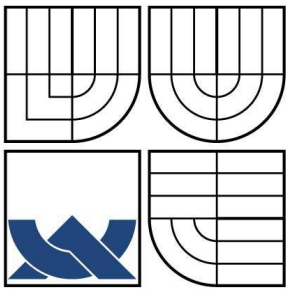
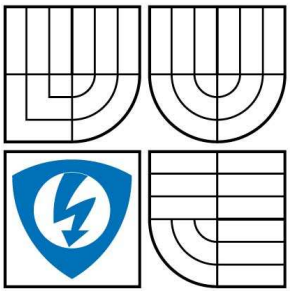


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

DIGITÁLNÍ ZVUKOVÝ EFEKT TYPU REVERB VYUŽÍVAJÍCÍ KONVOLUCI SIGNÁLU S IMPULSNÍ CHARAKTERISTIKOU POSLECHOVÉHO PROSTORU

REVERB DIGITAL AUDIO EFFECT BASED ON CONVOLUTION WITH IMPULSE RESPONSE OF ACOUSTIC ROOM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

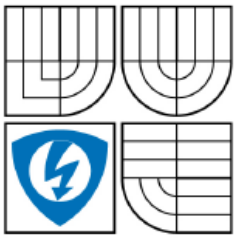
AUTOR PRÁCE
AUTHOR

Bc. VLADIMÍR TICHÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROMÍR MAČÁK

BRNO 2009



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Vladimír Tichý ***ID:*** 83188
Ročník: 2 ***Akademický rok:*** 2008/2009

NÁZEV TÉMATU:

Digitální zvukový efekt typu reverb využívající konvoluci signálu s impulsní charakteristikou poslechového prostoru

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte problematiku šíření zvukových vln v uzavřených akustických prostorech a metody konvoluce při číslicovém zpracování zvukových signálů v reálném čase. Navrhněte a v prostředí Matlab otestujte algoritmus filtrace zvukového signálu lineárním systémem popsaným pomocí impulsní charakteristiky akustického prostoru, kterou získáte měřením v reálných prostorách. Rozšiřte tento algoritmus na dvoukanálovou verzi umožňující simulaci binaurálního poslechu a implementujte jej pomocí technologie VST.

DOPORUČENÁ LITERATURA:

[1] ZÖLZER, U. Digital Audio Signal Processing, 1st ed. New York: McGraw-Hill, Inc., 1997, 290 p. ISBN 0-47-197226-6.
[2] GARCIA, G. Optimal Filter Partition for Efficient Convolution with Short Input/Output Delay. In 113th AES Convention Paper 5660. Los Angeles, USA: Audio Engineering Society, 2002, pp. 1-9.
[3] PICINALI, L. Techniques for the Extraction of the Impulse Response of a Linear and Time-Invariant System. In Proceedings of the DRMN-06: DMRN Doctoral Research Conference 2006. London, Great Britain: University of London, 2006.

Termín zadání: 9.2.2009 ***Termín odevzdání:*** 26.5.2009

Vedoucí práce: Ing. Jaromír Mačák

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

ABSTRAKT

Práce se zabývá simulací poslechového prostoru pomocí jeho impulsní charakteristiky. Popisuje různé modely simulace akustických prostorů, jejich výhody a nevýhody a dále se soustředí právě na fyzikální model, který k simulaci využívá impulsní charakteristiky akustického prostoru. Je uvedena čtveřice nejčastěji používaných metod určených k jejímu měření s uvedením podmínek a omezení jejich použití.

Dále jsou popsány metody výpočtu konvoluce vhodné pro číslicové zpracování signálů v reálném čase. Je provedena analýza výpočetní náročnosti algoritmů pracujících v kmitočtové rovině a vybrané algoritmy jsou implementovány a testovány v prostředí Matlab. Nejefektivnější algoritmus je pak vybrán a implementován do podoby VST plug-in modulu pro simulaci poslechového prostoru v reálném čase.

KLÍČOVÁ SLOVA

hudební efekt, simulace poslechového prostoru, metody výpočtu konvoluce, zpracování signálů v reálném čase, technologie VST, plug-in modul

ABSTRACT

This work deals with a computer simulation of an acoustic room using its impulse response. Two different approaches to the simulation are described with their pros and cons and then the work is focused on the physical approach, which uses room's impulse response during the simulation. Several methods for the extraction of the impulse response of the acoustic room are mentioned with their conditions of use.

The detailed description of various algorithms for a real-time convolution computing is followed by the cost analysis of frequency-domain block convolution algorithms. Several algorithms are chosen, implemented and tested in Matlab environment. Then the most effective of them is chosen to be implemented in VST technology as the plug-in module for real-time room simulation.

KEYWORDS

musical effect, room simulation, convolution computing algorithms, real-time signal processing, VST technology, plug-in module

TICHÝ, V. *Digitální zvukový efekt typu reverb využívající konvoluci signálu s impulsní charakteristikou poslechového prostoru.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 69 s. Vedoucí diplomové práce Ing. Jaromír Mačák.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Digitální zvukový efekt typu reverb využívající konvoluci signálu s impulsní charakteristikou poslechového prostoru“ jsem vypracoval samostatně pod vedením vedoucího diplomové 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 uvedeného diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové 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 § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API	(Application Programming Interface)	*aplikační rozhraní programu*
ASIO	(Audio Streaming Input-Output)	*technologie zpracování zvukových signálů*
BIR	(Binaural Impulse Response)	*binaurální impulsní odezva*
DAW	(Digital Audio Workstation)	*sytém pro digitální zpracování zvuku*
DLL	(Dynamic Linked Library)	*dynamicky sestavovaná knihovna*
DFT	(Discrete Fourier Transform)	*diskrétní Fourierova transformace*
FDL	(Frequency-domain Delay Line)	*osa zpoždění v kmitočtové rovině*
FFT	(Fast Fourier Transform)	*rychlá Fourierova transformace*
FIR	(Finite Impulse Response)	*konečná impulsní odezva*
GUI	(Graphic User Interface)	*grafické uživatelské rozhraní*
HRIR	(Head-Related Impulse Response)	*impulsní odezva vztažená k hlavě*
HRTF	(Head-Related Transfer Function)	*přenosová funkce vztažená k hlavě*
IFFT	(Inverse Fast Fourier Transform)	*zpětná rychlá Fourierova transformace*
IIR	(Infinite Impulse Response)	*nekonečná impulsní odezva*
IR	(Impulse Response)	*impulsní odezva (charakteristika)*
LTI	(Linear Time Invariant)	*lineární časově neproměnný*
madds	(Multiply-Adds)	*dvojice operací součin-součet*
MBMC	(Multiple-Block Convolution with Nonuniform Partition – Minimum Cost)	*algoritmus výpočtu konvoluce s dělením impulsní charakteristiky na bloky různých délek, verze s minimální výpočetní náročností*
MBUL	(Multiple-Block Convolution with Nonuniform Partition – Uniform Load)	*algoritmus výpočtu konvoluce s dělením impulsní charakteristiky na bloky různých délek, verze s rovnoměrným rozložením výpočetní náročností*
MBUP	(Multiple-Block Convolution with Uniform Partition)	*algoritmus výpočtu konvoluce využívající dělení impulsní charakteristiky na bloky stejné délky*
MIDI	(Musical Instruments Digital Interface)	*digitální komunikační rozhraní hudebních nástrojů*
MLS	(Maximum Length Sequence)	*pseudonáhodný signál maximální délky*
SB	(Single-Block Convolution)	*algoritmus výpočtu konvoluce bez dělení impulsní charakteristiky na menší bloky*
SDK	(Software Development Kit)	*sada nástrojů pro vývoj software*

TRM	(Time Reversal Mirror)	*technika časového zrcadla*
VST	(Virtual Studio Technology)	*technologie pro práci s virtuálními studiovými zařízeními*

$*$	operátor lineární konvoluce
$\otimes$	operátor cyklické konvoluce
$\oplus$	operátor cyklické korelace

c	rychlost šíření zvuku
c_0	rychlost šíření zvuku ve vzduchu při teplotě 20°C
$H(z)$	systémová přenosová funkce systému s diskrétním časem
$h(n)$	impulsní charakteristika diskrétního systému
$h_\mathrm{p}(n)$	periodická impulsní charakteristika diskrétního systému
$h(t)$	impulsní charakteristika spojitého systému
$\tilde{m}(n)$, $m(n)$	pseudonáhodný periodický signál maximální délky
$O_\mathrm{FFT}(N)$	výpočetní náročnost rychlé Fourierovy transformace o délce N
$O_\mathrm{OlapAdd}(N)$	výpočetní náročnost spojená s použitím metody přičtení přesahu
$O_\mathrm{SM}(N)$	výpočetní náročnost součinu dvou spektrálních funkcí délek N
$O_\mathrm{SingleFDL}(K)$	výpočetní náročnost algoritmu výpočtu konvoluce s dělením impulsní charakteristiky na bloky stejné délky a s využitím jednoduché FDL
$O_\mathrm{DoubleFDL}(K)$	výpočetní náročnost algoritmu výpočtu konvoluce s dělením impulsní charakteristiky na bloky dvou délek a s využitím dvojité FDL
$R_\mathrm{pmm}(n)$	autokorelační posloupnost
$R_\mathrm{pmy}(n)$	vzájemná korelační posloupnost
T_vz	vzorkovací perioda
$\delta(n)$	Diracův (jednotkový) impuls s diskrétním časem
$\delta_\mathrm{p}(n)$	periodický Diracův (jednotkový) impuls s diskrétním časem
$\delta(t)$	Diracův (jednotkový) impuls se spojitým časem

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Nezbytnou součástí každého nahrávacího studia jsou, kromě jiných, i procesory vytvářející umělý dozvuk. To je dáno způsobem snímání zaznamenávaného zvuku i samotnými akustickými vlastnostmi studia [21], [22]. Technika označovaná jako kontaktní (lokální) snímání, snímající zvuk ve velmi malé vzdálenosti (řádově centimetry), přináší minimální vliv konkrétního akustického prostředí. Navíc také minimalizuje přeslechy z případných dalších zvukových zdrojů. Na druhou stranu však u většiny nástrojů nedokáže zachytit jejich celkový zvukový charakter a je proto nutno počítat s dalšími úpravami, které kromě vytvoření umělého dozvuku zpravidla zahrnují i složité ekvalizační úpravy.

Další možností je plnohodnotné snímání zvuku nástroje ze vzdálenosti cca půl metru a více v relativně zatlumeném akustickém prostředí. To však klade vysoké nároky na akustické vlastnosti studia i na kvalitu záznamového řetězce. Jelikož je barva snímaného nástroje ovlivněna hlavně tzv. počátečními odrazy, největší vliv mají objekty v bezprostřední blízkosti (stěny, podlaha, paravány apod.), lze změnou struktury a charakteru povrchu těchto objektů barvu nástroje značně ovlivnit. Následné zpracování je tedy snazší, zahrnuje obvykle pouze přidání určité míry umělého dozvuku, který dodá záznamu přirozenější charakter.

Nejlepší výsledky přináší komplexní snímání zvuku v homogenním dozvukovém poli koncertního sálu nebo nahrávacího studia. Tento způsob snímání má nejvyšší nároky na kvalitu konkrétního akustického prostoru, avšak nevyžaduje žádné další přidávání umělého dozvuku. Oproti předchozím dvěma technikám, u kterých je umělý dozvuk generován pouze z monofonního nebo stereofonního záznamu zvuku z určité části nástroje, je přirozený dozvuk tvořen nekonečným počtem zvukových odrazů, které se šíří a kombinují v daném akustickém prostoru [22].

Moderní digitální technologie a postupy umožňují velmi věrně napodobit průběh dozvuku ve skutečných prostorech, jako jsou např. koncertní sály, kluby, velká studia nebo běžná prostředí – kancelář, obývací pokoj atp. Oproti dříve používaným prostředkům pro vytváření umělého dozvuku (spirála, dozvuková deska, dozvuková komora) mají digitální jednotky řadu předností. Mezi tu hlavní pak dozajista patří jejich variabilita – parametry dozvuku se dají měnit v širokém rozsahu a umožňují tak simulovat dozvuk prakticky jakéhokoli reálného prostoru nebo i vytvořit takový dozvuk, který nelze v reálných podmínkách napodobit [21]. Kromě hardwarových efektových procesorů se zejména v oblasti DAW (Digital Audio Workstation) prosazují i softwarová řešení, např. v podobě VST plug-in modulů.

Tato práce stručně popisuje charakter dozvuku v přirozeném prostředí a základní druhy přístupu k jeho simulaci. Dále se soustředí na tzv. fyzikální přístup, tj. takový, který napodobuje vlastnosti akustického prostoru na základě jeho impulsní charakteristiky. Jsou zde popsány i běžně používané metody měření impulsní charakteristiky akustických prostorů včetně aspektů jejich použití.

Pro efektivní realizaci výpočetně vysoce náročného fyzikálního přístupu při číslicovém zpracování signálů na osobním počítači je třeba nalézt co nejefektivnější způsob výpočtu konvoluce tak, aby mohla simulace probíhat v reálném čase. Zbytek práce se tedy zabývá hledáním takovéto metody, jejím testováním, optimalizací a implementací do podoby VST plug-in modulu pro zpracování signálů v reálném čase.

1 DOZVUK V PŘIROZENÉM PROSTŘEDÍ

Většinu svého života stráví člověk v prostředí s dozvukem. Ať již jde o rozhovor v kanceláři, koncert v klubu, procházku po ulici nebo v lese, všude slyšíme směs přímého zvuku a odrazů od objektů, které se v daném prostředí nacházejí. Jelikož jsou obecně dráhy odraženého zvuku delší než dráha zvuku přímého, přicházejí k nám odrazy s různými zpožděními a z různých směrů. Těchto odrazů bývá i několik tisíc a lidské vnímání zvuku si s tím naštěstí dokáže poradit. Spíše než aby byl člověk zmaten, odrazů si naopak nemusí vůbec všimnout.

Ve velkých členitých prostorech (kostel, koncertní hala) může dozvuk trvat velmi dlouho a tak vytvořit jakousi „kulisu“ na pozadí, která je značně odlišná od zvuku přímého. V malých uzavřených prostorech může být dozvuk natolik krátký, že jednotlivé odrazy nejsou vnímány samostatně, ale mění výsledný vjem zvuku – jeho hlasitost, barvu a zejména umístění v prostoru [8].

Příkladem ohraničeného prostředí bez dozvuku je tzv. mrtvá komora [21]. Jde o místnost obloženou maximálně zvukově pohltivými materiály, která tak postrádá vlastnosti přirozeného prostředí. Používá se např. k měření směrových charakteristik elektroakustických měničů.

1.1 Základy prostorové akustiky

Uvažujme obecný zvukový zdroj jako zdroj vzruchu, který se v prostoru šíří formou postupného podélného vlnění – zvukové vlny. Vlivem tohoto vzruchu se částice vzduchu v některých místech prostoru navzájem vzdalují či přibližují, čímž vzniká jejich zředění nebo zhuštění (podtlak a přetlak). Zvukové vlny se ve vzduchu šíří rychlostí c, kterou lze určit pomocí vztahu

$$c = c_0 + 0{,}607T\,, \tag{1.1}$$

kde c_0 je rychlost šíření zvuku při teplotě $T = 0°\mathrm{C}$ ($c_0 = 331{,}8$ m/s) a T značí teplotu vzduchu ve stupních Celsia. [19]

Ve volném prostoru se zvuk šíří od zdroje všemi směry volně a jeho šíření můžeme popsat tzv. vlnoplochami (spojnicemi všech míst zvukového pole, které mají v daný okamžik stejné parametry). Uvažujeme-li bodový (nekonečně malý) zdroj zvuku, mají vlnoplochy tvar koule s tím, že v dostatečné vzdálenosti od zdroje již považujeme vlnoplochy za rovinné. V případě zdroje tvaru např. ploché desky jsou vlnoplochy rovinné.

Při dopadu zvukové vlny na překážku dochází v závislosti na vztahu rozměru překážky k vlnové délce zvukové vlny buď k převažujícímu lomu nebo odrazu. Jsou-li rozměry překážky větší než je vlnová délka dopadajícího zvuku, dochází jen k jeho odrazu [19]. Od rovných, hladkých nepružných povrchů se vlny odrážejí pod úhlem, s jakým na překážku dopadly (zákon odrazu), zatímco odrazy od členitých povrchů jsou rozptýleny do mnoha směrů. Současně platí, že vlivem pohlcení části energie dopadajícího vlnění překážkou je intenzita odraženého vlnění I_R vždy menší než intenzita I_0 vlnění dopadajícího na překážku. Intenzita pohlceného vlnění I_A je pak dána vztahem

$$I_\mathrm{A} = I_0 - I_\mathrm{R}\,. \tag{1.2}$$

Na základě těchto hodnot můžeme určit několik koeficientů: [11]

- **koeficient odrazivosti (reflexe) *r***

$$r = \frac{I_R}{I_0}; \tag{1.3}$$

- **koeficient pohltivosti (absorpce) *α***

$$\alpha = \frac{I_A}{I_0}. \tag{1.4}$$

Uvažujeme-li i případ, kdy pohlcená energie není vnitřními ztrátami zcela přeměněna v teplo a vlnění tak může částečně překážkou projít (typicky skrze zeď do druhé místnosti), můžeme uvažovat také **koeficient zvukové průzvučnosti *τ***, který pak bude dán vztahem

$$\tau = \frac{I_T}{I_0}, \tag{1.5}$$

kde I_T je intenzita vlnění, které prošlo překážkou a lze ji určit jako rozdíl

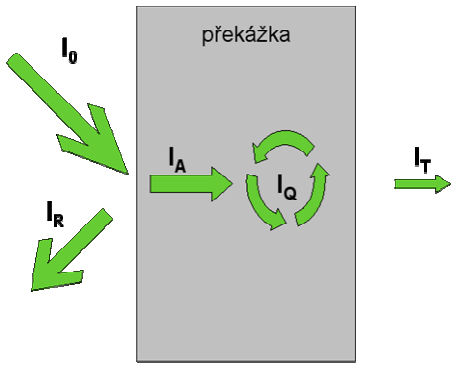
$$I_T = I_A - I_Q, \tag{1.6}$$

přičemž I_Q značí intenzitu vlnění přeměněného v teplo. Celou situaci ilustruje Obr. 1.1.

Ze vztahů (1.2) až (1.4) plyne, že součet $r + \alpha = 1$ a tedy, že je-li hodnota jednoho z koeficientů nulová, druhý nabývá maximální hodnoty 1. V tom případě by zvukové vlnění bylo buď bezezbytku pohlceno (pro $\alpha = 1$) a nebo bezezbytku odraženo (pro $r = 1$). Druhá možnost však v praxi nenastává. Maximální hodnoty 1 mohou nabývat pouze koeficienty α a τ a to v těchto modelových případech:

- $\alpha = 1$ v případě otevřeného okna, kdy všechna dopadající energie projde do venkovního prostoru a žádná se neodráží zpět;
- $\tau = 1$ v případě otevřených dveří, kdy všechna dopadající energie je odražena do druhé místnosti.

Zvukové vlny tedy po každém odrazu postupně slábnou, současně se uplatňuje i pohlcování akustické energie samotným vzduchem (markantní na vyšších kmitočtech) [21]. Teoreticky však znějí nekonečně dlouho, proto se za konec šíření vlny běžně uvažuje okamžik, kdy její intenzita poklesne na úroveň intenzity okolního hluku.



Obr. 1.1: Odraz vlnění od překážky.

1.1.1 Standardní doba dozvuku

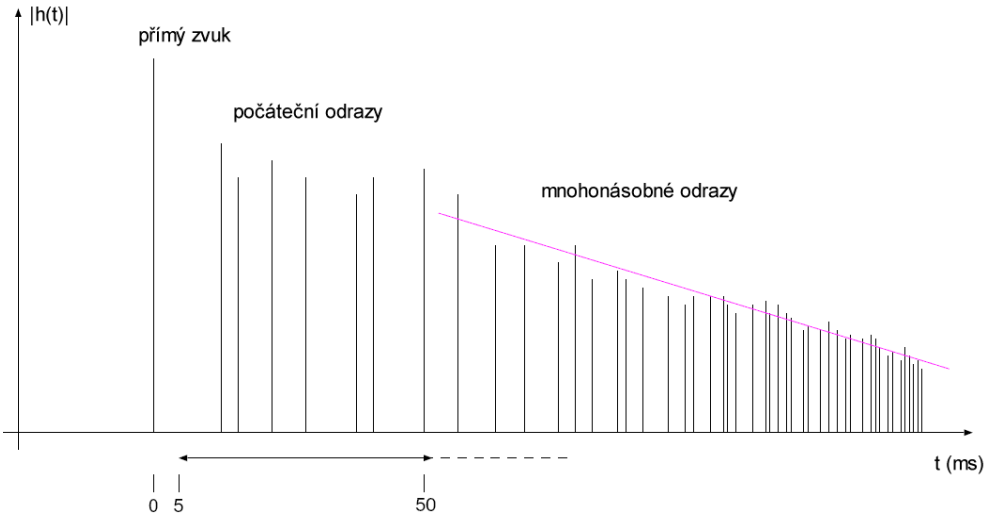
Jako základní parametr dozvuku se uvádí **standardní doba dozvuku** (označován také jako T_{60}), což je doba, za kterou poklesne úroveň odrazů o 60 dB, tedy na jednu miliontinu původní hodnoty. Tuto standardní dobu dozvuku je možno pro uzavřený prostor spočítat pomocí vztahu

$$T_{60} = 0{,}163\frac{V}{\alpha S} = 0{,}163\frac{V}{\sum_n \alpha_n S_n}, \tag{1.7}$$

kde T_{60} je doba dozvuku, V je objem uzavřeného prostoru, S_n je obsah pohltivých ploch a α_n je koeficient pohltivosti těchto ploch [24].

Koeficient pohltivosti daného materiálu je navíc závislý na kmitočtu odrážené vlny (dále závislé i na úhlu dopadu vlny a tvaru povrchu), odražený zvuk má tedy oproti původnímu zcela odlišné zabarvení. Obecně platí, že většina materiálů více odráží nižší kmitočty a je vysledováno, že přirozený dozvuk zpravidla obsahuje jen velmi málo kmitočtů nad hodnotou 5 kHz [21].

Na Obr. 1.2 je zobrazena typická impulsní odezva přirozeného prostředí. Tato se skládá z přímého zvuku, počátečních odrazů a mnohonásobných odrazů. Časový odstup přímého zvuku a prvního počátečního odrazu se označuje jako počáteční zpoždění, které společně s počátečními odrazy přináší nejvíce informací o velikosti daného prostoru [21]. Počet odrazů dále roste s časem až do podoby exponenciálně doznívajícího náhodného signálu (mnohonásobné odrazy).



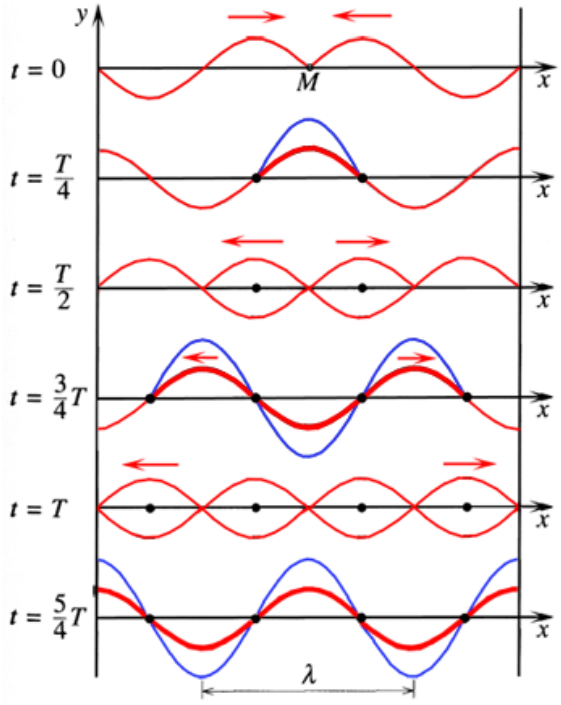
Obr. 1.2: Typická impulsní charakteristika přirozeného prostředí.

