	Fast Fourier transform
DAW	Digital audio workstation
CD	compact disk
A/D	analogově-digitální
WAV	Waveform audio file format
MAT	Matlab data file
GB	Gigabyte

Symboly:

y_P	podvzorkovaný signál	
M	faktor podvzorkování	
y_N	nadvzorkovaný signál	
L	faktor nadvzorkování	
f_{vz}	původní vzorkovací kmitočet	(Hz)
f_{vzn}	nový vzorkovací kmitočet	(Hz)
f_a	aktuální vzorkovací kmitočet	(Hz)

PŘÍLOHY

Část příloh skládající se z programů a z výběru několika testovacích signálů je uložena v souboru .zip a je dostupná z https://www.vut.cz/studenti/zav-prace/detail/153582. Databáze testovacích signálů je umístěna na sdíleném Google disku a dostupná z https://drive.google.com/drive/folders/1-HG2Xt2j3mtrODLTqs5Xsw7DoWsWIeDs?usp=sharing.

Nejdůležitějším programem je spouštěcí skript `resample_demo`. V prostředí tohoto skriptu je možné provést konverzi mezi jakýmikoliv dvěma vzorkovacími kmitočty, prostřednictvím zvolené metody a zvoleného testovacího signálu. Skript dále umožňuje vykreslovat grafy a ukládat data.

Dalšími programy ke spuštění jsou `compare_artificialmat`, `compare_artificialwav` a `compare_audio`, které slouží k porovnávání původního a převzorkovaného signálu prostřednictvím výpočtu *SDR, MSE* a doby trvání výpočtu. Všechny jsou podrobně popsány v kapitole 5.

Pro správnou funkci všech spustitelných programů je nutné stažení celé databáze testovacích signálů a její umístění do složky ve které se nacházejí spouštěné programy.