1.1.2 Stojaté vlnění

V důsledku interferencí přímých a odražených vln při šíření zvuku v uzavřených prostorách dochází na určitých frekvencích ke vzniku stojatého vlnění, někdy označovaného jako vlastní kmity (módy) [19]. Stojaté vlnění může být příčné i podélné (amplituda kmitů je kolmá, resp. rovnoběžná se směrem šíření vlny). Příčné stojaté vlnění je charakteristické pro pružná pevná tělesa – je např. zdrojem zvuku u strunných nástrojů. Podélné stojaté vlnění může vzniknout v tělesech všech skupenství, která jsou pružná při

změně objemu. Je zdrojem tónů u dechových hudebních nástrojů – vzniká podélným stojatým vlněním (chvěním) vzduchového sloupce.

Vznik příčného stojatého vlnění je naznačen na Obr. 1.3 jako superpozice dvou stejných vlnění postupujících opačným směrem (zobrazeno pro okamžiky celistvých násobků čtvrtiny periody těchto vlnění).



Obr. 1.3: Vznik a časový průběh příčného stojatého vlnění. [20]

Bod, který zůstává po celou dobu v klidu (tj. jeho amplituda výchylky je nulová), je označován jako uzel. Naopak bod, který kmitá s maximální amplitudou se označuje jako kmitna. Vzdálenost dvou sousedních uzlů (resp. kmiten) je rovna $\lambda/2$. Uzel leží vždy na poloviční vzdálenosti mezi dvěma kmitnami a naopak.

Stojaté vlnění je oproti vlnění postupnému, při kterém kmitají všechny jeho body se stejnou amplitudou výchylky ale s rozdílnou fází, odlišné v tom, že při stojatém vlnění kmitají všechny body mezi dvěma uzly se stejnou fází, ovšem s rozdílnou amplitudou výchylky. Postupným vlněním je přenášena energie, kdežto vlnění stojaté energii nepřenáší – pouze se cyklicky mění potenciální energie pružnosti v energii kinetickou a zpět.

V praxi se hodnoty kmitočtů vlastních kmitů určí jako řešení vlnové rovnice pro prostor tvaru kvádru o rozměrech l_x, l_y a l_z jako

$$f_n = \frac{c}{2}\sqrt{\left(\frac{n_x}{l_x}\right)^2 + \left(\frac{n_y}{l_y}\right)^2 + \left(\frac{n_z}{l_z}\right)^2}\,, \tag{1.8}$$

kde n_x, n_y, n_z jsou celočíselné násobky půlvln. Vlastní kmity se dále dají dělit podle toho, od kolika stěn se odrážejí. Uvádějí se tři typy: ***osové*** (odraz pouze od dvou protilehlých stěn, tzn. pouze jedno z čísel n_x, n_y, n_z je nenulové); ***tangenciální*** (odraz pouze od čtyřech stěn, tzn. dvě z čísel n_x, n_y, n_z jsou nenulová) a ***kosé*** (všechna čísla n_x, n_y, n_z nenulová).

U velkých místností leží kmitočty vlastních kmitů na velmi nízkých hodnotách frekvence, u malých místností jsou pak její nejnižší vlastní kmity posunuty mnohem výše

[24]. Nejnižší frekvence vlastního kmitu může být na základě znalosti největšího rozměru zkoumaného prostoru určena pomocí vztahu: [19]

$$f_{\min} = \frac{c}{2l_{\max}}, \tag{1.9}$$

kde $l_{\max}$ značí největší rozměr zkoumaného prostoru.

Vysoký počet vlastních kmitů ležících v slyšitelném pásmu (cca 16 Hz – 16 kHz) je základním předpokladem pro kvalitní přenos zvuku v uzavřeném prostoru. Toho lze dosáhnout v případě, kdy se nejnižší vlastní kmit nachází hluboko pod nejnižším přenášeným kmitočtem. Tato situace však platí pouze u velkých prostorů (cca nad 200 m^2). Problém s kvalitním přenosem jsou největší právě v oblasti nízkých frekvencí, a to díky tomu, že hustota vlastních kmitů se s rostoucím kmitočtem zvyšuje. V této oblasti, kde je vlnová délka zvukové vlny srovnatelná s rozměry prostoru, jsou hodnoty kmitočtů vlastních kmitů rozloženy nerovnoměrně a současně navzájem značně vzdáleny. [19]

Poznatků z oblasti stojatého vlnění se v praxi hojně využívá v tzv. absorpčních rezonátorech, které umožňují dosáhnout velmi dobré zvukové pohltivosti v žádaném rozsahu frekvencí. Jejich konstrukce vychází z faktu, že při vzniku stojatého vlnění (i na pouze přechodnou dobu) je v místě odrazu vlny (zeď apod.) vždy uzel, kde je energie vlny nulová. Maximum energie je naopak v kmitně, která je od uzlu vzdálená o $\lambda/4$; vložíme-li do tohoto místa vhodnou pohlcující látku, dosáhneme žádané pohltivosti pro danou frekvenci. [11]

1.2 Účinky počátečních odrazů na zvukový vjem člověka

Účinky počátečních odrazů můžeme zkoumat tak, že uvažujeme vjem přímého zvuku a pouze jednoho odrazu. Této situace lze snadno dosáhnout v mrtvé komoře nebo za použití sluchátek. Bude-li zpoždění odrazu od přímého zvuku větší než cca 80 ms a oba zvuky přicházejí frontálně, odražený zvuk bude vnímán jako přesné echo (ozvěna) zvuku přímého. Při zmenšování doby zpoždění odrazu od přímého zvuku se tyto začnou „slévat“ do jediného zvukového vjemu, u kterého se bude měnit jeho barva. Odražený zvuk může také značně zvýšit hlasitost zvuku přímého.

Pokud odraz přichází k posluchači z laterálního (bočního) směru, může značně ovlivnit prostorový charakter zvuku. Malé zpoždění (do 5 ms) může zapříčinit zdánlivý posun zvukového zdroje. Větší zpoždění může zvětšit zdánlivou velikost zdroje, což záleží také na kmitočtovém obsahu zvuku, nebo dokonce navodit pocit obklopení zvukem. [8]

2 SIMULACE AKUSTICKÝCH PROSTORŮ

Dle [8] se z hlediska zpracování signálů k úloze simulace akustických prostorů přistupuje tak, že je zavedena souvislost mezi zkoumaným prostorem se zvukovými zdroji a posluchači a systémem se vstupy a výstupy, které odpovídají hodnotám vhodné fyzikální veličiny v určitých bodech prostoru. Například uvažujme systém s jedním vstupem odpovídajícím zvukovému zdroji a dvěma výstupy, které odpovídají okamžitým hodnotám akustického tlaku působících na ušní bubínky posluchače.

Za předpokladu, že na zkoumaný prostor můžeme nahlížet jako na lineární časově invariantní systém (dále LTI) [16], můžeme nalézt takovou přenosovou funkci, která kompletně popisuje transformaci hodnoty akustického tlaku ze zvukového zdroje na hodnoty působící na posluchače (popisuje šíření zvukových paprsků od zdroje zvuku k posluchači). Můžeme tedy simulovat vliv prostoru konvolucí vstupního signálu s *binaurální impulsní odezvou* (díle BIR), což je dvojice impulsních odezev prostoru příslušejících pravému a levému uchu posluchače. Výstupní signály pak určíme jako:

$$\begin{aligned} y_{\mathrm{L}}(t) &= \int_0^{\infty} h_{\mathrm{L}}(\tau) x(t-\tau)\,\mathrm{d}\tau, \\ y_{\mathrm{R}}(t) &= \int_0^{\infty} h_{\mathrm{R}}(\tau) x(t-\tau)\,\mathrm{d}\tau, \end{aligned} \tag{2.1}$$

kde $h_{\mathrm{L}}(t)$ a $h_{\mathrm{R}}(t)$ jsou impulsní odezvy systému pro levé, resp. pravé ucho; $x(t)$ je vstupní signál a $y_{\mathrm{L}}(t)$ a $y_{\mathrm{R}}(t)$ jsou výstupní signály pro levé, resp. pravé ucho.

Vlastnosti systému jsou pak nejvíce ovlivněny následujícími parametry simulovaného prostoru: rozměry, objem a geometrické uspořádání uzavřeného prostoru, materiál stěn a objektů v prostoru (viz kapitola 1.1), poloha zdroje zvuku a posluchače. [3]

2.1 Druhy simulací

Základní rozlišení druhů simulací akustických prostor můžeme dle [3] učinit na základě toho, kolik výstupů má použitý simulační systém. Pro zjednodušení v následujícím výkladu uvažujeme systém s pouze jedním vstupem, reprezentujícím bodový zdroj zvuku.

2.1.1 Jednokanálová simulace

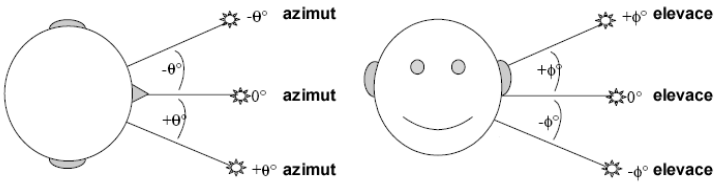
Systém, kterým simulujeme akustický prostor, má pouze jeden výstup. Popisuje tedy šíření zvukových vln od bodového zdroje do místa příjmu, které je reprezentováno pouze jedním bodem v prostoru. Tento druh simulace ovšem z principu neudává žádné informace o směru příchozích vln a proto se používá hlavně jako výchozí bod pro další druhy modelů. Je ho totiž možno velmi jednoduše popsat např. jednou impulsní odezvou.

2.1.2 Dvoukanálová simulace (binaurální)

Akustický prostor je simulován systémem se dvěma výstupy, popisujícím šíření zvuku od bodového zdroje k příjemci, který je nejčastěji specifikován polohou hlavy v daném prostoru. Jeden z výstupů je pak určen levému uchu a druhý pravému. Tímto způsobem je tedy možné přesně popsat akustické vlastnosti simulovaného prostoru vzhledem k poloze posluchače v něm a k samotné poloze jeho hlavy, která je vzhledem k poloze zdroje

popsána úhlem natočení ve vodorovném směru (azimut) a elevací, tedy úhlem natočení ve směru svislém.

Předpokladem věrné simulace akustického prostoru touto metodou je použití sluchátek při reprodukci zvukového snímku. Simulovaný akustický signál totiž musí vznikat v pokud možno minimální vzdálenosti od ucha. Lze použít i klasické reproduktorové soustavy ve stereofonním uspořádání, ale kvalita simulace je tímto snížena.



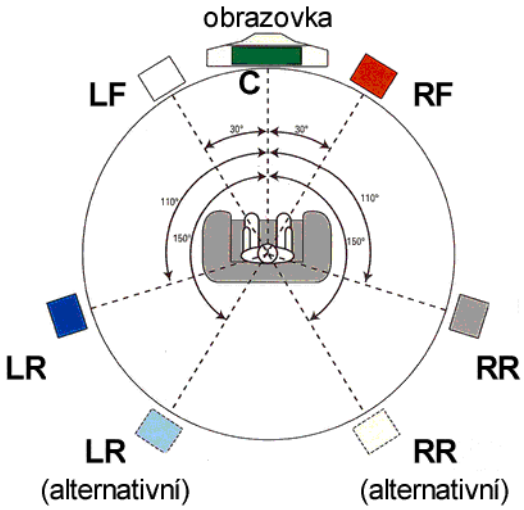
Obr. 2.1: Určení hodnoty azimutu a elevace.

2.1.3 Vícekanálová simulace

Systém simulující akustický prostor má více výstupů – typicky 3, 5, 7 – podle toho, kterému standardu vícekanálového zvuku má systém odpovídat. Na Obr. 2.2 je zobrazeno uspořádání LF-C-RF-LR-RR, tedy standardu s pěti soustavami.

Princip vícekanálové simulace akustického prostoru spočívá v simulaci celého akustického prostoru v jiném akustickém prostoru. V ideálním případě by měl být tento druhý prostor dokonale zatlumen (mrtvá komora), ovšem běžně se jedná jen o částečně zatlumené akustické prostory jako je například obývací pokoj apod. Každá z reproduktorových soustav představuje pro danou konfiguraci zvukové paprsky přicházející z daného směru, tj. jisté výseče pomyslného kruhu okolo posluchače. Nový akustický prostor je takto vytvořen součinností všech soustav, signál není reprodukován v blízkosti ucha, jak tomu bylo u dvoukanálové simulace.

Tímto způsobem nelze dosáhnout opravdu přesné simulace zamýšleného akustického prostoru vzhledem k poloze posluchače a jeho hlavy. Jde o kompromisní řešení umožňující vytvořit zvukové pole pro více posluchačů najednou. Vícekanálová simulace se používá zejména pro zvukový doprovod filmů.



Obr. 2.2: Doporučené uspořádání reproduktorových soustav pro vícekanálový zvuk LF-C-RF-LR-RR.

Na algoritmus simulace poslechového prostoru můžeme nahlížet jako na lineární systém, který simuluje vstupně-výstupní chování skutečného nebo virtuálního prostoru. K samotné realizaci tohoto systému můžeme přistupovat dvěma různými způsoby. [8]

2.2 Fyzikální model simulace akustických prostor

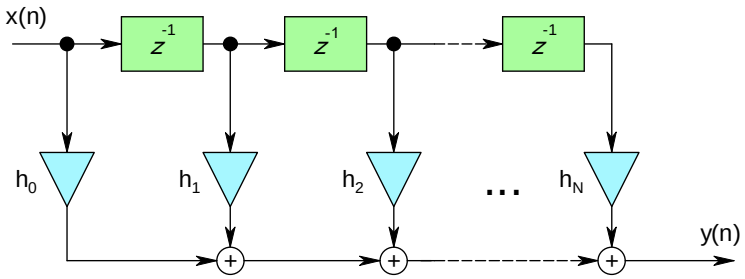
Tento přístup se snaží o přesnou simulaci šíření zvuku od zdroje k příjemci v daném akustickém prostoru. Nejjednodušším způsobem je využití již zmíněné binaurální impulsní odezvy (kapitola 2) pro daný prostor a polohu zdroje a příjemce. Výstupní signály se pak určí pomocí konvoluce BIR se vstupním signálem (rovnice (2.1)).

Problematika měření impulsních charakteristik reálných prostorů bude podrobně popsána v kapitole 3.

V případě, že chceme simulovat nějaký virtuální akustický prostor, snažíme se určit jeho impulsní odezvu pouze na základě jeho fyzikálních vlastností. To vyžaduje detailní znalosti geometrických vlastností prostoru, parametrů všech povrchů místnosti i objektů v něm a pozic zvukového zdroje a posluchače. Na základě těchto znalostí je možné aplikací zákonitostí šíření zvukových vln prostředím a jejich interakce s překážkami určit, jak se bude zvuk ve zkoumaném prostoru šířit. Tento proces se označuje *auralizace* [12] a v praxi využívá pro výpočty metodu konečných prvků.

V případě práce s binaurálním modelem virtuálního akustického prostoru nám stačí výše zmíněným procesem zjistit pouze jednokanálovou impulsní odezvu. Na ni po té aplikujeme HRTF (Head-Related Transfer Function) [4], čímž získáme dvojici impulsních odezev pro pravé, resp. levé ucho.

Takto získané impulsní odezvy jsou pak jednoduše realizovány pomocí filtrů typu FIR (Finite Impulse Response) [16]. Struktura filtru typu FIR je zobrazena na Obr. 2.3. Pro délku impulsní odezvy vyšší než cca 200 vzorků [3] (literatura [7] dokonce uvádí hodnotu 64 vzorků) je již však přímý výpočet konvoluce pomocí FIR filtru neefektivní, proto se pro simulace akustických prostor využívá algoritmus rychlé konvoluce. Tato problematika bude přiblížena v kapitole 4.



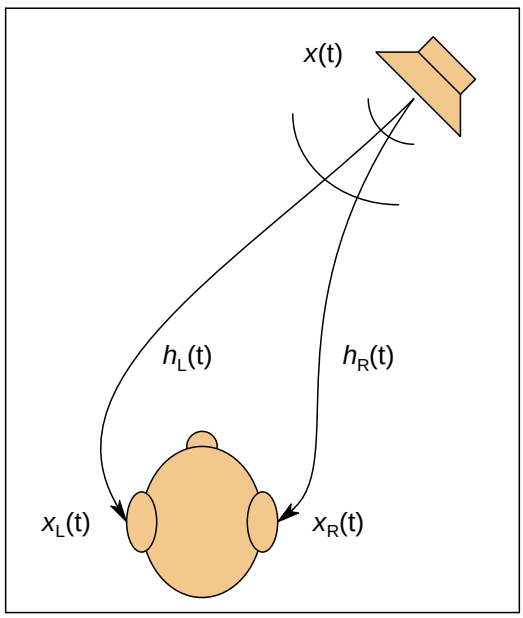
Obr. 2.3: Struktura filtru FIR.

2.2.1 Head-Related Transfer Function

HRTF, nebo-li přenosová funkce vztažená k hlavě, popisuje, jak je zvuková vlna, popsaná svým kmitočtem a směrem, ze kterého přichází, před dopadem na ušní bubínek ovlivněna (filtrována) vlivem vnějšího ucha, hlavy a horní části trupu. Uplatňují se zde zejména mechanismy ohybu a odrazu zvukové vlny. Specifická míra filtrace těchto částí těla, závislá na kmitočtu vlny a zejména na směru, ze kterého vlna přichází, značně napomáhá schopnosti lidského sluchu lokalizovat zvukový zdroj. [4]

HRTF jsou složité funkce kmitočtu a třech sférických veličin: vzdálenosti, elevace a azimutu (Obr. 2.1). Jedna z metod, které se používají ke zjištění této přenosové funkce pro určitou lokaci zdroje, spočívá ve změření HRIR (Head Related Impulse Response), tedy impulsní odezvy vztažené k hlavě. Přenosovou funkci HRTF pak získáme jako Fourierovu transformaci HRIR (pro každé ucho zvlášť).

Měření se provádí za pomoci speciální figuríny v mrtvé komoře z důvodu minimalizace vlivu dozvuku na hledanou charakteristiku. Tato figurína, tzv. KEMAR, simuluje tvar horní části lidského těla a obsahuje speciální měřící mikrofony umístěné v uších, kterými jsou HRIR měřeny. Tento proces je naznačen na Obr. 2.4.



Obr. 2.4: Měření HRIR za účelem výpočtu HRTF.

2.3 Perceptuální model

Tento přístup, narozdíl od předchozího, nepracuje přímo s impulsní charakteristikou akustického prostoru. Simuluje pouze jeho specifické vlastnosti, které mají vliv na vjem tohoto prostoru člověkem. Pro potřeby tohoto přístupu je nezbytné prostor všech vjemů způsobených šířením zvuku v daném akustickém prostoru rozdělit do N nezávislých dimenzí. Může-li každý z vjemů popsat určitou fyzikální vlastností akustického prostoru, je možno navrhnout digitální filtr s N parametry, které pak reprezentují tyto fyzikální vlastnosti a tak simulují N nezávislých vjemů [8], [3].

Pokud bychom tedy chtěli simulovat konkrétní akustický prostor, změřili bychom jeho impulsní charakteristiku a její analýzou určili N parametrů, které bychom použili jako vstupní parametry nějakého univerzálního algoritmu. Tento algoritmus by pak produkoval dozvuk nerozeznatelný od skutečného, i když změřená impulsní charakteristika akustického prostoru by se od té, která je reprezentována N parametry, mohla lišit.

Hlavním přínosem perceptuálního přístupu je snížení výpočetní náročnosti simulačního algoritmu. Mezi jeho další výhody patří: [8]

- Realizace algoritmu simulujícího dozvuk může být založena na použití zpožďovacích linek a filtrů typu IIR (Infinite Impulse Response), což je oproti filtrům typu FIR mnohem efektivnější.
- Je umožněna změna všech N parametrů v reálném čase. Tato změna se projeví bez znatelného zpoždění (doba zpoždění u fyzikálního modelu je dána délkou impulsní charakteristiky akustického prostoru). Parametry dále nemusejí být korelovány (v reálných prostorech často jsou).
- Teoreticky je možno mít pouze jeden univerzální algoritmus, kterým by bylo možno realizovat jakýkoli dozvuk.

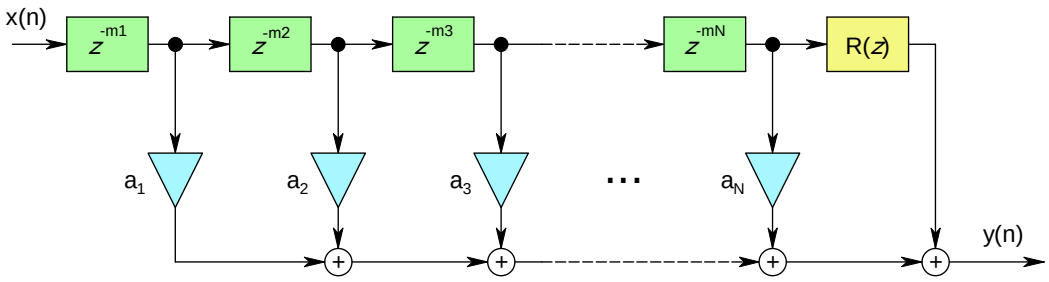
Nevýhodou tohoto přístupu ale je, že ne vždy nabízí jednoduchý způsob, kterým by šlo snadno změnit určitou fyzikální vlastnost simulovaného prostoru. Dále je nutné si uvědomit, že použitím perceptuálního modelu nikdy nedosáhneme opravdu přesné simulace daného akustického prostoru. Můžeme se k ní pouze přiblížit.

Fakt, že prvotní a mnohonásobné odrazy (Obr. 1.2) mají různé vlastnosti i dopad na vjem akustického prostoru člověkem vede k tomu, že perceptuální model má zpravidla dvě základní podčásti. Každá z nich pak používá k simulaci prvotních, resp. mnohonásobných odrazů, svých specifických algoritmů.

2.3.1 Simulace prvotních odrazů

Pokud opomeneme vliv rozptylu a pohlcování zvuku při jeho odrazech a šíření akustickým prostorem, můžeme z počátku pomyslné impulsní charakteristiky vybrat pouze nejvýznamnější koeficienty, které pak budou reprezentovat prvotní odrazy. Pak lze takovýto filtr jednoduše realizovat pomocí filtru typu FIR používajícím takových zpoždění, která odpovídají časům, ve kterých jednotlivé zvukové odrazy k posluchači dorazí.

Prvním modelem realizujícím výše zmíněný postup byl Schroederův základní model uvedený na Obr. 2.5. Jednotlivá zpoždění m_i odpovídají odstupům mezi vnímanými odrazy a koeficienty zesílení a_i modelují klesající charakter hlasitosti dozvuků. Schroeder ve svém modelu uvedl i filtr $R(z)$, který realizuje mnohonásobné odrazy.

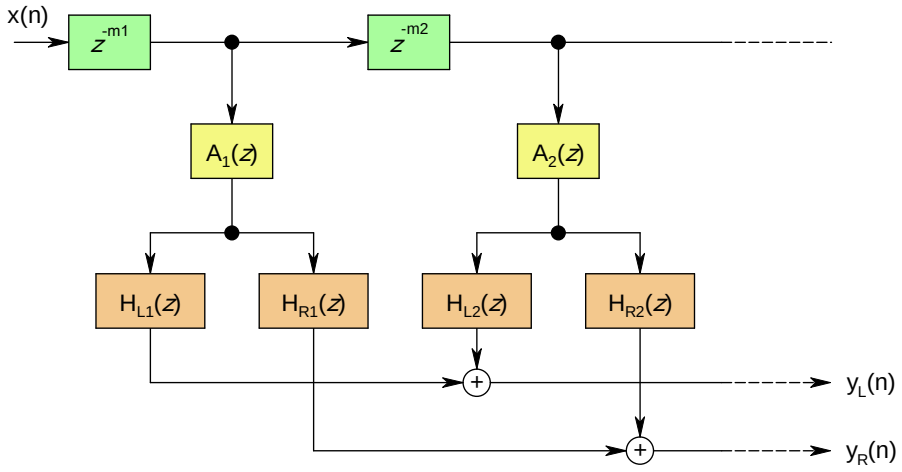


Obr. 2.5: Schroederův základní model simulace prvotních odrazů.

Tento model je velmi jednoduchý ale na druhou stranu má mnoho nedostatků. Absorpční ztráty vzniklé odrazem zvuku od různých materiálů, reprezentovány koeficienty a_i, neuvažují kmitočtovou závislost absorpce (kap. 1.1). Model je pouze jednokanálový a tak neumožňuje zvukovou lokalizaci. To vše spolu s malým počtem odrazů způsobuje dojem nepřirozenosti výsledné simulace.

Tyto nedostatky se Schroeder pokusil odstranit zdokonalením svého základního modelu, jeho rozšířená verze je uvedena na Obr. 2.6. Kmitočtově závislé absorpční ztráty při odrazech zvuku od překážek i při jeho samotném šíření vzduchem reprezentují kmitočtově závislé filtry s přenosovými funkcemi $A_i(z)$ pro každý z výstupů zpožďovací linky. Model je dvoukanálový, umožňuje tedy přesnou lokalizaci každého z odrazů. K tomu se používají HRTF (kap. 2.2.1) odpovídající směru, odkud odraz přichází.

Rozšířený Schroederův model dále řeší i problém s nepřirozeně znějícím dozvukem, k tomu ale potřebuje řádově desítky výstupů ze zpožďovacích linek. V tom případě značně roste výpočtová náročnost až k mezi použitelnosti tohoto modelu pro simulaci v reálném čase.



Obr. 2.6: Schroederův rozšířený model.

2.3.2 Simulace mnohonásobných odrazů

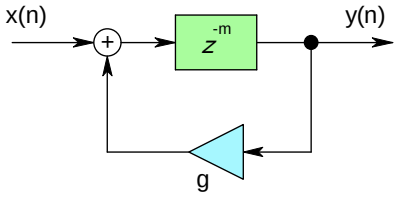
Následující podkapitola pojednává o algoritmech simulace mnohonásobných odrazů. Ty budou představeny v chronologickém pořadí tak, jak vznikly. První algoritmy byly založeny na použití hřebenového filtru a fázovacího článku, novější algoritmy pak používají zpětnovazebních zpožďovacích sítí. [8]

Schroederovy modely

První algoritmy simulující dozvuk byly navrženy již v šedesátých letech 20. století německým vědcem M. R. Schroederem. Jeho původní návrhy byly založeny na hřebenových filtrech a fázovacích článcích. Blokové schéma hřebenového filtru je zobrazeno na Obr. 2.7. Jeho přenosová funkce je

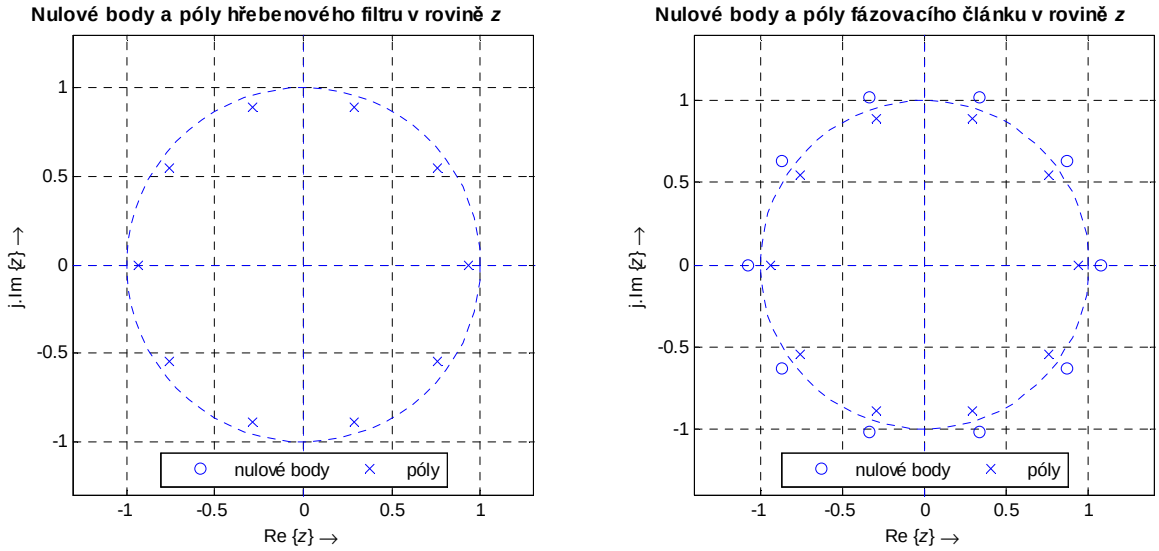
$$H(z) = \frac{z^{-m}}{1 - gz^{-m}}, \tag{2.2}$$

kde m značí délku zpoždění ve vzorcích a g je zesílení ve zpětné vazbě.

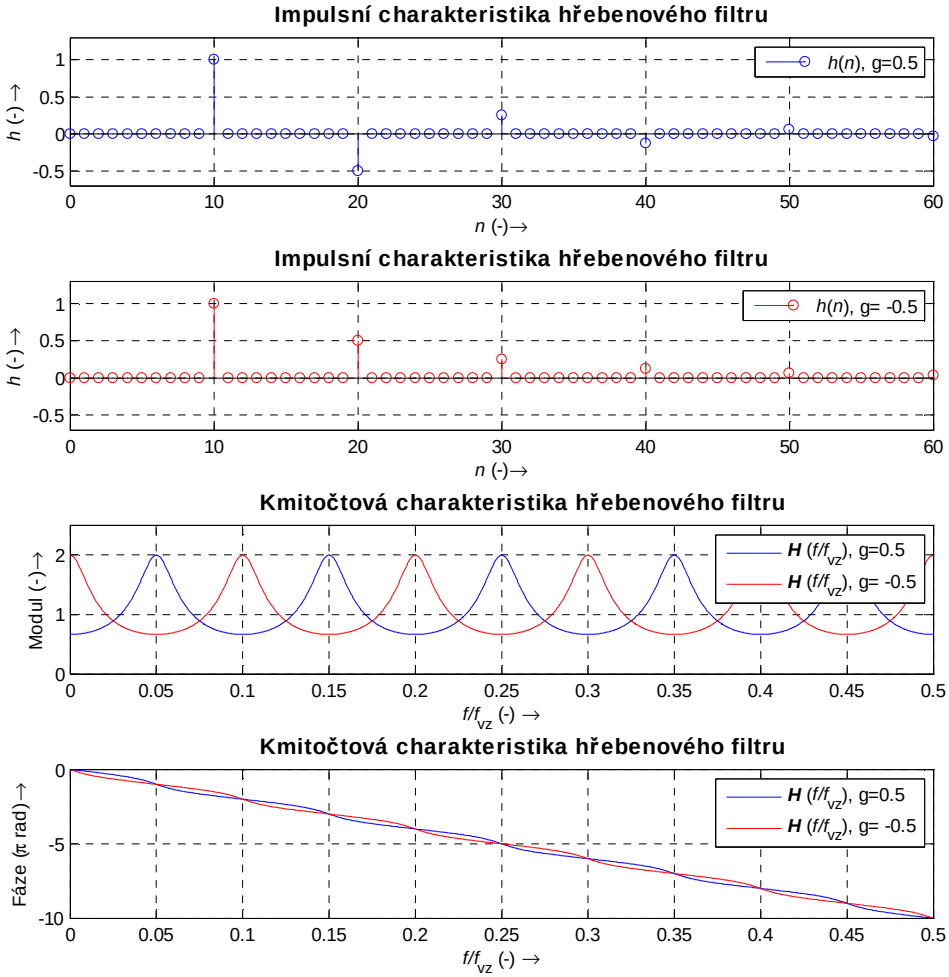


Obr. 2.7: Blokové schéma obecného hřebenového filtru.

Impulsní a kmitočtová charakteristika hřebenového filtru pro hodnoty zpoždění $m = 10$ a zesílení zpětné vazby $g = \pm 0{,}5$ jsou znázorněny na Obr. 2.9. Můžeme vidět, že impulsní charakteristika má tvar exponenciálně klesající posloupnosti impulzů navzájem vzdálených právě o m vzorků. Kmitočtová charakteristika pak připomíná hřeben s m vrcholy rozmístěnými periodicky na kmitočtech odpovídajících kmitočtům pólů – viz Obr. 2.8.

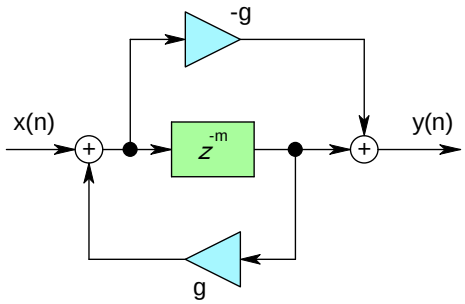


Obr. 2.8: Rozložení nulových bodů a pólů hřebenového filtru a fázovacího článku v rovině z.

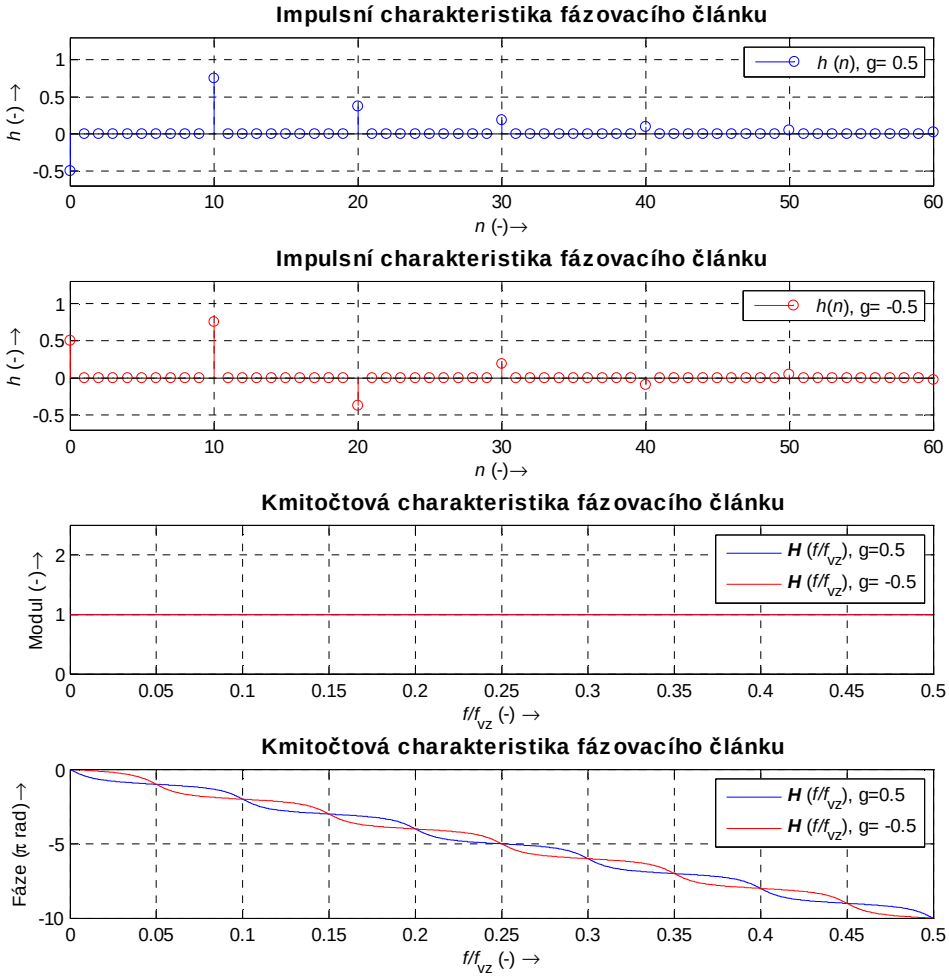


Obr. 2.9: Impulsní a kmitočtová charakteristika hřebenového filtru (m = 10, g = ± 0,5).

Schroeder dále zjistil, že hřebenový filtr lze snadno upravit tak, aby jeho modulová kmitočtová charakteristika nabývala pro všechny kmitočty hodnoty 1. Toho dosáhl smícháním výstupního signálu hřebenového filtru s jeho vstupním signálem, jak je naznačeno na Obr. 2.10.



Obr. 2.10: Blokové schéma obecného fázovacího článku.



Obr. 2.11: Impulsní a kmitočtová charakteristika fázovacího článku (m = 10, g = ± 0,5).

Takto vzniklý filtr se v angličtině označuje jako „allpass“ právě díky tomu, že má pro všechny kmitočty jednotkový přenos. Této vlastnosti je dosaženo kompenzací pólů přenosové funkce fázovacího článku nulovými body v rovině z (Obr. 2.8).

Pokud bychom uvažovali systém realizovaný pouze jedním hřebenovým filtrem, dobu dozvuku T_{60} můžeme určit ze vztahu

$$\frac{20\log(g)}{mT_{vz}} = \frac{-60}{T_{60}}, \tag{2.3}$$

kde T_{vz} je vzorkovací perioda.

Pro danou dobu dozvuku tedy musíme volit poměr doby zpoždění m a zesílení zpětné vazby g. Pro krátké doby zpoždění, které vedou k rychlému výskytu odrazů ve výstupním signálu, má modulová kmitočtová charakteristika tvar navzájem velmi vzdálených úzkých špiček. Tyto špičky odpovídají kmitočtům, ze kterých jsou tvořeny odrazy; zatímco kmitočty ležící mezi těmito špičkami v odrazech zastoupeny nebudou. Tak je výstupní signál výrazně, obvykle nepříjemně, zabarven. Hustotu špiček v modulové kmitočtové

charakteristice můžeme zvýšit změnou doby zpoždění k vyšším hodnotám, to ale vede ke snížení hustoty odrazů v časové oblasti. Následkem toho jsou odrazy vnímány samostatně jako souhrn jednotlivých ozvěn. [8]

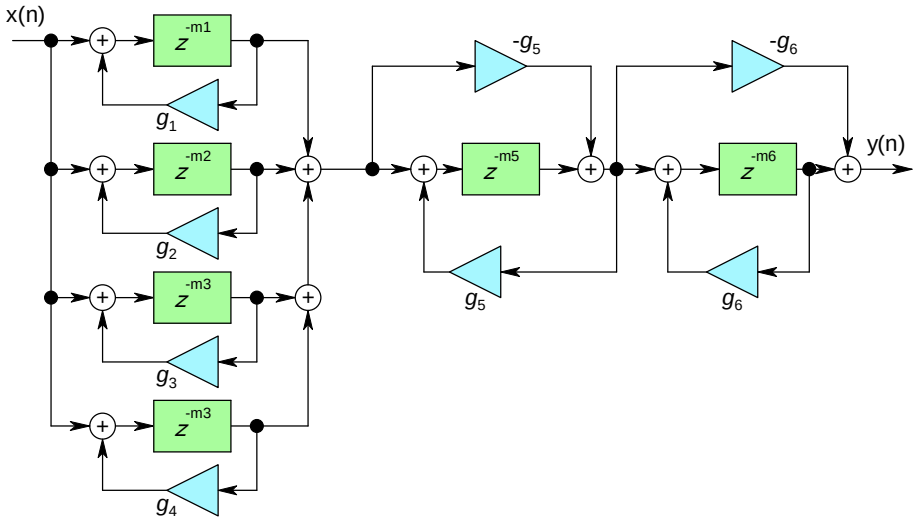
Dalo by se očekávat, že nahrazením hřebenového filtru fázovacím článkem, který má jednotkový přenos pro všechny kmitočty (Obr. 2.9 a Obr. 2.11), lze tento problém snadno vyřešit. Opak je pravdou – výsledek by zněl velice podobně. To je způsobeno vlastnostmi lidského sluchu, který provádí krátkodobou kmitočtovou analýzu, kdežto vlastnosti fázovacího článku jsou definovány pro analýzu v dlouhé (ideálně nekonečné) době.

Sériovou kombinací fázovacích článků můžeme hustotu odrazů značně zvýšit – každý odraz generovaný prvním filtrem totiž vytvoří další odrazy ve filtru následujícím. Pro sériovou kombinaci naopak nejsou vůbec vhodné hřebenové filtry – jejich kombinace by totiž propustila pouze ty kmitočty, které leží v překrývajících se částech špiček v modulové kmitočtové charakteristice. Naproti tomu je možno dát do série libovolný počet fázovacích článků a přenos bude stále jednotkový. Jsou tedy vhodné ke zvýšení hustoty odrazů bez vlivu na modulovou kmitočtovou charakteristiku systému.

Hřebenové filtry je vhodné kombinovat paralelně, výsledná kmitočtová charakteristika totiž obsahuje součet špiček všech filtrů. Aby se špičky nepřekrývaly, je nutno volit nesoudělné hodnoty zpoždění jednotlivých filtrů. Navíc, výsledná hustota odrazů je také dána součtem příspěvků všech filtrů – teoreticky můžeme dosáhnout dostatečné hustoty odrazů bez nepříjemného zabarvení kombinací dostatečného počtu hřebenových filtrů.

Schroeder navrhl systém složený z paralelní kombinace čtyř hřebenových filtrů a série dvou fázovacích článků (Obr. 2.12). Takto sestavený systém pracuje tak, že paralelní kombinace hřebenových filtrů produkuje množinu exponenciálně doznívajících odrazů a fázovací články zvyšují jejich hustotu. Zpoždění hřebenových filtrů je vhodné zvolit v rozmezí 30–45 ms, zatímco u fázovacích článků jsou vhodné hodnoty nižší (řádově několik ms). Hodnoty zesílení zpětné vazby je u fázovacích článků vhodné volit cca 0,7; pro hřebenové filtry lze hodnoty g_i dopočítat podle zvolené doby dozvuku jako

$$g_i = 10^{\left(\frac{-3m_iT_{vz}}{T_{60}}\right)}. \tag{2.4}$$



Obr. 2.12: Schroederův systém složen z paralelní kombinace hřebenových filtrů a sériové kombinace fázovacích článků.

3 METODY MĚŘENÍ IMPULSNÍ CHARAKTERISTIKY AKUSTICKÉHO PROSTORU

Při číslicovém zpracování signálů můžeme ke zjištění přenosové funkce systému s výhodou použít Diracova impulsu. Problém však nastane, pokud bychom chtěli podobný princip uplatnit v analogové oblasti. V praxi totiž není vůbec možné Diracův impuls vygenerovat. Z tohoto důvodu vzniklo v uplynulých letech několik technik, pomocí kterých je možno získat impulsní charakteristiku lineárních časově invariantních systémů.

3.1 Vstupní znalosti

Před samotným výčtem metod určených k měření impulsní charakteristiky akustického prostoru je vhodné osvětlit několik základních pojmů.

3.1.1 Vlastnosti lineárního časově invariantního systému

Pod pojmem systém si můžeme obecně představit nějaký blok, černou krabičku apod., který určitým daným způsobem provádí transformaci vstupního signálu na signál výstupní podle vztahu

$$y(t) = F\{x(t)\}. \tag{3.1}$$

Má-li být systém **lineární**, musí pro něj platit princip superpozice. Ten říká, že pokud je vstupní signál složen z více složek, transformace celého signálu je rovna součtu transformací těchto složek. Jsou-li tedy

$$\begin{aligned} y_1(t) &= F\{x_1(t)\}, \\ y_2(t) &= F\{x_2(t)\}, \end{aligned} \tag{3.2}$$

pak

$$y(t) = F\{ax_1(t) + bx_2(t)\} = aF\{x_1(t)\} + bF\{x_2(t)\} = ay_1(t) + by_2(t), \tag{3.3}$$

kde *a, b* jsou obecně komplexní konstanty.

Časově invariantní (neparametrický, stacionární) systém nemění v čase vlastnosti prováděné transformace. Platí tedy

$$\begin{aligned} y(t) &= F\{x(t)\}, \\ y(t-\tau) &= F\{x(t-\tau)\}. \end{aligned} \tag{3.4}$$

3.1.2 Diracův impuls

Diracův impuls $\delta(t)$, též označovaný jako jednotkový impuls (unit impulse, delta function) představuje tzv. zobecněnou funkci neboli distribuci. Jeho definice neurčuje jak impuls probíhá nebo vypadá, ale jaké má účinky. Nechť $f(t)$ je funkce spojitá v bodě $t = \tau$. Diracův impuls $\delta(t)$ se pak vyznačuje touto vlastností: [17]

$$\int_{-\infty}^{\infty} f(t)\delta(t-\tau)\mathrm{d}t = f(\tau). \tag{3.5}$$

V číslicových systémech se můžeme setkat s definicí Diracova impulsu v podobě

$$\delta(n) = \begin{cases} 1 \text{ pro } n = 0, \\ 0 \text{ pro } n \neq 0. \end{cases} \tag{3.6}$$

Často se vychází z představy, že jednotkový (Diracův) impuls je nekonečně vysoký obdélníkový impuls s nekonečně malou šířkou, jehož výška je rovna převrácené hodnotě šířky Δ (viz Obr. 3.1). Přesná realizace však není možná, protože Diracův impuls má nekonečnou energii.

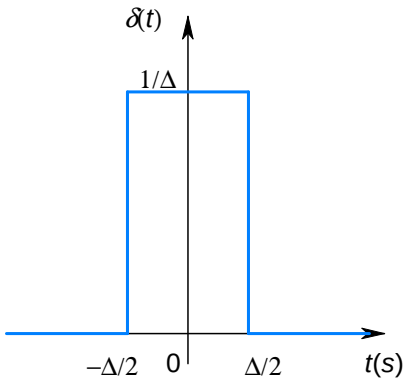
Energie Diracova impulsu je rovnoměrně rozložena přes všechny kmitočty, modulová kmitočtová charakteristika tedy nabývá konstantní hodnoty 1. Proto je pro popis lineárního časové invariantního systému (LTI) možno použít jeho odezvu na jednotkový impuls (impulsní charakteristiku)

$$h(t) = F\{\delta(t)\}. \tag{3.7}$$

Výstupní signál takového systému pak lze snadno určit konvolucí vstupního signálu s impulsní charakteristikou systému jako

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)\mathrm{d}\tau, \tag{3.8}$$

kde symbol * značí operaci konvoluce (v tomto případě dvou signálů se spojitým časem).



Obr. 3.1: Diracův impuls.

3.2 Dekonvoluce využívající Diracova impulsu

Teoreticky by bylo získání impulsní charakteristiky systému velmi jednoduché, stačilo by přivést na vstup signál Diracova impulsu – vztah (3.7). V praxi se ale setkáváme s několika obtížemi: je třeba generovat impulsní zvuk dostatečné hlasitosti (cca 60 dB nad úroveň okolního hluku) a velmi malé délky trvání – např. při práci se vzorkovacím kmitočtem 96 kHz by měl impuls trvat $1/96000$ sekundy [10]. Bylo by možno uvažovat použití např. výstřelu z pistole, ale i tento zvuk nemá dobu trvání rovnu jednomu vzorku, ale několika desítek vzorků.

Tento problém lze řešit použitím techniky časového zrcadla (Time Reversal Mirror) [10], kterou se můžeme k Diracovu impulsu dosti přiblížit. Princip této techniky spočívá v konvoluci signálu $x(n)$ se sebou samým ale obráceným v čase (tedy tak, že poslední vzorek se stane prvním), který je označen jako $x^{-1}(n)$ (vztah (3.9)). Omezujícím faktorem metody TRM je vysoká obtížnost nalezení $x^{-1}(n)$ s dostatečnou přesností.

Bylo by také možné získat syntézou digitální impuls a reprodukovat jej elektroakustickým měničem. Ovšem reprodukce takto krátkého a hlasitého impulsu není možná bez kmitočtového i fázového zkreslení.

Mějme vstupní a výstupní signál číslicového LTI systému $x(n)$, resp. $y(n)$ a hledanou impulsní charakteristiku tohoto systému $h(n)$. Máme-li vstupní signál $x(n)$ takový, že platí

$$x(n) * x^{-1}(n) = \delta(n), \tag{3.9}$$

můžeme odvodit

$$y(n) * x^{-1}(n) = h(n) * x(n) * x^{-1}(n) = h(n) * \delta(n) = h(n). \tag{3.10}$$

To znamená, že na základě znalosti $x^{-1}(n)$ a výstupního signálu $y(n)$ je možno získat impulsní charakteristiku $h(n)$.

3.3 FFT algoritmus využívající bílý a růžový šum

Abychom se vyhnuli výše uvedeným problémům, je vhodné užitím Fourierovy transformace přejít z časové oblasti do oblasti kmitočtové. Konvoluce dvou signálů v časové oblasti tedy přejde v součin jejich spektrálních funkcí v oblasti kmitočtové ve tvaru

$$y(n) = x(n) * h(n) \Leftrightarrow Y(f) = X(f)H(f). \tag{3.11}$$

Každá harmonická složka je násobena jedním z k koeficientů, což představuje právě k matematických operací (v časové oblasti by jich bylo potřeba k^2), což spolu s použitím algoritmu FFT značně snižuje výpočetní náročnost. Nalezení funkce $H(f)$, tedy k koeficientů, je snadné. Tato funkce, označovaná také jako kmitočtová charakteristika LTI systému, totiž vyjadřuje vztah mezi spektry vstupního signálu $X(f)$ a výstupního signálu $Y(f)$. Jakmile tedy zjistíme koeficienty kmitočtové charakteristiky, zpětnou Fourierovou transformací získáme hledanou impulsní charakteristiku $h(n)$. Naneštěstí zde můžeme narazit na problém se stabilitou systému – je-li pro nějaký kmitočet hodnota koeficientu nulová, příslušný koeficient funkce $H(f)$ diverguje [10].

Tento problém je možno obejít využitím vlastností růžového a bílého šumu. Bílý šum je náhodný proces (signál) s konstantní výkonovou spektrální hustotou. Růžový šum je pak

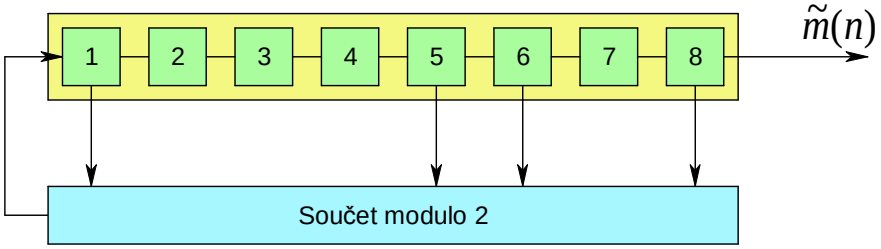
takový náhodný proces, jehož výkonová spektrální hustota je přímo úměrná převrácené hodnotě frekvence. Tedy např. při zdvojnásobení hodnoty frekvence klesne energie o 3 dB [19]. „Bohatost“ frekvenčního spektra těchto signálů je činí užitečnými pro mnoho aplikací.

Hlavním problémem je, že hodnoty vzorků jsou generovány náhodně; proto je spektrum při zobrazení pomocí krátké okénkové funkce spíše nespojité a fáze není známa vůbec. Zvětšování okénka Fourierovy transformace přináší, kromě zvýšení doby výpočtu, problémy s rozlišením v časové oblasti [10].

Tyto problémy dělají tuto techniku nepoužitelnou pro případy, kdy je nezbytné dostatečné rozlišení fáze a kmitočtu.

3.4 Autokorelační algoritmus využívající MLS

Kromě signálů růžového a bílého šumu můžeme při měření impulsní charakteristiky akustického prostoru využít pseudonáhodný signál maximální délky (MLS). MLS je binární signál $\tilde{m}(n)$ generovaný jako posloupnost 0 a 1 pomocí posuvného registru se zpětnou vazbou, která obsahuje součet modulo 2 (XOR) – viz Obr. 3.2. Takto generovaný signál je periodický s periodou $L = 2^N - 1$, kde N je počet buněk posuvného registru [9]. Vhodným nastavením vazeb do bloku součtu modulo 2 můžeme získat různé druhy MLS signálů [10]. Pro potřeby měření IR akustických prostorů se binární signál $\tilde{m}(n)$ obvykle převádí na symetrický dvouúrovňový signál $m(n)$ tak, že logické 1 odpovídá úroveň amplitudy signálu $-A$ a logické 0 úroveň $+A$.



Obr. 3.2: Generátor MLS.

Algoritmus pracuje s předpokladem, že LTI systém můžeme (podobně jako impulsní charakteristikou $h(n)$) popsat pomocí periodické impulsní charakteristiky $h_\mathrm{p}(n)$. Ta je definována jako odezva systému na periodický Diracův impuls

$$\delta_\mathrm{p}(n) = \begin{cases} 1 \text{ pro } n \bmod L = 0, \\ 0 \text{ jinde.} \end{cases} \tag{3.12}$$

Mezi impulsními charakteristikami soustavy $h(n)$ a $h_\mathrm{p}(n)$ pak platí

$$h_\mathrm{p}(n) = \delta_\mathrm{p}(n) * h(n) = \ldots = \sum_{k=-\infty}^{\infty} h(n + kL) \tag{3.13}$$

a je tedy zřejmé, že pokud je délka sekvence L menší než délka impulsní odezvy $h(n)$, dojde k její deformaci. Tento jev se nazývá časový aliasing. [9]

Přivedeme-li na vstup soustavy periodický signál $x_p(n)$, její periodická odezva $y_p(n)$ (perioda všech tří signálů je L) bude mít tvar

$$y_p(n) = x_p(n) \otimes h_p(n) = \sum_{k=0}^{L-1} x_p(k) h_p(n-k), \tag{3.14}$$

kde operátor $\otimes$ značí cyklickou konvoluci.

Důležitou vlastností MLS signálů je, že pomocí jejich autokorelační funkce můžeme získat Diracův impuls. Autokorelační funkci MLS signálu dostaneme jako

$$R_{pmm}(n) = m(n) \oplus m(n) = \delta_p(n) - \frac{1}{L+1}, \tag{3.15}$$

kde operátor $\oplus$ značí cyklickou korelaci.

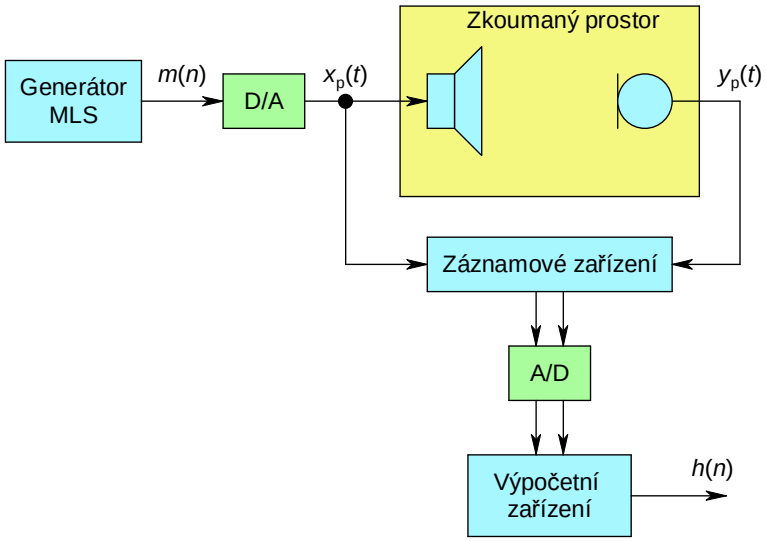
Přivedeme-li tedy na vstup soustavy MLS signál $m(n)$, lze její periodickou impulsní charakteristiku $h_p(n)$ určit pomocí vzájemné korelace vstupního signálu se signálem výstupním jako

$$R_{pmy}(n) = m(n) \oplus y_p(n) = m(n) \oplus \big(m(n) \otimes h_p(n)\big). \tag{3.16}$$

U lineárních operací cyklické korelace a cyklické konvoluce ve vztahu (3.16) můžeme zaměnit jejich pořadí a po dosazení vztahu (3.15) dostaneme

$$R_{pmy}(n) = \big[m(n) \oplus m(n)\big] \otimes h_p(n) = \left(\delta_p(n) - \frac{1}{L+1}\right) \otimes h_p(n) \approx h_p(n). \tag{3.17}$$

Nevýhodou metody využívající signálu MLS je vysoká citlivost na linearitu zkoumaného systému. Pokud není systém dokonale lineární a časově invariantní, naměřená IR bude zkreslena. Velké nároky jsou kladeny již na samotný měřící systém – ten totiž využívá elektroakustického měniče (reprodukce měřícího signálu), který sám o sobě zanáší do systému značné zkreslení [1], [10].



Obr. 3.3: Blokové schéma měření impulsní charakteristiky pomocí signálu MLS.

3.5 Konvoluční algoritmus využívající SWEEP signál

Tato metoda navazuje na metodu uvedenou v kap. 3.2. K měření impulsní charakteristiky však využívá harmonického signálu, který mění svoji frekvenci v čase. Jedná se tedy o čistý tón, jehož frekvence roste lineárně, popř. logaritmicky. Průběh výkonové spektrální hustoty lineárního SWEEP signálu je analogický s průběhem této veličiny u bílého šumu; logaritmický SWEEP signál pak vykazuje vlastnosti šumu růžového, proto se při měření IR často upřednostňuje [1]. V tom případě totiž připadne více energie na kritickou zónu nižších kmitočtů a přechod ke kmitočtům vyšším je rychlejší [10].

Výhodou SWEEP signálů je snadné nalezení jejich inverzní podoby v čase, tedy přechodu $x(n) \Rightarrow x^{-1}(n)$. V případě lineárního SWEEP signálu se jedná o pouhé obrácení signálu v čase, u logaritmického SWEEP signálu je navíc potřeba použít zesílení o 6 dB na oktávu.

Navíc, protože je v jednom okamžiku produkován signál pouze na jednom kmitočtu, případné zkreslení se bude skládat pouze z vyšších harmonických tohoto kmitočtu. Použijeme-li tedy signál s rostoucím kmitočtem, vyšší harmonické budou generovány na kmitočtech ležících nad kmitočtem měřícího signálu. Po provedení dekonvoluce se tedy jakékoli zkreslení projeví v části impulsní charakteristiky se zápornými hodnotami času, takže může být z velké části potlačeno [10].

Máme-li tedy signál $x^{-1}(n)$ a naměřenou odezvu systému $y(n)$, můžeme získat impulsní odezvu $h(n)$ dle vztahu (3.10).

Z uvedeného je patrné, že každá z technik má své klady i zápory. Z tohoto důvodu je každý z měřících signálů (Diracův impuls, bílý a růžový šum, signály MLS a SWEEP) i použitých algoritmů (dekonvoluce, rychlá Fourierova transformace, autokorelace a konvoluce) vhodný k různým účelům a pro zkoumání rozdílných systémů.

Například k získání akustických parametrů divadelního sálu je nejvhodnější generování krátkého impulsu pomocí střelné zbraně, za účelem digitální simulace takového akustického prostoru ale bude nejvhodnější užití konvolučního algoritmu a SWEEP signálu.

Základem je tedy zvolení správného postupu měření impulsní charakteristiky daného akustického prostoru, který má být simulován. Je třeba vzít v potaz všechny aspekty – tedy od výpočetní náročnosti použitého algoritmu až po závislost na linearitě systému i samotného měřícího zařízení.

4 METODY VÝPOČTU KONVOLUCE

4.1 Zpracování v časové rovině

Jak již bylo zmíněno dříve, odezvu lineárního časově invariantního systému na vstupní signál lze určit pomocí konvoluce vstupního signálu s impulsní charakteristikou tohoto systému. Výstupní signál systému se spojitým časem je dán vztahem

$$y(t) = x(t) * h(t) = \int_0^{\infty} h(\tau)x(t-\tau)\mathrm{d}\tau\,, \tag{4.1}$$

kde $x(t)$ je vstupní signál systému, $h(t)$ jeho impulsní charakteristika a * značí operaci konvoluce.

V případě diskrétního LTI systému je jeho odezva dána lineární diskrétní konvolucí vstupního signálu $x(n)$ konečné délky N_1 a jeho impulsní charakteristiky $h(n)$ délky N_2 jako

$$y(n) = x(n) * h(n) = \sum_{k=0}^{N-1} h(k)x(n-k)\,, \tag{4.2}$$

kde $N = N_1+N_2-1$ je délka výstupního signálu.

Z uvedeného vztahu je vidět, že přímý výpočet lineární diskrétní konvoluce v časové rovině obnáší veliké množství matematických operací součtu a součinu. Z tohoto důvodu je tento postup pro práci v reálném čase vhodný pouze pro délku impulsní charakteristiky systému N v řádu několika set vzorků [5]. Výpočetní náročnost (počet operací součet-součin vztažený na jeden výstupní vzorek) roste lineárně s délkou impulsní charakteristiky [7].

Na druhou stranu ovšem není pro implementaci tohoto algoritmu v časové rovině zapotřebí žádné vyrovnávací paměti. Vstupní vzorek je načten a zpracován, po té je odeslán výsledek. Algoritmus tedy nezanáší další zpoždění spojené s naplněním vyrovnávací paměti před samotným započetím výpočtu; celkové zpoždění je dáno dobou zpracování vzorku.

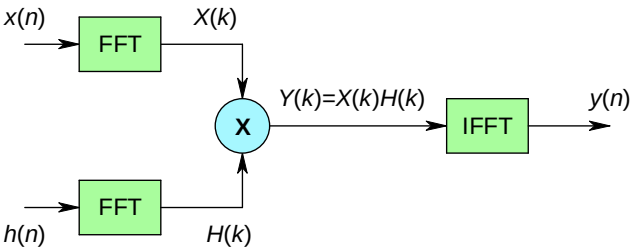
4.2 Zpracování v kmitočtové rovině

Vlastností Fourierovy transformace je, že konvoluci dvou signálů v časové rovině převádí na součin jejich spekter v rovině kmitočtové. Při následujícím popisu různých algoritmů předpokládáme, že délka impulsní charakteristiky je mnohem větší než délka vstupního signálu. V tom případě existuje několik způsobů výpočtu, které zohledňují výpočetní náročnost nebo zpoždění, které souvisí s nutným předcházejícím naplněním vyrovnávací paměti před samotným výpočtem.

4.2.1 Algoritmus bez dělení impulsní charakteristiky na bloky

Mějme vstupní signál $x(n)$ a impulsní charakteristiku LTI systému $h(n)$ konečných délek tak, jak je uvedeno v kap. 4.1, přičemž předpokládáme $N_2 >> N_1$. Budou-li mít obrazy diskrétní Fourierovy transformace (DFT) těchto signálů alespoň délku N, platí:

$$Y(k) = X(k)H(k)\,. \tag{4.3}$$



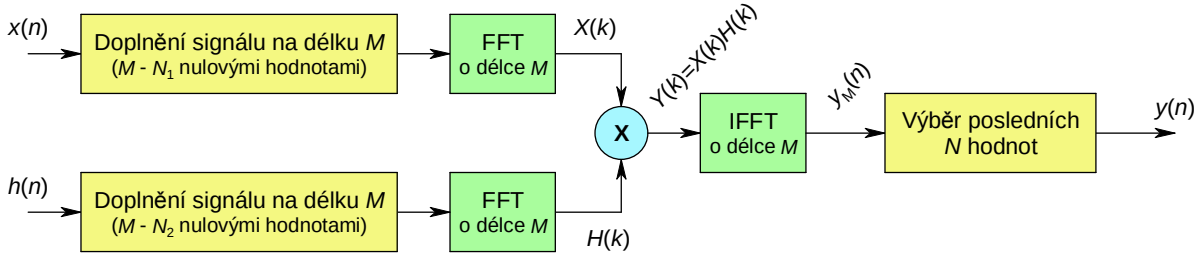
Obr. 4.1: Zjednodušené schéma výpočtu rychlé konvoluce pomocí FFT.

Je-li k výpočtu DFT použito algoritmu rychlé Fourierovy transformace (FFT), označuje se tento postup jako rychlá konvoluce – Obr. 4.1. [16]

Problémem ovšem je, že algoritmus FFT předpokládá, že analyzovaný blok signálu je periodický. Proto by přímá implementace podle Obr. 4.1 bez dalších upřesnění vedla ke zkreslení výstupního signálu. Periodicita způsobená algoritmem FFT musí být z výstupního signálu odstraněna [5]. K tomu slouží dvě metody: metoda přičtení přesahu nebo metoda odstranění přesahu, které budou podrobněji popsány dále.

Pro objasnění bude stručně naznačen postup s použitím **metody odstranění přesahu** (Obr. 4.2). Jak víme, délka výsledku konvoluce dvou signálů délek N_1 a N_2 je $N = N_1+N_2-1$. Abychom předešli zkreslení výstupního signálu (časovému aliasingu), musí být transformace vstupních signálů provedeny FFT o délce M, přičemž musí platit $M > N$. Dále, aby byla DFT provedena opravdu pomocí algoritmu FFT, musí platit $M = 2^a$, kde a je celé kladné číslo; tedy že délka FFT je rovna nejbližší vyšší mocnině 2 vzhledem k N. Vstupní signály jsou tedy doplněny nulovými hodnotami (vloženými před) na délku M, po té je provedena jejich FFT. Součin obrazů obou signálů v kmitočtové rovině $Y(k)$ je poté převeden zpět do roviny časové pomocí zpětné rychlé Fourierovy transformace (IFFT) délky taktéž M. Z výstupního signálu je vybráno pouze posledních N hodnot, které nejsou zkresleny časovým aliasingem a odpovídají výsledku lineární diskrétní konvoluce vstupních signálů (vztah (4.3)).

Výpočetní náročnost algoritmu pracujícího výše zmíněným způsobem v kmitočtové rovině roste pouze logaritmicky s délkou impulsní charakteristiky. Ovšem tato nízká výpočetní náročnost je vykoupena značným zpožděním, které je dáno naplněním vstupních vyrovnávacích pamětí. Zpoždění mezi vstupem a výstupem (latence) je tedy rovno přinejmenším délce impulsní charakteristiky. [7]



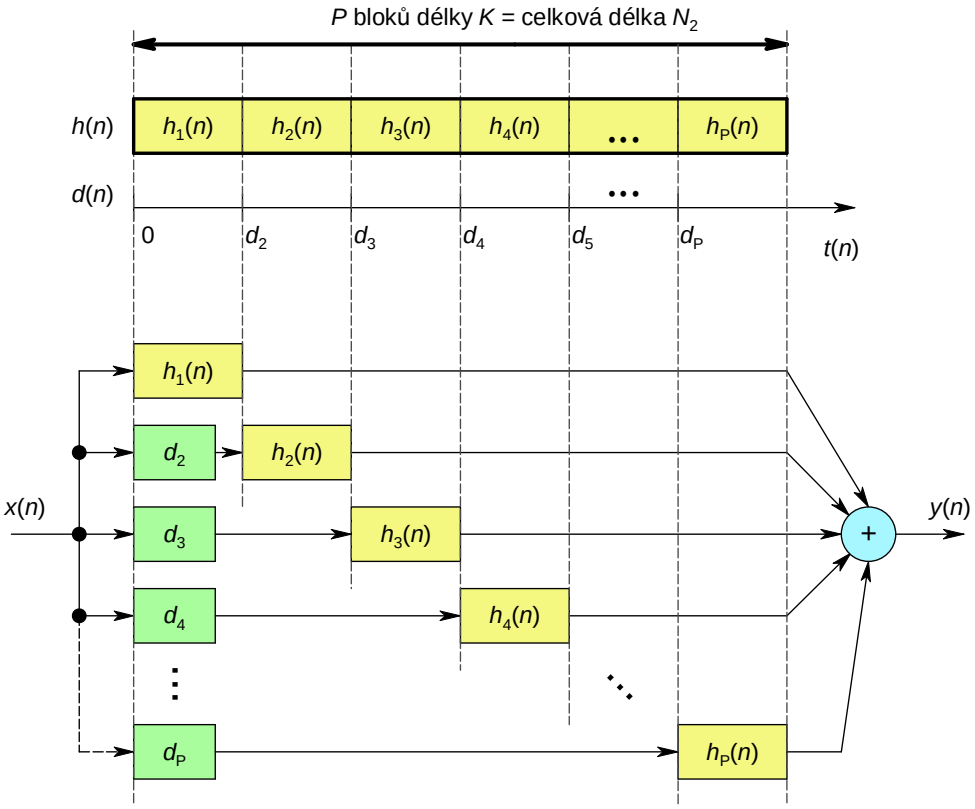
Obr. 4.2: Výpočet rychlé konvoluce využívající metody odstranění přesahu.

4.2.2 Algoritmus s dělením impulsní charakteristiky na bloky stejné délky

Jednoduchým způsobem, jak dosáhnout nízké latence a přitom zachovat nízkou výpočetní náročnost je rozdělit, zpravidla velmi dlouhou impulsní charakteristiku $h(n)$, na více kratších bloků. Každý z těchto bloků je pak zpracován samostatně způsobem popsaným v kap. 4.2.1.

Uvažujme tedy impulsní charakteristiku konečné délky N_2, která je rozdělena do P bloků stejné délky K. Délka bloku K je s ohledem na implementaci algoritmu a použití FFT volena jako mocnina 2, tedy $K = 2^b$, kde b je kladné celé číslo. V případě, že délka impulsní charakteristiky N_2 není celočíselným násobkem délky bloku K, může být na potřebnou délku doplněna nulovými hodnotami.

Tímto způsobem tedy můžeme převést výchozí filtr na odpovídající paralelní kombinaci filtrů kratších, jak je naznačeno na Obr. 4.3. Každá paralelní větev se skládá z jednoho bloku impulsní odezvy a zpoždění d_i, které odpovídá pozici bloku v původní IR. Po samostatném zpracování signálu v každé z větví jsou jejich výstupy spojeny zpravidla využitím metody přičtení přesahu. [7]



Obr. 4.3: Dělení impulsní charakteristiky na bloky stejné délky.

Metoda přičtení přesahu (Overlap-add Method)

Jde o ekvivalentní způsob k metodě odstranění přesahu, jak lze spočítat konvoluci dvou signálů, z nichž jeden je výrazně delší než druhý, v kmitočtové oblasti [16]. Princip bude naznačen na výpočtu konvoluce bloku vstupního signálu délky L s mnohem delší impulsní charakteristikou LTI systému.

Předpokládejme, že impulsní charakteristika bude rozdělena na bloky o délce K, která odpovídá délce bloku vstupního signálu L. Ta je dána velikostí vyrovnávací paměti pro vstupní signál (čímž je určena latence), která má vzhledem ke své implementaci vždy velikost rovnu nějaké mocnině 2. Jak víme, abychom zamezili vzniku zkreslení (časového aliasingu) ve výstupním signále při výpočtu rychlé konvoluce, musíme pro transformaci signálů do kmitočtové roviny použít FFT o délce M větší než $N = L + K - 1$. Vzhledem k tomu, že pro M musí platit $M = 2^a$, kde a je kladné celé číslo; je volena hodnota $M = 2K$.

Každý blok impulsní charakteristiky délky K tedy musí být nejdříve doplněn na délku M nulovými hodnotami, které jsou oproti metodě odstranění přesahu vloženy za blok vzorků impulsní charakteristiky. Stejným způsobem je doplněn i blok vstupního signálu. Výhodou dělení impulsní charakteristiky na bloky se stejnou délkou je, že blok vstupního signálu je transformován pouze jednou – ve všech větvích je totiž prováděna rychlá konvoluce se stejnou délkou bloků [7]. To samé platí pro všechny bloky impulsní charakteristiky – jejich transformaci stačí provést pouze jednou.

Je tedy provedena FFT o délce M bloku vstupního signálu a všech bloků impulsní charakteristiky. Obraz výstupní posloupnosti m-tého bloku získáme jako

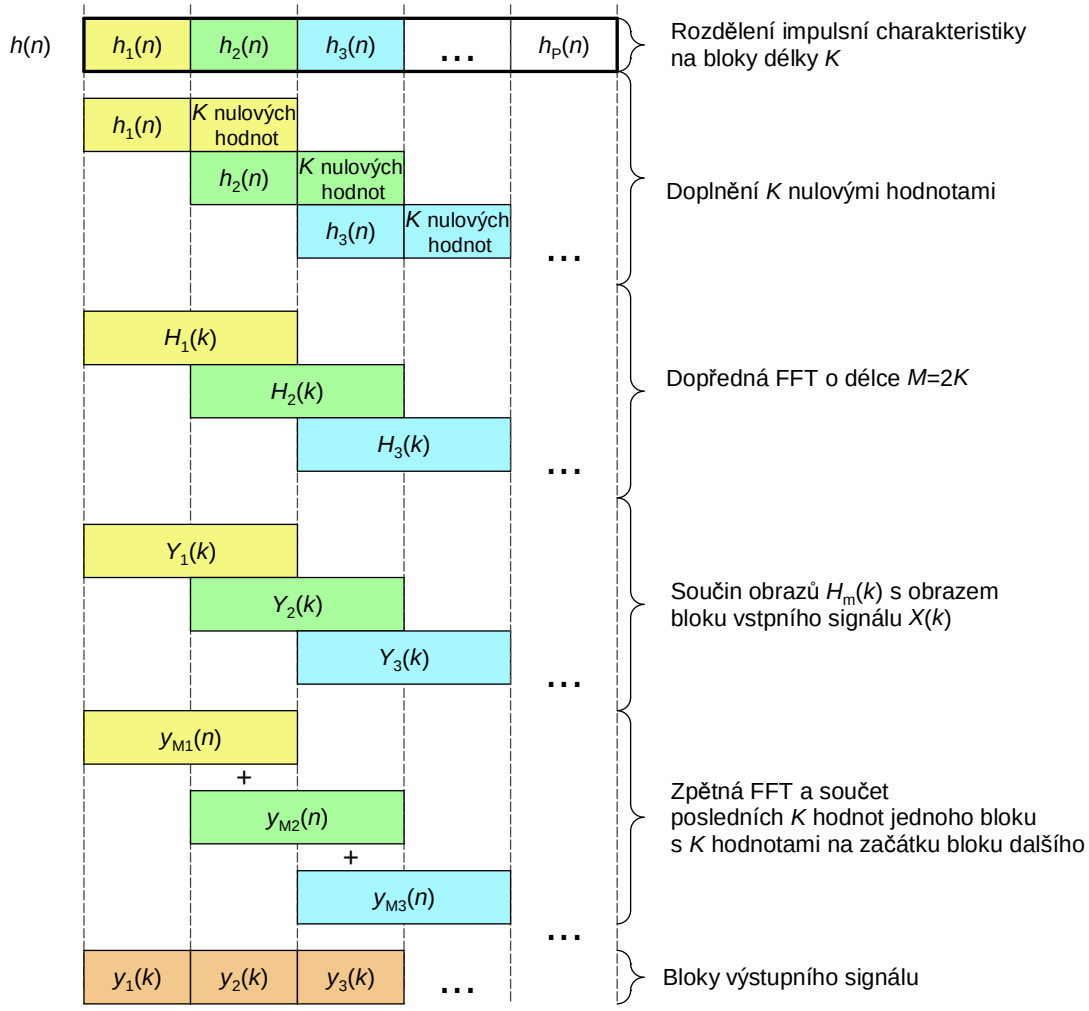
$$Y_m(k) = X(k)H_m(k), \tag{4.4}$$

kde $X(k)$ je obraz jednoho bloku vstupního signálu a $H_\mathrm{m}(k)$ je obrazem m-tého bloku impulsní charakteristiky.

Po provedení IFFT obrazu $Y_m(k)$ získáme výstupní posloupnost $y_{\mathrm{M}m}(n)$, která má však vlivem prodloužení každé ze vstupních posloupností dvojnásobnou délku (tím bylo zabráněno časovému aliasingu). Princip metody přičtení přesahu spočívá v tom, že se sečte K hodnot na konci jednoho bloku s K hodnotami na začátku následujícího bloku dat. Tak jsou určeny bloky výstupního signálu $y_m(n)$. Celý proces je naznačen na Obr. 4.4.

Metoda přičtení přesahu nabízí několik možností pro optimalizaci výpočetní náročnosti. Například, jak je vidět z Obr. 4.4, přičtení přesahu může být uplatněno již v kmitočtové rovině; tedy že obrazy součinů $Y_m(k)$ jsou sečteny na tzv. „ose zpoždění v kmitočtové rovině“ (FDL – Frequency-domain Delay Line). Tak by bylo možné použít pouze jednu zpětnou FFT pro výstupní signál příslušející každému bloku signálu vstupního. Tento algoritmus má konvexní průběh závislosti výpočetní náročnosti na délce bloku; optimální délka bloku tedy může být zjištěna pomocí derivace. Problémem ovšem je, že pro filtry s dlouhou impulsní charakteristikou je optimální délka bloku obvykle příliš velká vzhledem k požadované latenci a oproti tomu při snižování délky bloku značně roste výpočetní náročnost. [7]

Metody využívající FDL budou popsány v kapitolách 4.2.4 a 4.2.5.



Obr. 4.4: Znázornění metody přičtení přesahu.

4.2.3 Algoritmus s dělením impulsní charakteristiky na bloky různých délek

Výše zmíněný problém s velikostí bloku, na které má být impulsní charakteristika dělena ve spojitosti s latencí a výpočetní náročností algoritmu lze vyřešit dělením IR na bloky různých délek. Dělení probíhá do bloků s rostoucí délkou; vlivem kratších bloků na začátku IR je dosaženo nízké latence, zatímco následující bloky velké délky snižují výpočetní náročnost konvoluce. [7]

Možnost volby délky bloků činí tuto metodu efektivnější v porovnání s dělením na bloky stejných délek. Literatura [7] uvádí dvě schémata rozdělení délek bloků:

- **Algoritmus minimální výpočetní náročnosti**

  Toto schéma dělí impulsní charakteristiku do I+1 bloků o velikostech

  $K, K, 2K, 4K, 8K, ..., 2^{(I-1)}K,$

  kde K je celočíselná mocnina 2.

- **Algoritmus rovnoměrného rozložení výpočetní náročnosti**

Toto schéma slouží k rovnoměrnému zatížení procesoru a dělí IR do 2*I* bloků o velikostech

$$K, K, 2K, 2K, 4K, 4K, ..., 2^{(I-1)}K, 2^{(I-1)}K.$$

V případě, že je vyžadováno nulové zpoždění mezi vstupním a výstupním signálem, první blok délky *K* je zpracován přímo v časové oblasti – viz kapitola 4.1.

4.2.4 Algoritmus s dělením impulsní charakteristiky na bloky dvou různých délek s využitím dvojité FDL

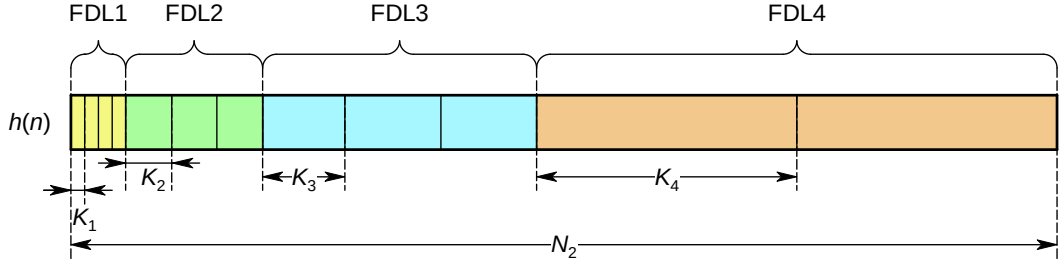
Využití FDL při výpočtu konvoluce v kmitočtové oblasti pomocí algoritmu s dělením impulsní charakteristiky na bloky stejné délky a využívající metody přičtení přesahu bylo naznačeno již v závěru kapitoly 4.2.2. Principem bylo, že přičtení přesahu bylo provedeno již v kmitočtové oblasti a tím se značně snížil počet nutných zpětných FFT. Stále zde však byl problém s optimální délkou bloku dělené impulsní charakteristiky pro přípustnou výpočetní náročnost, která byla v rozporu s požadovanou nízkou latencí.

Jedním z řešení je rozdělit filtr pomocí dvojice FDL, tedy použít dvojí velikost bloků pro dělení IR. První filtr pak používá pro začátek impulsní charakteristiky dělení na bloky malé délky (K_1), která je daná požadovanou latencí. Druhý filtr pak již za účelem snížení výpočetní náročnosti používá bloků větší délky (K_2). Způsobem, jak dosáhnout co největší efektivity tohoto algoritmu, je nalézt optimální velikost a počet bloků druhého filtru a počet bloků filtru prvního. Výpočetní náročnost algoritmu s dvojitou FDL je tedy při dosažení menší latence podstatně nižší než u výše uvedených algoritmů.

4.2.5 Algoritmus s dělením impulsní charakteristiky na bloky různých délek s využitím mnohonásobné FDL

Algoritmus popsaný v kapitole 4.2.4 nabízí další možnost optimalizace v podobě použití více FDL. V tom případě je kmitočtová charakteristika filtru rozdělena na tolik segmentů, kolik FDL algoritmus používá. Každý segment je pak složen z různého počtu bloků stejné délky, přičemž velikost bloku se s každým segmentem postupně zvyšuje – viz Obr. 4.5.

Problém tohoto algoritmu ovšem spočívá v tom, jak nalézt optimální schéma dělení ve smyslu: počet segmentů (tedy FDL); velikost a počet bloků v jednotlivých segmentech. Jediná známá velikost bloku je K_1, která je určena hodnotou požadované latence. Například, pokud bychom simulovali dozvuk o délce 3 s (což při vzorkovacím kmitočtu f_{vz} = 44,1 kHz představuje filtr s impulsní charakteristikou o délce cca 150000 vzorků) a požadovaná doba latence by byla 5,8 ms (odpovídá K_1 = 256), celkový počet možných způsobů rozdělení kmitočtové charakteristiky překračuje 70 milionů. [7]



Obr. 4.5: Příklad rozdělení impulsní charakteristiky délky N_2 na čtyři segmenty s rozdílnými délkami bloků K_1, K_2, K_3 a K_4.

4.3 Srovnání výpočetních náročností algoritmů pracujících v kmitočtové rovině

V této kapitole budou srovnány výpočetní náročnosti uvedených algoritmů pracujících v kmitočtové rovině dle [7]. Výpočetní náročnost (cena výpočtu) zde bude měřena jako počet dvojic operací součin-součet („madds“ z anglického multiply-adds) na jeden výstupní vzorek. Budeme předpokládat, že diskrétní Fourierova transformace bude počítán výhradně pomocí FFT. Obraz impulsní charakteristiky v kmitočtové rovině může být získán předem, proto není do následujících výpočtů zahrnut.

Celkový počet operací nutných pro výpočet rychlé Fourierovy transformace posloupnosti reálných čísel délky N je

$$C_{\mathrm{FFT}}(N) = kN\log_2(N), \tag{4.5}$$

kde k je konstanta závislá na konkrétní implementaci výpočtu FFT (typicky $k = 1{,}5$). Vztaženo na jeden výstupní vzorek tedy platí

$$O_{\mathrm{FFT}}(N) = \frac{kN\log_2(N)}{N} = k\log_2(N). \tag{4.6}$$

Algoritmus bez dělení impulsní charakteristiky na bloky

Při délce impulsní charakteristiky označené jako N (namísto dříve použitého N_2 pro zpřehlednění následujících rovnic) musejí obě metody využívané tímto algoritmem, tedy metoda přičtení přesahu a metoda odstranění přesahu, k zamezení vzniku časového aliasingu ve výstupním signále provádět FFT o délce $2N$. Vždy je tedy doplněno N hodnot. Cena výpočtu přímé FFT na jeden výstupní vzorek pak je

$$O_{\mathrm{FFT}}(N) = \frac{k(2N)\log_2(2N)}{N} = 2k\log_2(2N). \tag{4.7}$$

Takto vzniklý blok komplexních hodnot je poté násoben s obrazem impulsní charakteristiky, což představuje N součinů komplexních čísel – to odpovídá $4N$ operacím součin-součet reálných čísel a platí tedy

$$O_{\mathrm{SM}}(N) = \frac{4N}{N} = 4. \tag{4.8}$$

Zpětná FFT má stejnou cenu jako dopředná – vztah (4.7). Je-li použita metoda přičtení přesahu, je navíc potřeba N operací součtu (přičtení druhé poloviny předchozího bloku s první polovinou bloku aktuálního – viz Obr. 4.4), tedy navíc jeden součet na výstupní vzorek. Metoda odstranění přesahu žádné další operace nepřidává a je-li použita, celková výpočetní náročnost algoritmu bez dělení IR na bloky je

$$O_{\mathrm{SB}}(N) = O_{\mathrm{FFT}}(N) + O_{\mathrm{SM}}(N) + O_{\mathrm{IFFT}}(N) = 4k\log_2(2N) + 4. \tag{4.9}$$

Algoritmus s dělením impulsní charakteristiky na bloky stejné délky

V tomto případě je impulsní charakteristika délky N rozdělena na P bloků délky K – viz Obr. 4.3 (impulsní charakteristika může být za tímto účelem prodloužena o nezbytný počet nulových vzorků). K získání výstupního bloku je zapotřebí pouze jediné FFT o délce $2K$, protože všechny bloky IR mají stejnou délku. Po té je provedeno násobení spekter a

zpětná transformace každého z P bloků součinů zpět do časové roviny, kde je uplatněna metoda přičtení přesahu. Celý tento proces je znázorněn na Obr. 4.4.

Celková výpočetní náročnost algoritmu s dělením IR na bloky stejné délky tedy je

$$\begin{aligned} O_{\mathrm{MBUP}}(K) &= O_{\mathrm{FFT}}(K) + \frac{N}{K}\left[O_{\mathrm{SM}}(K) + O_{\mathrm{IFFT}}(K) + O_{\mathrm{OlapAdd}}(K)\right] \\ &= 2k\log_2(2K) + \frac{N}{K}\left[4 + 2k\log(2K) + 1\right] \\ &= \left(\frac{N}{K} + 1\right)2k\log(2K) + 5\frac{N}{K}. \end{aligned} \tag{4.10}$$

Tato závislost výpočetní náročnosti na zvolené délce bloku K je naznačena na Obr. 4.6. Jak je vidět, pro danou délku filtru N se výpočetní náročnost mění souběžně se zvolenou délkou bloku. Zmenšování hodnoty K, tedy snaha o dosažení menší latence, je vykoupeno zvýšením ceny výpočtu; minimální výpočetní náročnosti by bylo dosaženo bez jakéhokoli dělení IR na bloky (vykoupeno velkou latencí). [7]

Algoritmus s dělením impulsní charakteristiky na bloky různých délek

Princip tohoto algoritmu byl popsán v kapitole 4.2.3 a byly uvedeny jeho dvě verze. Na verzi algoritmu s **minimální výpočetní náročností** lze nahlížet jako na superpozici algoritmu bez dělení impulsní charakteristiky na bloky (kap. 4.2.1) pro sérii bloků K, $2K$, $4K$, $8K$, ..., $2^{(I-1)}K$ a operací nutných pro zpracování prvního bloku délky K, tedy spektrální násobení a IFFT.

Celková výpočetní náročnost algoritmu s dělením IR na bloky různých délek s minimální výpočetní náročností pak je

$$O_{\mathrm{MBMC}}(K) = O_{\mathrm{SM}}(K) + O_{\mathrm{IFFT}}(K) + \sum_{i=0}^{I-1} O_{\mathrm{SB}}\left(2^i K\right), \tag{4.11}$$

kde $I = \log_2(N/K)$. Po vyjádření a dosazení vztahu (4.9) dostaneme

$$\begin{aligned} O_{\mathrm{MBMC}}(K) &= 4 + 2k\log_2(2K) \\ &\quad + \log_2\left(\frac{N}{K}\right)\left(4k\log_2 K + 2k + 4\right) \\ &\quad + 2k\left[\log_2\left(\frac{N}{K}\right)\right]^2. \end{aligned} \tag{4.12}$$

Bez odvozování uveďme vztah pro verzi algoritmu s **rovnoměrným rozložením výpočetní náročnosti**:

$$\begin{aligned} O_{\mathrm{MBUL}}(K) &= \left(6k\log_2 N + 3k + 8\right)\log_2\left(1 + \frac{N}{2K}\right) \\ &\quad + 3k\left[\log_2\left(1 + \frac{N}{2K}\right)\right]^2. \end{aligned} \tag{4.13}$$

Algoritmus s dělením impulsní charakteristiky na bloky stejné délky s využitím jednoduché FDL

Princip byl objasněn v závěru kapitoly 4.2.2. K získání výstupního bloku je nutno nejprve provést FFT o délce 2*K* bloku vstupního signálu. Po té je pro všech *P* obrazů bloků impulsní charakteristiky provedeno násobení spekter a uplatněna metoda přičtení přesahu již v kmitočtové oblasti. Tyto dvě operace lze realizovat za cenu čtveřice operací součin-součet pro každý z *P* obrazů bloků IR. Blok získaný z FDL je pak pomocí IFFT o délce 2*K* převeden do časové roviny. Celková výpočetní náročnost je pak dána vztahem

$$\begin{aligned} O_{\text{SingleFDL}}(K) &= O_{\text{FFT}}(K) + O_{\text{IFFT}}(K) + \left(\frac{N}{K}\right) O_{\text{SM\&OlapAdd}}(K) \\ &= 4k\log_2(2K) + 4\frac{N}{K}. \end{aligned} \tag{4.14}$$

Z průběhu této závislosti (Obr. 4.6) je vidět, že cena výpočtu se nemusí v intervalu $K \in < 2, N >$ nutně měnit rovnoměrně, ale že může nabývat menších hodnot pro hodnoty *K* blízkých *N*. To je zapříčiněno tím, že cena výpočtu FFT je nejvyšší pro velké hodnoty *K*, zatím co pro malé hodnoty *K* dominuje výpočetní náročnost násobení spektrálních funkcí.

Optimální velikost bloku pro tento algoritmus můžeme určit jako

$$K_{\text{opt}} = \frac{\ln(2)}{k} N = \frac{0{,}6931}{k} N \tag{4.15}$$

a zaokrouhlením této hodnoty na nejbližší celočíselnou mocninu 2 tak, že dosadíme do vztahu (4.14) nejbližší vyšší a nižší takovou hodnotu a vybereme tu, pro kterou je výsledná hodnota výpočetní náročnosti nižší.

Dále bylo zjištěno, že i když není dosažení nízké latence prvořadým cílem, je výhodné použít dělení impulsní charakteristiky za předpokladu, že použitá implementace FFT nepatří mezi nejefektivnější (tj. pro $k >> \ln(2)$). [7]

Algoritmus s dělením impulsní charakteristiky na bloky dvou různých délek s využitím dvojité FDL

I když algoritmus využívající jednoduché FDL vykazuje nižší hodnoty výpočetní náročnosti oproti algoritmu s dělením impulsní charakteristiky na bloky různých délek bez využití FDL, výpočetní náročnost značně roste s hodnotou *K* zmenšující se ze svého optima. Je-li požadována nízká hodnota latence, tedy malá délka bloků *K*, algoritmus pracuje daleko od svého optimálního stavu a jeho výpočetní náročnost je značná.

Je-li například délka impulsní charakteristiky $N = 65536$ vzorků a s ohledem na latenci požadujeme $K = 128$, cena výpočtu bude 2096 madds. Při použití optimální hodnoty $K_{\text{opt}} = 32768$ by však cena výpočtu byla pouze 104 madds.

Jak již bylo popsáno v kapitole 4.2.4, tento algoritmus dělí IR na bloky dvou různých délek. Délka bloků první FDL (K_1) je dána požadovanou latencí, délka bloků druhé FDL (K_2) je proměnná, která nám umožní minimalizovat výpočetní náročnost algoritmu pro zadanou délku impulsní charakteristiky *N* a délku bloku první FDL K_1. Pro jednoduchost předpokládejme, že velikosti K_1, K_2 jsou celočíselnými mocninami 2.

Optimální velikost K_2 můžeme nalézt pomocí vztahu

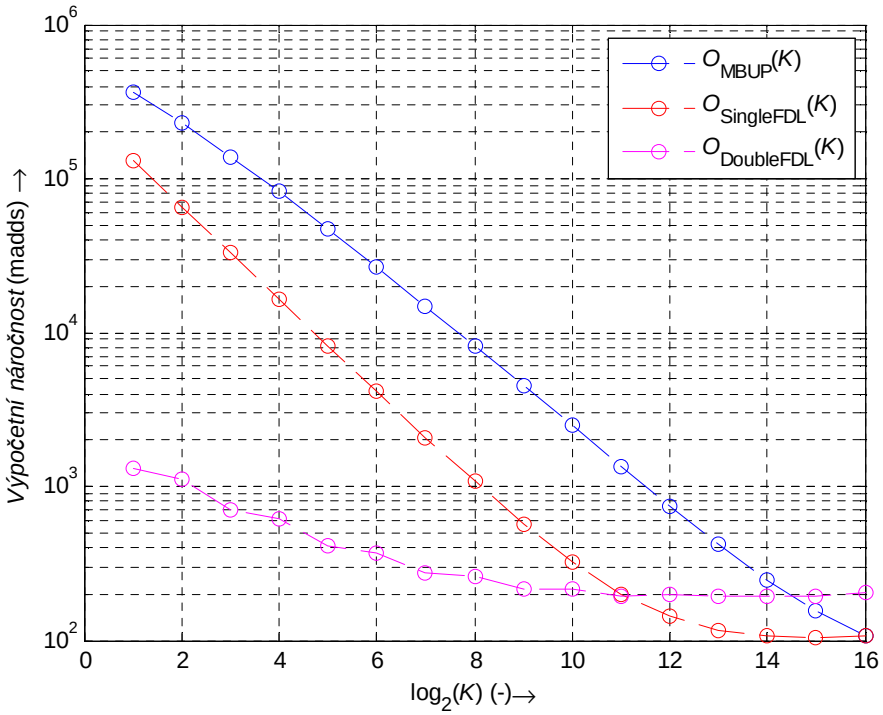
$$K_2 = \frac{-kK_1}{2\ln(2)} + \sqrt{\left(\frac{kK_1}{2\ln(2)}\right)^2 + NK_1} \tag{4.16}$$

a zaokrouhlení na nejbližší celočíselnou mocninu 2.

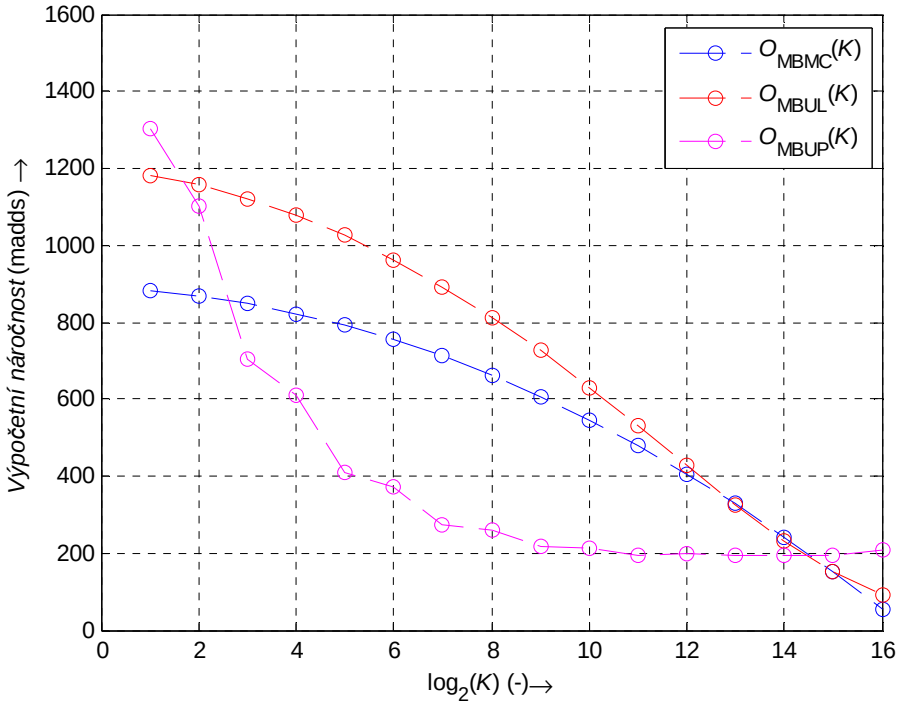
Výpočetní náročnost algoritmu využívajícího dělení IR na bloky dvou různých délek a využívající dvojité FDL pak má s přihlédnutím ke vztahu (4.14) tvar

$$\begin{aligned} O_{\text{DoubleFDL}}(K) &= 4k\log_2(2K_1) + 4\frac{K_2}{K_1} \\ &\quad + 4k\log_2(2K_2) + 4\left(\frac{N}{K_2} - 1\right), \end{aligned} \tag{4.17}$$

kde první řádek odpovídá první FDL a druhý řádek FDL druhé.



Obr. 4.6: Výpočetní náročnost algoritmů používajících dělení IR: na bloky stejné velikosti (MBUP); na bloky stejné velikosti využívající jednoduché FDL; na bloky dvou velikostí využívající dvojité FDL ($K_1 = K$).



Obr. 4.7: Výpočetní náročnost algoritmů používajících dělení impulsní charakteristiky na bloky různých délek – verze s minimální výpočetní náročností (MBMC) a s rovnoměrným rozložením výpočetní náročnosti (MBUL) v porovnání s algoritmem využívajícím dvojitou FDL.

5 IMPLEMENTACE VYBRANÝCH ALGORITMŮ V PROSTŘEDÍ MATLAB

V první fázi této práce byly v prostředí Matlab implementovány a otestovány tři vybrané algoritmy, které uplatňují metodu přičtení přesahu v časové rovině: metoda s dělením impulsní charakteristiky na bloky stejné délky (MBUP) a s dělením IR na bloky různých délek – verze s minimální výpočetní náročností (MBMC) a verze s rovnoměrným rozložením výpočetní náročnosti (MBUL). Po té se práce věnuje dvěma metodám pracujícím s FDL – jednoduché a dvojité FDL. Právě tyto dvě metody byly vybrány pro implementaci do podoby VST plug-in modulu (programovací jazyk C++). Jejich modifikace pro stereofonní zpracování zvukových signálů a optimalizaci pro efektivní a bezproblémovou funkci zpracování signálů v reálném čase bude popsána v kapitole 7.

5.1 Algoritmy uplatňující metodu přičtení přesahu v časové rovině

Všechny algoritmy jsou navrženy pro zpracování monofonního zvukového signálu. Vstupními parametry jsou tedy vektor vstupního signálu *x*, vektor *h* reprezentující impulsní charakteristiku systému a parametr *K*. Hodnota *K* udává u algoritmu MBUP velikost všech bloků, u algoritmů MBMC a MBUL pak velikost prvního bloku impulsní charakteristiky. Na stejně dlouhé bloky je dělen také vektor *x*. Vzhledem k požadované minimální latenci je první blok IR zpracován přímo v časové oblasti.

Vývojový diagram **algoritmu s dělením impulsní charakteristiky na bloky stejné délky MBUP** (viz kap. 4.2.2) je uveden v Příloha C.1. Využívá předpřipravení obrazů bloků impulsní charakteristiky v kmitočtové oblasti před výpočtem hlavní smyčky algoritmu pro jednotlivé bloky vstupního signálu.

Algoritmus s dělením impulsní charakteristiky na bloky různých délek – verze s minimální výpočetní náročností MBMC vznikl modifikací algoritmu MBUP. Předpřipravení obrazů jednotlivých bloků IR v kmitočtové oblasti a samotnému výpočtu zde musí předcházet nalezení způsobu rozdělení impulsní charakteristiky dle pravidel uvedených v kapitole 4.2.3 na základě zjištěné délky IR a zadaného parametru *K*. Velikosti jednotlivých bloků jsou uloženy do vektoru *lst*, ze kterého jsou dále podle potřeby zjišťovány. Vývojový diagram tohoto algoritmu je zobrazen v Příloha C.2

Algoritmus s dělením impulsní charakteristiky na bloky různých délek – verze s rovnoměrným rozložením výpočetní náročnosti MBUL je relativně složitější než předchozí dva. Jeho vývojový diagram uvádí Příloha C.3. Opět obsahuje část hledající způsob rozdělení impulsní charakteristiky dle zákonitostí uvedených v kapitole 4.2.3 a uložení velikostí bloků do vektoru *lst*. Jelikož se ale velikost bloku nemění s každým blokem, je potřeba rozlišit, zda došlo ke změně a v tom případě nejprve získat nový obraz bloku vstupní posloupnosti *x* pomocí FFT o délce 2*aktK*. Mechanismus metody přičtení přesahu je v tomto případě také mírně odlišný.

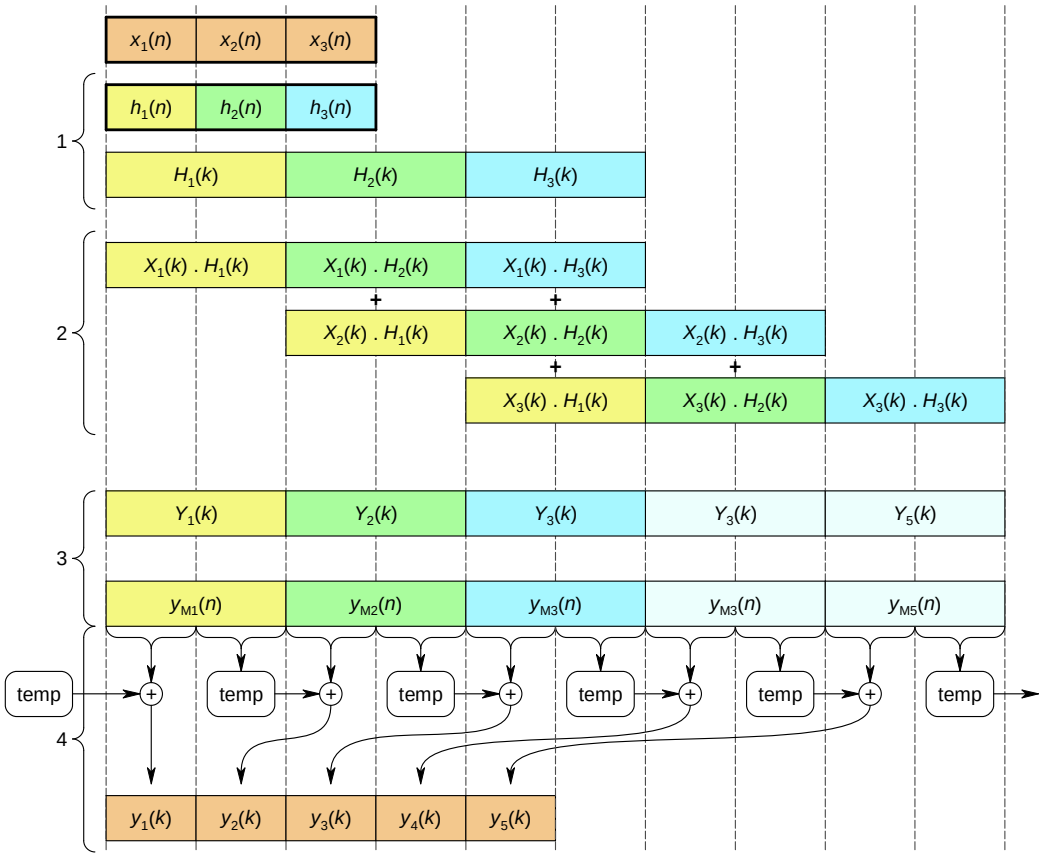
5.2 Algoritmy pracující s FDL

Jak již bylo uvedeno v kapitolách 4.2.2 a 4.2.4, metody využívající FDL značně snižují počet potřebných zpětných Fourierových transformací tím, že je metoda přičtení přesahu realizována již v kmitočtové oblasti. Použitím jednoduché FDL můžeme tedy značně snížit výpočetní náročnost algoritmu výpočtu, ovšem pouze za předpokladu dělení IR na bloky o velikosti blízké k optimální. To je ale v rozporu s požadavky na nízkou latenci (kap. 4.3) a právě tento problém je pak důvodem vzniku metody využívající dvojité FDL.

Narozdíl od předešlých algoritmů, které nepoužívají FDL, byly tyto algoritmy v prostředí Matlab implementovány jako funkce, jejichž vstupním parametrem je (kromě jiných) přímo blok vstupního signálu. Výstupem funkce je pak odpovídající blok výstupního signálu. Všechny ostatní potřebné proměnné včetně přípravy obrazu IR v kmitočtové oblasti musejí být připraveny zvláštním skriptem. Jde o obdobný postup, který je používán v technologii VST (bude blíže popsáno v kapitole 6).

5.2.1 Algoritmus využívající jednoduché FDL

Princip funkce jednoduché FDL pro tři bloky vstupního signálu a impulsní charakteristiku rozdělenou do třech bloků téže délky je naznačen na Obr. 5.1. Prvním krokem je předpřipravení obrazů IR v kmitočtové oblasti podle principů metody přičtení přesahu (detailněji na Obr. 4.4). Ve druhé části dochází ke spektrálnímu násobení obrazu aktuálního vstupního bloku $X_i(k)$ s obrazy bloků impulsní charakteristiky $H_j(k)$ a přičtení těchto součinů k předchozím hodnotám do výpočetní vyrovnávací paměti (bufferu), která je naznačena v části třetí – $Y_i(k)$.



Obr. 5.1: Znázornění funkce algoritmu jednoduché FDL.

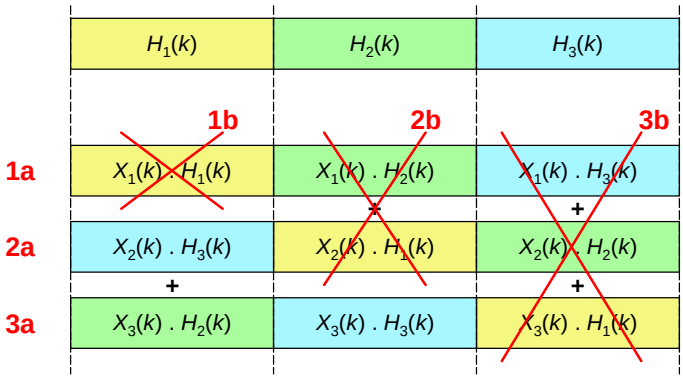
Jelikož bylo sečtení těchto součinů provedeno již v kmitočtové oblasti, došlo k očekávanému uspoření mnoha aplikací zpětné Fourierovy transformace. Po převedení jednotlivých bloků výpočetního bufferu zpět do časové oblasti ($y_{Mi}(n)$) je ve čtvrtém kroku za použití metody přičtení přesahu sestaven výstupní signál.

Výpočetní vyrovnávací paměť tak, jak je naznačena na Obr. 5.1, je zbytečně velká. S každým příchozím blokem vstupního signálu je totiž po přičtení všech spektrálních součinů na příslušnou pozici v paměti aktuální blok paměti převeden zpět do časové oblasti a již dále nevyužit. To umožňuje zavedení kruhové paměti, tedy takové, která pro indexaci vnitřní pozice využívá operace modulo. Za použití této techniky je dostatečná velikost této výpočetní paměti rovna délce obrazu impulsní charakteristiky.

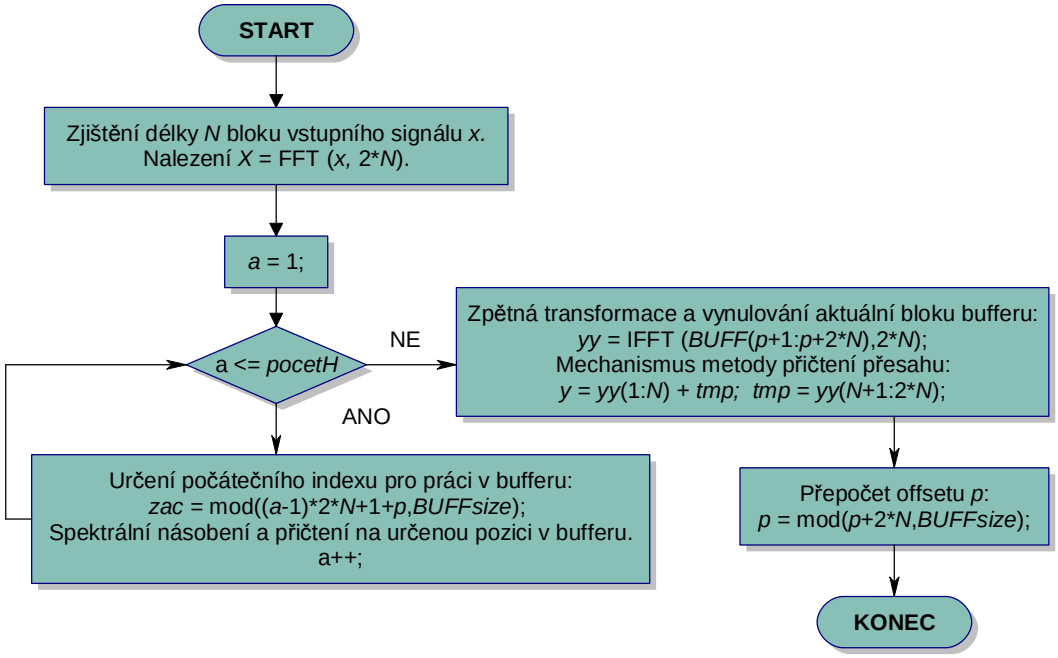
Princip práce s takovouto pamětí je naznačen na Obr. 5.2. V kroku 1a dojde nejprve k přičtení součinů k blokům bufferu, v kroku 1b pak k přečtení hodnoty aktuálního bloku, jeho IFFT a vynulování. Tak je tento blok bufferu uvolněn a připraven pro práci s dalším blokem vstupního signálu.

Pro testování funkce algoritmu jednoduché FDL v prostředí Matlab slouží skript, který řeší rozdělení impulsní charakteristiky podle zadané délky bloku a předpřipravení obrazu IR v kmitočtové oblasti; zjištění velikosti výpočetního vyrovnávací paměti *BUFF* a proměnné *tmp*, sloužící pro realizaci přičtení přesahu, jejich inicializaci a vynulování. Je také třeba inicializovat index *p*, který slouží jako offset pro posun ve výpočetní vyrovnávací paměti s každým blokem vstupního signálu, na hodnotu $p = 0$. Dále již skript dělí vstupní signál do bloků a ve smyčce jej posílá na vstup výpočetní funkce jednoduché FDL a zapisuje bloky výstupního signálu za sebe do vektoru výstupního signálu.

Vývojový diagram samotné výpočetní funkce jednoduché FDL je na Obr. 5.3. Po převedení bloku vstupního signálu do kmitočtové oblasti je provedeno spektrální násobení s obrazy bloků impulsní odezvy a přičtení do výpočetního bufferu *BUFF* dle výše zmíněného způsobu využívajícího funkce modulo. Následně je podle hodnoty indexu *p* vybrán aktuální blok z *BUFF*, provedena jeho zpětná Fourierova transformace a vynulování. Metodou přičtení přesahu je sestaven aktuální výstupní blok a přesah pro blok další je uložen do *tmp*. Posledním krokem je přepočet hodnoty indexu *p*.



Obr. 5.2: Princip práce s kruhovou výpočetní vyrovnávací pamětí.

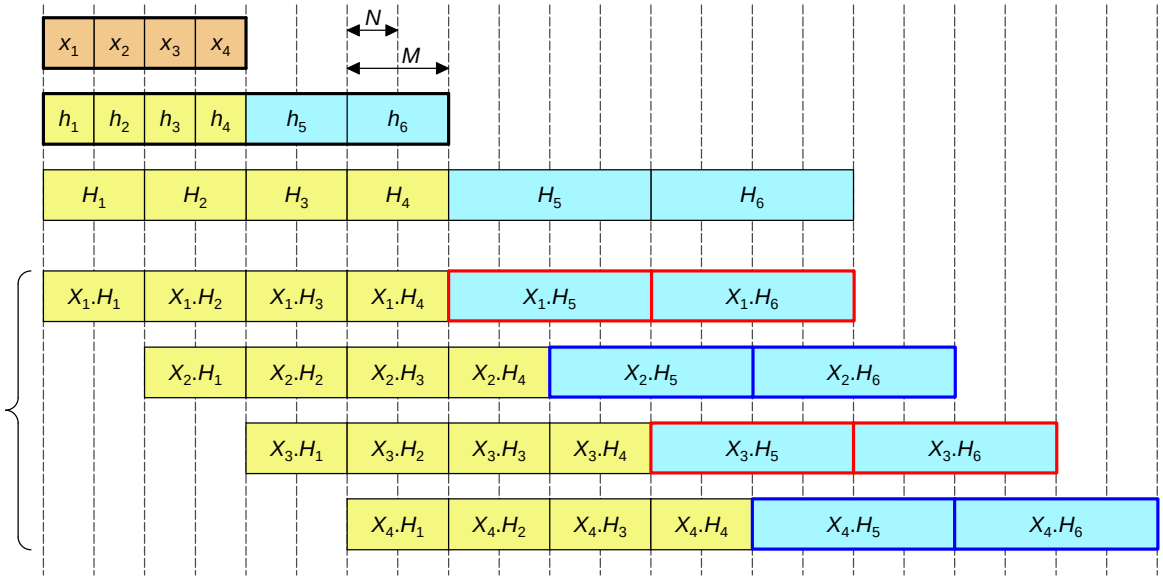


Obr. 5.3: Vývojový diagram algoritmu využívajícího jednoduché FDL.

5.2.2 Algoritmus využívající dvojité FDL

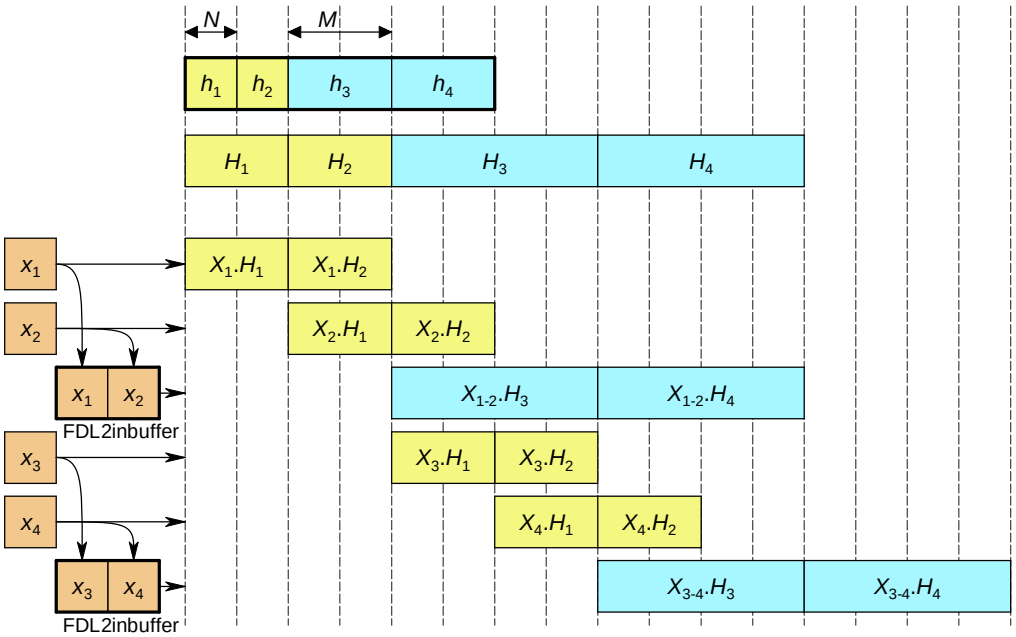
Princip této metody byl popsán v kapitole 4.2.4. Velikost bloku první FDL je dána požadovanou latencí, velikost bloku druhé FDL je určena dle vztahu (4.16). Použitím dělení IR na bloky dvou různých délek vyvstane problém s tím, jak správně připravit blok vstupního signálu pro výpočet druhé FDL.

Pokud by byl použit blok vstupního signálu délky N tak, že by byl pouze doplněn nulami na potřebnou délku a převeden do kmitočtové oblasti pomocí FFT o délce $2M$, výpočty a ukládání ve výpočetních vyrovnávacích pamětech by probíhalo způsobem naznačeným na Obr. 5.4. Jak je vidět, přičítání do výpočetního bufferu (označen závorkou) druhé FDL v tomto případě probíhá odlišně od výše zmíněného principu. K překrytí bloků součinů totiž dochází pouze s každým R-tým blokem vstupního signálu, kde $R = M / N$. Tak se vlastně celá druhá FDL počítá R-krát. To si žádá použití R-násobného výpočetního bufferu (zde dvojnásobný – červené a modré orámování) a spolu se složitějším způsobem sestavení výstupu způsobuje minimální snížení výpočetní náročnosti celého algoritmu.



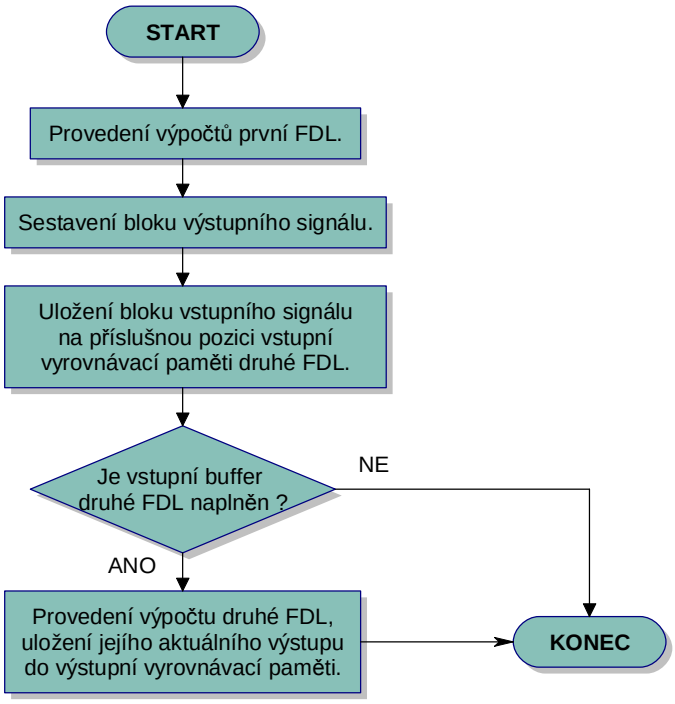
Obr. 5.4: Způsob práce s výpočetními buffery při pouhém doplnění bloku vstupního signálu nulami na délku 2**M* (pro přehlednost zjednodušené značení).

Řešení spočívá v použití vyrovnávací paměti na vstupu druhé FDL. Tato paměť velikosti *M* je plněna bloky vstupního signálu délky *N* a samotný výpočet druhé FDL proběhne až tehdy, když je tato paměť naplněna. Tím je dosaženo správného překrytí bloků součinů druhé FDL, která pak pracuje dle předpokladů. Zároveň je tím vyřešen problém určení počtu bloků první FDL – ten musí být minimálně roven poměru M / N, tedy *R*. Princip práce s výpočetními buffery i se vstupní vyrovnávací paměti druhé FDL v závislosti na blocích vstupního signálu je naznačen na Obr. 5.5.



Obr. 5.5: Princip algoritmu s dvojitou FDL využívajícího vyrovnávací paměť na vstupu druhé FDL.

Při samotné implementaci je vhodné použití separátní výpočetní vyrovnávací paměti pro každou FDL (na Obr. 5.5 označeno žlutou a modrou barvou). Ve výsledku se tedy jedná o dvojici zcela nezávislých algoritmů používajících jednoduché FDL, které pracují dle vývojového diagramu uvedeného na Obr. 5.3. Jediná vzájemná vazba je pomocí vyrovnávacích pamětí na vstupu a výstupu druhé FDL. Z výstupní vyrovnávací paměti (délka *M* vzorků) jsou vzorky postupně po blocích délky *N* přičítány k výstupním blokům první FDL a tak je sestaven výstup celého algoritmu. Zjednodušený vývojový diagram algoritmu využívajícího dvojité FDL je na Obr. 5.6.



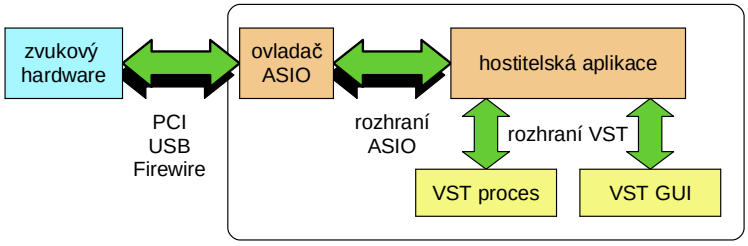
Obr. 5.6: Vývojový diagram algoritmu využívajícího dvojité FDL.

6 TECHNOLOGIE VST

Technologie VST (Virtual Studio Technology) byla vyvinuta firmou Steinberg jako systém pro zpracování zvukových signálů na počítači [18], [23]. Architekturu tohoto systému tvoří dvě základní části: hostitelské zařízení a tzv. VST plug-in modul. Hostitelským zařízením může být speciální software nebo i hardwarové zařízení, které umožňuje načtení, provoz a řízení VST plug-in. Samotné zpracování signálu pak provádí VST plug-in modul. V dalším popise budeme předpokládat hostitelskou aplikaci, tedy software.

Obě části spolu komunikují pomocí VST rozhraní tak, že hostitelská aplikace předává plug-in modulu již připravená zvuková data, řídící parametry, MIDI data a události. Tím je modul zbaven problémů se získáváním zvukových datových toků z HW zařízení počítače. Ty jsou zcela v režii hostitelské aplikace, která zase na oplátku neví nic o zpracování dat, která posílá přes rozhraní, VST plug-in modulem. Ten pro ni představuje jakousi černou skříňku, se kterou komunikuje přes rozhraní řízením parametrů, zasílá jí připravená zvuková data a dostává je, pro ni neznámým způsobem, zpracovaná zpět.

Zatímco verze VST 2.4 podporovala maximálně 8 přiřazených zvukových kanálů (počet vstupů a výstupů se může lišit), v nové verzi VST3 již nejsou plug-in moduly omezeny pevným počtem kanálů. Konfigurace vstupních/výstupních kanálů se může dynamicky měnit podle momentální aplikace modulu [18]. VST také umožňuje pro VST plug-in modul vytvořit grafické uživatelské rozhraní. To je však od samotného zpracování signálů odděleno a komunikace probíhá stejně jako u hostitelské aplikace, přes VST rozhraní. Tím je umožněno použití univerzálního uživatelského rozhraní hostitelské aplikace v tom případě, kdy není vytvořeno vlastní. Architektura technologie VST je zachycena na Obr. 6.1. [15]



Obr. 6.1: Architektura technologie VST.

V hostitelské aplikaci je často společně s VST implementována také technologie ASIO (Audio Stream Input-Output), která zajišťuje vstupně-výstupní operace se zvukovým hardwarem počítače (zvuková karta, streaming, datová média...). Pro bezproblémovou funkci a vývoj VST plug-in modulů je zapotřebí mít zvukovou kartu, která technologii ASIO podporuje. Mezi další používané technologie pro vstupně-výstupní operace patří např. Windows MME, DirectSound, CoreAudio a další. [13]

Podle toho, co dělají, lze VST plug-in moduly dělit do různých skupin. Nejhrubější dělení může být například na: [23]

- **VST nástroje** – jejich funkcí je generovat zvuk. Jedná se zpravidla o virtuální syntezátory nebo samplery.
- **VST efekty** – zpracovávají vstupní zvukový signál. Jedná se o virtuální provedení klasických zvukových efektů (reverb, delay, distortion atd.). Patří sem ale i efekty, které zvuk nijak neupravují – jsou to tzv. monitorovací efekty, jako např. ukazatel úrovně signálu atp.

- **VST MIDI efekty** – zpracovávají MIDI data, ovšem např. oproti virtuálnímu syntezátoru, jehož výstupem je zvukový signál, jsou výstupem těchto efektů opět MIDI data. Bývají většinou předřazeny virtuálním nástrojům (popř. hardwarovým zařízením) a provádí nějakou modifikaci MIDI dat, např. transpozici.

6.1 Sada nástrojů pro vývoj software (SDK)

Sada nástrojů pro vývoj software (SDK – Software Development Kit) technologie VST zahrnuje kód technologie a její aplikační rozhraní (API – Application Programming Interface) [13]. Výhodou tohoto vývojového prostředí oproti ostatním je fakt, že již od začátku (1996) se jedná o open source. Programový kód i aplikační rozhraní technologie VST je tedy dodáván ve formě zdrojových kódů. Jistá omezení však vyplívají ze specifických vlastností vývojového prostředí a platformy, na které je VST SDK použito:

- na operačním systému jsou závislé grafické funkce používané knihovnou VST GUI;
- na operačním systému a typu procesoru jsou závislé datové typy (64 a 32 bitové platformy) a binární formáty knihoven plug-in modulů (DLL ve Windows, raw Code resource v MacOS 9, Bundle v Mac OSX a library v BeOS a Unixu);
- na vývojovém prostředí je závislá konvence volání callback funkcí a vstupních bodů knihoven plug-in modulu. [13]

Zdrojové kódy VST SDK jsou uzpůsobeny tak, aby byla provedena podmíněná definice volacích konvencí, datových typů a grafických funkcí podle typických definovaných symbolů integrovaných vývojových prostředí. Jsou tak rozeznávány různé kompilátory napříč mnoha platformami.

6.2 Použitá šablona VST plug-in modulu

Při implementaci plug-in modulu v rámci této práce se nevycházelo přímo ze sady vývojových nástrojů VST SDK, které je ke stažení na webových stránkách výrobce. Namísto toho bylo využito šablony VST plug-in modulu `VST Template`, kterou zhotovil Ing. Jiří Schimmel, Ph.D.; konkrétně ve verzi 1.2 (viz. Příloha A).

Tato šablona vychází z VST SDK verze 2.4 a je k dispozici ve formě projektu pro vývojové prostředí MS Visual C++ s verzí projektu 8 (určeno pro MS Visual Studio 2005 a vyšší). [14] Projekt obsahuje několik konfigurací *Debug* a *Release*, např. možnost *Debug with GTPlayer EDU*, která sestavuje plug-in modul v příslušném adresáři tohoto programu, který je v šabloně obsažen. Šablona obsahuje celkem dva nástroje pro testování VST plug-in modulů – již zmíněný GTPlayer EDU a VST Plugin Analyser. Výsledný modul je sestaven do podoby dynamicky sestavované knihovny DLL.

Šablona obsahuje ve svém adresáři `source` několik souborů určených k editaci uživatelem. Celý vývoj VST plug-in modulu se tedy může omezit na práci s těmito soubory. Stručný přehled souborů a jejich účelu: [14]

`user_variables.h`

- Obsahuje pomocné proměnné potřebné pro implementaci algoritmu.

`vst_temp_defs.h`

- Definuje makra a výčtové typy, které určují základní vlastnosti plug-in modulu:
  - `def_canDoubleReplacing`: je-li definováno, je podporována funkce `processDoubleReplacing()`;
  - `def_Editor`: je-li definováno, je použit editor;
  - `def_tempInput`: je-li definováno, pracuje funkce `processReplacing` s kopií vstupních dat předanými z hostitelské aplikace;
  - `def_tempOutput`: je-li definováno, výstupní data nejsou ukládána přímo do výstupní vyrovnávací paměti hostitelské aplikace, ale do dočasné;
  - `def_VendorVersion`: definice verze;
  - `def_UniqueID`: definice jednoznačného identifikátoru (4 znaky);
  - `def_isSynth`: je-li definováno, plug-in modul se chová jako VST nástroj (viz. kapitola 6);
  - `def_ChNumber`: počet vstupních a výstupních kanálů;
  - `def_ProgNumber`: počet programů (ve skutečnosti pouze jeden, podpora více programů není implementována);
  - `def_RowsInEditor`: počet řádků parametrů v editoru.
- Definuje pojmenované konstanty indexů parametrů pomocí výčtového typu `tagTemplateParameter`.

`vst_temp.cpp`

- Definuje konstanty:
  - `char def_DefaultProgName[kVstMaxProgNameLen]:` standardní jméno programu;
  - `char def_VendorString[kVstMaxVendorStrLen]:` textový identifikátor výrobce;
  - `char def_ProductString[kVstMaxProductStrLen]:` textový identifikátor produktu;
  - `char def_EffectName[kVstMaxEffectNameLen]:` jméno efektu;
  - uživatelem definované konstanty pro algoritmus.
- Definuje reálné hodnoty, vlastnosti a chování parametrů algoritmu pomocí pole struktur `unsortedParams`. Pomocí struktury `ParamInfo` lze přímo definovat např. minimální, maximální a výchozí hodnotu parametru, jeho jméno, jednotku a typ zobrazení. (podrobněji v [14])
- Obsahuje funkce:
  - `setUserVariables` pro inicializaci a `clearUserVariables` pro deinicializaci proměnných definovaných `user_variables.h`;
  - `parameterChanged`: funkce pro práci s parametry, je volána při každé jejich změně.; není ji nutno implementovat, pokud nevyžadujeme řízení parametrů algoritmu změnou parametrů uživatelského rozhraní;
  - pro implementaci vlastního algoritmu zpracování signálů:
    `processReplacing` (formát dat je `float` – 32 bitů),
    `processDoubleReplacing` (formát dat je `double` – 64 bitů).

7 IMPLEMENTACE VST PLUG-IN MODULU

Jak již bylo uvedeno, při implementaci vybraného algoritmu do podoby VST plug-in modulu se v rámci této práce vycházelo ze šablony `VST Template`. Zároveň bylo záměrem zapouzdřit veškeré proměnné a funkce potřebné pro vlastní výpočet do vlastní třídy. Pro implementaci výpočtu konvoluce byly vybrány algoritmy pracující s FDL, přičemž algoritmus s jednoduchou FDL byl pouze odrazový můstek k algoritmu s dvojitou FDL. Výchozí algoritmy implementované v prostředí Matlab tedy byly modifikovány na dvoukanálové verze a přepsány tak, aby využívaly funkcí prostředí C++ se všemi náležitostmi. K projektu bylo také využito knihoven pro matematické funkce a knihovny FFTW implementující výpočet rychlé Fourierovy transformace.

7.1 Knihovna FFTW

Pro realizaci výpočtu FFT a IFFT byla vybrána externí knihovna FFTW (viz. Příloha A). Tato knihovna obsahuje velké množství transformací, ať již pro různé kombinace formátu vstupních a výstupních dat FFT (reálná, komplexní v několika formátech) nebo podporu vícerozměrných transformací. Hlavní výhodou této knihovny však je, že pro výpočet FFT (IFFT) nepoužívá jeden fixní algoritmus, ale výpočetní algoritmus je vyladěn pro dosažení nejvyššího výkonu na konkrétním hardware. [6]

Samotný výpočet dané transformace je rozdělen do dvou částí. Nejprve je pomocí plánovače zjištěn nejefektivnější způsob výpočtu požadované transformace na daném HW a tento způsob je popsán pomocí struktury označované jako plán. Podle tohoto plánu je později provedena samotná transformace. Uvažujeme-li dnešní vysoce výkonné aplikace, které provádějí velké množství FFT a IFFT, je počáteční zpoždění potřebné pro nalezení plánu, vzhledem k dosažení nejvyšší možné efektivity výpočtu, zanedbatelné.

Následující příklad vytvoří plán pro jednorozměrnou FFT se vstupním vektorem reálných čísel a komplexním výstupem:

```
fftwf_plan fftwf_plan_dft_r2c_1d(int n, float *in,
           fftwf_complex *out, unsigned flags),
```

kde `n` udává logickou délku transformace, `in` je ukazatel na vstupní vektor datového typu `float`, `out` ukazuje na výstupní vektor datového typu `fftwf_complex` a parametr `flags` udává způsob hledání optimálního algoritmu výpočtu. Možnými hodnotami jsou:

- `FFTW_ESTIMATE` – namísto testování mnoha různých postupů je zjednodušeným způsobem rychle sestaven plán, který ale není příliš optimalizován.
- `FFTW_MEASURE` – je provedeno několik zkušebních transformací a je vybrán nejefektivnější způsob. Tento proces zpravidla trvá několik sekund.
- `FFTW_PATIENT` – obdoba předchozího, ovšem je testováno více způsobů výpočtu. Vyžaduje několikanásobně více času.
- `FFTW_EXHAUSTIVE` – obdoba předchozích, je však testováno ještě mnohem více způsobů výpočtu k nalezení optimálního plánu. Vyžaduje podstatně více času.
- `FFTW_WISDOM_ONLY` – k sestavení plánu využívá dříve získaných „zkušeností“.

Pozdější spuštění plánu se provede pomocí příkazu

```
fftwf_execute(fftwf_plan).
```

Je také nutno zmínit datový typ pro komplexní čísla používaný třídou FFTW. Výchozí formát pro uložení reálných čísel je datový typ `double` a pro uložení komplexního čísla je definován datový typ `fftw_complex` ve tvaru:

```
typedef double fftw_complex[2],
```

kde prvek `[0]` obsahuje hodnotu reálné složky a `[1]` hodnotu složky imaginární. Jelikož si ale při samotném zpracování zvukových signálů v plug-in modulu vystačíme s datovým typem `float`, můžeme při práci s knihovnou FFTW využít datový typ `fftwf_complex` definován jako:

```
typedef float fftwf_complex[2].
```

To vyžaduje úpravu tvaru předpony volaných funkcí třídy z `fftw_` na `fftwf_`. Pro úplnost ještě uveďme možnost použití `fftwl_`, požadujeme-li datový typ `long double`.

7.2 Třída myFDL

Všechny proměnné a funkce potřebné pro realizaci uvedených algoritmů jsou zapouzdřeny do třídy myFDL. Diagram této třídy je uveden v Příloha D. Všechny proměnné i většina metod třídy jsou deklarovány se specifikátorem `private`, jsou tedy přístupné pouze v metodách dané třídy a spřátelených funkcích. Jsou to zejména metody určené pro výpočet hodnot pomocných proměnných, korektní alokaci dynamických proměnných a předpřipravení obrazu impulsní charakteristiky v kmitočtové oblasti.

Přístupnými metodami třídy jsou tedy pouze ty, které je potřeba volat přímo ze šablony VST plug-in modulu. Jsou to:

`void loadIR(float irR[],float irL[], int delkaIRin)`

- Funkce pro načtení impulsní charakteristiky délky `delkaIRin` vzorků pro dva kanály reprezentované vektory `irR[]` a `irL[]`do vnitřní proměnné třídy.

`void changeN(int Nin)`, resp. `void DchangeN(int Nin)`

- Funkce slouží pro změnu velikosti bloku, která je použita při výpočtech metodou s jednoduchou, resp. dvojitou, FDL. S tím souvisí i přepočet mnoha hodnot včetně předpřipravení obrazu IR v kmitočtové oblasti se zvolenou délkou bloku. K tomuto účelu metoda sdružuje několik dalších `private` metod třídy.

`void isSetBypass(void)`

- Ošetření proměnných třídy, je-li ve VST plug-in modulu zapnut bypass. V tom případě je zvukový efekt vyřazen z řetězce a na výstup jsou zkopírována neupravená vstupní data. Před opětovným zařazením efektu (vypnutí bypass) je třeba vynulovat ty proměnné, ve kterých zůstala stará zvuková data, která by se jinak objevila na výstupu.

`void singleFDL(float** in, float** out,int sampleFrames)`

- Samotná výpočetní funkce metody využívající jednoduché FDL.

`void doubleFDL(float** in, float** out,int sampleFrames)`

- Samotná výpočetní funkce metody využívající dvojité FDL.

Obě výpočetní funkce mají stejné vstupní parametry jako funkce šablony `processReplacing`. Hodnota `sampleFrames` udává počet vzorků v bloku vstupního signálu, který předala hostitelská aplikace ke zpracování. Zápis vstupu a výstupu je zde řešen pomocí ukazatele na ukazatele. Pro získání ukazatele např. na levý výstupní kanál můžeme použít funkce šablony `getOutputBuffer` a nebo přímo zápis:

```
float *outLeft = out[0].
```

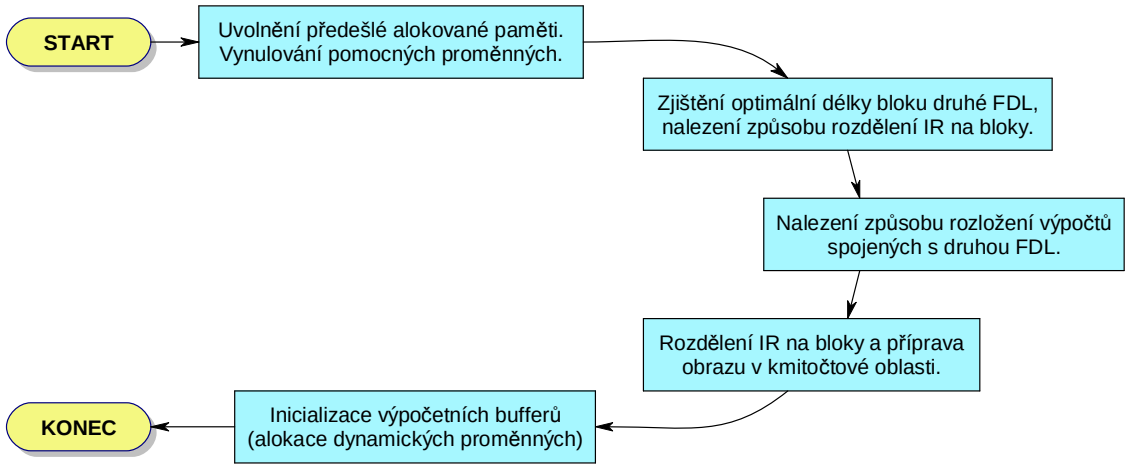
V plug-in modulu je pro výpočet konvoluce využito metody s dvojitou FDL, další text tedy bude zaměřen pouze na metody spojené s tímto způsobem výpočtu.

7.2.1 Příprava proměnných třídy

Způsob práce s proměnnými v rámci VST plug-in modulu je obdobný tomu, který byl použit při implementaci této metody v prostředí Matlab. I zde je tedy třeba před voláním samotné výpočetní funkce připravit potřebné proměnné, určit způsob rozdělení IR a připravit její obraz v kmitočtové rovině. Jádrem těchto příprav je funkce `DchangeN` přístupná z VST plug-in modulu, odkud je možné parametrem modulu přímo měnit délku výpočetního bloku metody. Co vše obstarává uvedená funkce je naznačeno na Obr. 7.1.

Za bližší popis stojí zejména třetí blok, tedy nalezení způsobu rozložení výpočtů spojených s druhou FDL. Jak bylo popsáno v kapitole 5.2.2, výpočet druhé FDL proběhne až tehdy, když je naplněn její vstupní buffer. To v praxi znamená, že je-li například délka bloku druhé FDL 32krát větší než u první FDL, celkem 31krát dojde k pouhému uložení vstupního bloku do bufferu a teprve po té, v kroku 32, k celému výpočtu. Je si ale třeba uvědomit, že počet bloků druhé FDL je obvykle velký a objem výpočtů potřebných k jejich zpracování obrovský. To zapříčiní nerovnoměrné „špičkové“ rozložení výpočetní náročnosti vedoucí k tomu, že v kombinaci s malým požadovaným zpožděním nelze výpočet včas dokončit. To se projeví výpadky signálu, lupáním atp. Je tedy potřeba nalézt efektivní způsob, jak rozložit výpočet spektrálního násobení obrazu bloku vstupního signálu (ze vstupního bufferu FDL2) s bloky obrazu impulsní charakteristiky.

Nalezenou optimální velikost bloku FDL2 (M) a způsob rozdělení impulsních charakteristik třech délek pro danou velikost bloku FDL1 (N) uvádí Tab. 1. Připomeňme, že počet bloků FDL1 odpovídá poměru $M / N = R$.



Obr. 7.1: Postup příprav před samotným výpočtem.

Tab. 1: Velikost bloku druhé FDL (*M*) a počet bloků jednotlivých FDL pro různé délky impulsní charakteristiky a zvolenou velikost bloku první FDL (*N*).

Délka bloku první FDL (*N*)	Délka impulsní charakteristiky								
	44100			120000			220000		
	M	počet bloků		*M*	počet bloků		*M*	počet bloků	
		FDL1	FDL2		FDL1	FDL2		FDL1	FDL2
32	1024	32	43	2048	64	58	2048	64	107
64	2048	32	21	2048	32	58	4096	64	53
128	2048	16	21	4096	32	29	4096	32	53
256	4096	16	10	4096	16	29	8192	32	26
512	4096	8	10	8192	16	14	8192	16	26
1024	4096	4	10	8192	8	14	16384	16	13

Nabízí se možnost rozložit násobení i do těch běhů funkce, kdy je plněn vstupní buffer pro další vstupní blok FDL2 (dále označeno jako načítací cyklus). Po naplnění vstupního bufferu tedy musí být zpracován minimálně jeden blok IR, jinak by nemohl být sestaven výstup. Další bloky IR jsou zpracovány následně. Jak je vidět z Tab. 1, při rozdělení impulsní charakteristiky na bloky mohou nastat tři situace:

a) je-li počet bloků FDL2 menší než *R*, stačilo by k rozložení výpočtu využít jen několika načítacích cyklů (stanou se výpočetními);

b) je-li počet bloků FDL2 větší než *R* a současně menší než 2*R*, je možno v každém načítacím cyklu mimo posledního zpracovat jeden blok IR (tuto hodnotu označme *FDL2part*) a v posledním cyklu jejich zbytek;

c) je-li počet bloků FDL2 větší než 2*R* (výjimečná situace), je možno provést rozložení obdobně jako v bodě b), přičemž najednou bude zpracován více než jeden blok IR (tedy *FDL2part* = 2, 3...) a jejich zbytek opět v posledním cyklu.

Při aplikaci rozložení výpočtu dle bodů b) a c) však může nastat situace, kdy počet bloků FDL je jen o málo menší, než nějaký celistvý násobek *R* (2*R*, 3*R*...). V tom případě bude počet bloků IR zpracovávaných v posledním výpočetním cyklu příliš vysoký (v nejhorším případě jich bude $R-1$) a tak nebude rozložení výpočtu efektivní a může docházet k výpadkům, lupání.

Použitý mechanismus tedy zohledňuje počet bloků IR, které by zbyly pro poslední cyklus výpočtu. Jako maximální počet byla testováním určena hodnota $R/4$. Pokud by při aktuální hodnotě *FDL2part* zbylo více než $R/4$ bloků, zvýší se hodnota *FDL2part* o jedna. Takto sice může být k výpočtu využito „zbytečně málo“ načítacích cyklů, nicméně je tak úspěšně zabráněno výpadkům.

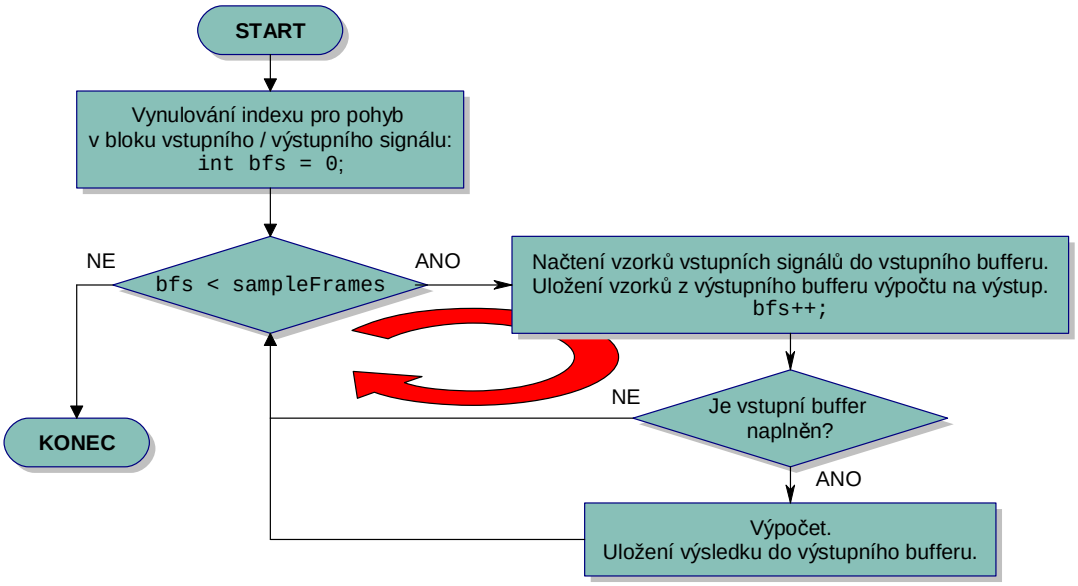
7.2.2 Výpočetní funkce využívající algoritmu s dvojitou FDL

Jak bylo uvedeno v kapitole 7.2, samotná výpočetní funkce metody výpočtu konvoluce použité pro implementaci v tomto VST plug-in modulu má z praktických důvodů stejné vstupní parametry, jako funkce šablony určená pro implementaci algoritmu číslicového zpracování signálu. Protože si vystačíme s datovým typem `float`, budeme pracovat s funkcí šablony `processReplacing`, která je deklarována jako:

```
void processReplacing (float** inputs, float** outputs,
                       VstInt32 sampleFrames)
```

V této souvislosti je také potřeba zrušit definici `def_canDoubleReplacing` v souboru `vst_temp_defs.h` a tak zakázat použití funkce `processDoubleReplacing`.

Tato funkce šablony je volána s každým příchozím blokem vstupního signálu z hostitelské aplikace. Problém z hlediska implementace metod s FDL je ten, že velikost tohoto vstupního bloku není konstantní. Počet vzorků v aktuálním vstupním bloku (`inputs`) i požadovaný počet vzorků, který má funkce vrátit na výstup (`outputs`), udává parametr `sampleFrames`. Protože implementovaná metoda výpočtu předpokládá vstupní i výstupní blok konstantní, zvolené velikosti, je nutno výpočtu předřadit pomocnou vyrovnávací paměť – buffer. Tento buffer je na vstupu i výstupu výpočtu a slouží pro oba zvukové kanály, princip jejich práce je naznačen na Obr. 7.2. Základem je smyčka, ve které jsou vstupní vzorky načítány do vstupního bufferu výpočtu a z výstupního bufferu jsou na výstup ukládány předchozí vypočtené hodnoty. Je-li vstupní buffer naplněn, proběhne výpočet. Vstupní buffer výpočtu může být plněn i během několika volání výpočetní funkce `doubleFDL`.



Obr. 7.2: Práce se vstupní a výstupní vyrovnávací pamětí výpočtu.

7.3 Integrace do šablony VST plug-in modulu

Aby bylo možné s třídou myFDL v šabloně pracovat, je nutno vytvořit objekt této třídy v souboru `user_variables.h`. Zde také deklarujeme další pomocné proměnné, které se v našem případě vztahují k možnosti změny parametru VST plug-in modulu, kterým se nastavuje velikost bloku první FDL, tedy požadované zpoždění. Soubor tedy obsahuje:

```
myFDL T;            // objekt tridy myFDL
bool isBypass;      // stav TRUE znaci zapnuti bypass
bool change;        // stav TRUE znaci zmenu parametru
int Nchange;        // nova hodnota odvozena z parametru
```

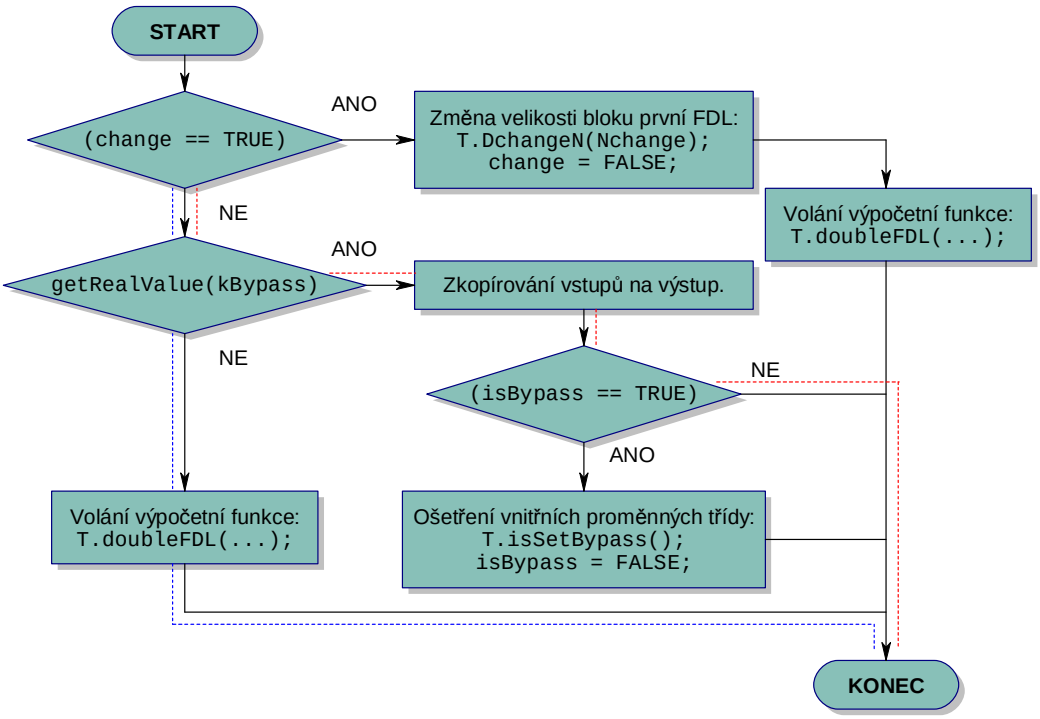
Výchozí hodnoty proměnných `change` a `isBypass` je nutno ve funkci šablony `setUserVariables` nastavit na `FALSE`.

V celé šabloně se tedy k metodám třídy myFDL přistupuje přes objekt této třídy `T`. Impulsní charakteristika není pevně uložena ve třídě myFDL, je ji potřeba načíst. To je provedeno v konstruktoru šablony voláním funkce `loadIR`. Jako referenční je touto metodou načtena dvoukanálová impulsní charakteristika o délce 220000 vzorků, pomocí které bude simulován akustický prostor kostela s dobou dozvuku cca 5 sekund. Dalším

potřebným krokem, který je v konstruktoru třeba provést, je výchozí nastavení délky bloku první FDL voláním funkce `DchangeN`. To je třeba provést až po načtení IR, protože při volání této funkce již musí být k dispozici ve vnitřní proměnné třídy.

Vyvíjený VST plug-in modul má dva parametry, které může uživatel měnit. První z nich umožňuje uživateli zapnout bypass, tj. vyřadit funkci zvukového efektu. Má tedy pouze dvě hodnoty, zapnuto a vypnuto. Typ zobrazení parametru je tedy nastaven na `SHOW_ONOFF`. (viz kapitola 6.2, popř. [14]) Druhý parametr slouží spíše testovacímu účelu – umožňuje totiž změnit velikost bloku první FDL (a tím i zpoždění a výpočetní náročnost celého výpočetního algoritmu). Lze nastavit hodnoty 64, 128, 256, 512 a 1024 vzorků. Jelikož reálná hodnota parametru se pohybuje v rozmezí 0–1, je nutno aplikovat odpovídající přepočet jak pro nastavení velikosti bloku, tak pro zobrazení hodnoty parametru. Za tímto účelem byla funkce šablony `getParameterDisplay` doplněna typem zobrazení `SHOW_AtoB`.

Změna některého z parametrů je zachycena funkcí šablony `parameterChanged`, ve které je ošetřeno nastavení pomocných proměnných, které indikují změnu a popř. i udávají novou hodnotu odvozenou ze změněného parametru. Těmito proměnnými je pak řízen přesný průběh událostí ve funkci šablony `processReplacing`, tak jak je podrobněji naznačeno na Obr. 7.3. Za normálního stavu (modrá cesta), kdy nedošlo ke změně parametru řídícího velikost bloku první FDL a není zapnutý bypass, je volána výpočetní funkce. Ke zjištění aktivace bypass je využita funkce šablony, která sleduje přímo hodnotu parametru modulu `getRealValue(kBypass)`. Vrací-li tato funkce hodnotu 1, je bypass zapnut a vstupní data jsou bez úpravy zkopírována na výstup (červená cesta). Pomocná proměnná `isBypass` značí změnu – pouze v tom případě je nutné volat funkci `isSetBypass()` (důvody uvedeny v kapitole 7.2). Změní-li uživatel parametr řídící velikost bloku první FDL, je zavolána funkce třídy `DchangeN(Nchange)`, kde `Nchange` je nová hodnota velikosti bloku odvozená z příslušného parametru. Až po té je zavolána výpočetní funkce, která již pracuje podle aktuálních parametrů.



Obr. 7.3: Podmíněné větvení ve funkci `processReplacing`.

8 ZÁVĚR

Práce se zabývala různými metodami pro vytváření umělého dozvuku – tedy způsoby simulace akustických prostorů. Proto byl nejprve charakterizován princip vzniku dozvuku v přirozeném prostředí včetně fyzikálních mechanizmů, které se na jeho tvorbě podílejí a byl přiblížen vliv dozvuku na zvukový vjem člověka.

Druhá kapitola se již zabývala přístupem k simulaci poslechového prostoru. Byla zde zavedena souvislost mezi simulovaným prostorem a lineárním časově invariantním systémem a naznačen způsob jeho simulace pomocí binaurální impulsní charakteristiky systému. Bylo provedeno rozdělení simulací podle počtu výstupů ze simulačního systému a byly charakterizovány dva základní modely simulace akustických prostor: fyzikální a perceptuální. Byly uvedeny základní principy a výhody i nevýhody obou modelů a byl podrobněji popsán perceptuální model – včetně struktur používaných pro jednotlivé bloky simulující počáteční, resp. mnohonásobné odrazy.

Dále se již práce zabývá fyzikálním modelem simulace akustického prostoru, tedy takového modelu, který simuluje přesné šíření zvuku od zdroje k příjemci na základě příslušné impulsní charakteristiky. V této souvislosti byly zmíněny metody měření impulsní charakteristiky akustického prostoru. Byly popsány nejčastěji používané techniky včetně jejich kladů a záporů, podle kterých je možno zvolit optimální metodu pro konkrétní situaci.

Ve čtvrté kapitole byly uvedeny metody výpočtu konvoluce při číslicovém zpracování zvukových signálů v reálném čase pracující jak v časové, tak v kmitočtové rovině. Nalezení efektivní metody výpočtu konvoluce bylo základním předpokladem úspěšné realizace fyzikálního modelu simulace akustických prostorů. Bylo popsáno několik různých algoritmů pracujících jak bez dělení impulsní charakteristiky, tak s dělením impulsní charakteristiky na bloky stejných, resp. rozdílných délek. Byla také zmíněna možnost uplatnění metody přičtení přesahu již v kmitočtové rovině, které využívají metody pracující s některou z forem FDL. Byla provedená analýza výpočetní náročnosti uvedených algoritmů, která prokázala přínos metod pracujících s FDL k výraznému snížení výpočetní náročnosti výpočtu konvoluce.

V prostředí Matlab bylo implementováno a otestováno několik vybraných algoritmů. Kromě těch, které uplatňují metodu přičtení přesahu v časové rovině (algoritmus s dělením impulsní charakteristiky systému na bloky stejné délky a metoda s dělením impulsní charakteristiky na bloky různých délek ve dvou svých verzích), to byly zejména algoritmy pracující s FDL. Byly to algoritmy s dělením impulsní charakteristiky na bloky stejných, resp. dvou různých délek, s využitím jednoduché, resp. dvojité, FDL. Všechny uvedené algoritmy byly také úspěšně otestovány na reálných signálech.

V předposlední části byla charakterizována technologie VST, která se používá při zpracování zvukových signálů na osobních počítačích. Byly uvedeny základní principy, klasifikace VST plug-in modulů a zejména detailně popsána šablona, která umožňuje vlastní implementaci VST plug-in modulu.

Poslední část byla věnována popisu implementace digitálního zvukového efektu typu reverb, který využívá konvoluci signálu s impulsní charakteristiku akustického prostoru, do podoby VST plug-in modulu. Pro výpočet konvoluce byl vybrán algoritmus pracující s dvojitou FDL. Byla vytvořena třída myFDL zapouzdřující všechny proměnné a funkce potřebné pro výpočet a byl popsán způsob její integrace do šablony VST plug-in modulu. VST plug-in modul byl úspěšně odladěn a otestován.

SEZNAM POUŽITÉ LITERATURY

[1] ADRIAENSEN, Fons. Acoustical Impulse Response Measurement with ALIKI. In ZKM, Karlsruhe, Germany. *LAC2006 Proceedings : 4th International Linux Audio Conference*. [s.l.] : [s.n.], 2006. s. 9-14. Dostupný z WWW: <http://lac.zkm.de/2006/papers/lac2006_proceedings.pdf>.

[2] ADRIAENSEN, Fons. Design of a Convolution Engine optimised for Reverb. In ZKM, Karlsruhe, Germany. *LAC2006 Proceedings : 4th International Linux Audio Conference*. [s.l.] : [s.n.], 2006. s. 49-53. Dostupný z WWW: <http://lac.zkm.de/2006/papers/lac2006_proceedings.pdf>.

[3] BALÍK, Miroslav. Simulace akustických prostorů - modely pro simulace v reálném čase. *Internetový časopis Elektrorevue* [online]. 2001, č. 16 [cit. 2008-10-10]. Dostupný z WWW: <http://www.elektrorevue.cz/clanky/01016/index.html>.

[4] DUDA O., Richard. *HEAD RELATED TRANSFER FUNCTION* [online]. 1996-2000 , 6/26/00 [cit. 2008-10-28]. Dostupný z WWW: <http://interface.cipic.ucdavis.edu/CIL_tutorial/3D_HRTF/3D_HRTF.htm>.

[5] FARINA, Angelo, ANDERS, Torger. *Real-time partitioned convolution for ambiophonics surround sound* [online]. 2001 [cit. 2008-10-12]. Dostupný z WWW: <http://www.acoustics.net/objects/pdf/article_farina04.pdf>.

[6] FRIGO, Matteo, JOHNSON, Steven G. *FFTW for version 3.2.1* [online]. 2009. [cit. 2009-3-12]. Dostupný z WWW: <http://www.fftw.org/fftw3.pdf>.

[7] GARCÍA, Guillermo. *Optimal Filter Partition for Efficient Convolution with Short Input/Output Delay*. In Proceedings of the AES 113th Convention. Los Angeles, California, USA. October 2002.

[8] GARDNER G., William. Reverberation algorithms. In KAHRS, Mark, BRANDENBURG, Karlheinz. *Applications of Digital Signal Processing to Audio and Acoustics*. Boston : Kluwer Academic Publishers, 1998. s. 85-131. ISBN 0-7923-8130-0.

[9] KOPECKÝ, Petr. Měření a analýza elektroakustických soustav na modelech. In DSP Research Group, . *Konference MATLAB 2003*. [s.l.] : [s.n.], 2003. Dostupný z WWW: <http://dsp.vscht.cz/konference_matlab/matlab03/kopecky.pdf>.

[10] PICINALI, L. *Techniques for the Extraction of the Impulse Response of a Linear and Time-Invariant System*. In Proceedings of the DRMN-06: DMRN Doctoral Research Conference 2006. London, Great Britain: University of London, 2006.

[11] REICHL, Jaroslav, VŠETIČKA, Martin. *Encyklopedie fyziky : Odraz zvuku, pohlcování zvuku* [online]. 2006-2008 [cit. 2008-10-26]. Dostupný z WWW: <http://fyzika.jreichl.com/index.php?sekce=browse&page=197>.

[12] RINDEL, Jens Holger, LYNGE CHRISTENSEN, Claus. *Room acoustic simulation and auralization : How close can we get to the real room?* [online]. 2003 [cit. 2008-10-28]. Dostupný z WWW: <http://www.odeon.dk/pdf/WESPAC8.pdf>.

[13] SCHIMMEL, Jiří. *Implementace číslicového zpracování signálů pomocí technologie VST*. Studijní text pro předmět MCSI, VUT v Brně, 2007.

[14] SCHIMMEL, Jiří. *Šablona VST plug-in modulu*. Studijní text pro předmět MCSI, VUT v Brně, 2008.

[15] SCHIMMEL, Jiří. *Zpracování signálů v reálném čase*. Studijní text pro předmět MCSI, VUT v Brně, 2007.

[16] SMÉKAL, Zdeněk. *Číslicové zpracování signálů*. Elektronická skripta, VUT v Brně, 2008.

[17] SMÉKAL, Zdeněk. *Signály a soustavy*. Elektronická skripta, VUT v Brně, 2008.

[18] STEINBERG : VST3: New Standard for Virtual Studio Technology [online]. c2009 , 2009 [cit. 2009-05-02]. Dostupný z WWW: <http://www.steinberg.net/en/company/steinberg_technology/virtual_studio_technology.html>.

[19] SYROVÝ, Václav. *Hudební akustika*. 1. vyd. Praha : Akademie múzických umění, 2003. 427 s. ISBN 80-7331-901-2.

[20] VLACHOVÁ , Magda. *Matematicko-fyzikální web : Fyzika - odraz vlnění, stojaté vlnění* [online]. 2006-2008 , 11.8.2008 [cit. 2008-10-26]. Dostupný z WWW: <http://mfweb.wz.cz/fyzika/141.htm>.

[21] VLACHÝ, Václav. *Praxe zvukové techniky*. Druhé aktualizované vydání. Nakladatelství Muzikus, Praha, 2000. 257 s. ISBN 80-86253-05-8.

[22] VLACHÝ, Václav. Kurzy zvukové techniky XX : Přirozený a umělý dozvuk - tři alternativy pro získání prostorové perspektivy. *MUZIKUS : Magazín pro muzikanty*. 2008, roč. XVIII, č. 12, s. 38.

[23] Wikipedia contributors. Virtual Studio Technology [Internet]. Wikipedia, The Free Encyclopedia; 2009 Apr 21, 22:02 UTC [cit. 2009-5-2]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Virtual_Studio_Technology&oldid=285324122>.

[24] ZÖLZER, U. Digital Audio Signal Processing, 1st ed. New York: McGraw-Hill, Inc., 1997, 290 p. ISBN 0-47-197226-6.

SEZNAM PŘÍLOH

A. Obsah přiloženého CD

Adresář `MATLAB_funkce`	Obsahuje funkce, které byly v rámci této práce implementovány v prostředí Matlab[1]
Adresář `FFTW`	Obsahuje zdrojové soubory a dokumentaci knihoven pro výpočet FFT a IFFT, které byly při implementaci použity
Adresář `VST_Template`	Obsahuje výchozí šablonu pro VST plug-in modul v prostředí MS Visual C++[2] ve verzi 1.2
Adresář `VST_plugin`	Obsahuje zdrojové soubory implementace zvukového efektu typu reverb do šablony VST plug-in modulu v prostředí MS Visual C++[2]
Adresář `VST_release`	Obsahuje zkompilovaný[3] VST plug-in modul v podobě DLL pro prostředí Windows

[1] verze 7.4.0.287 (R2007a)

[2] ve formě tzv. *solution* pro MS Visual Studio 2008

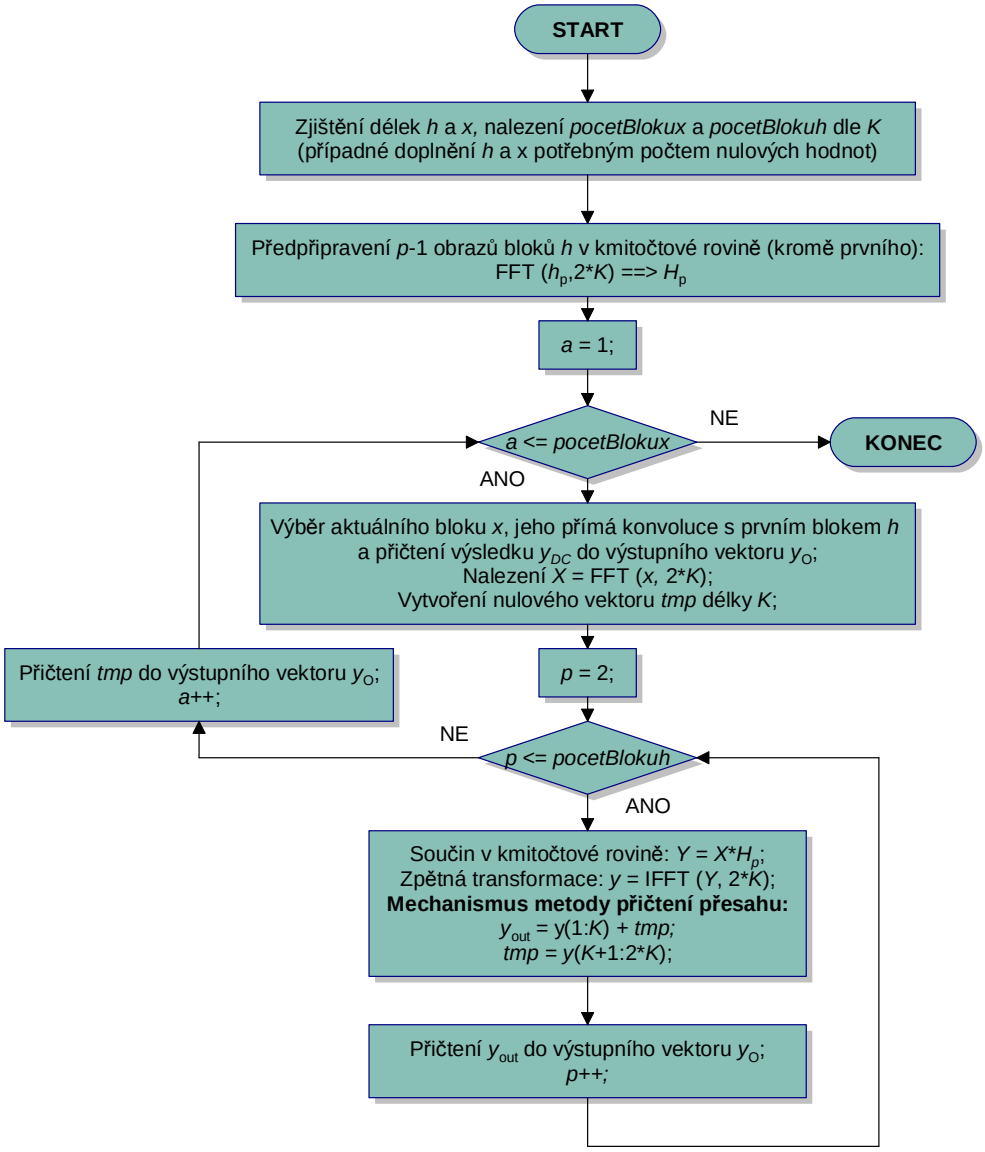
[3] použit MS Visual C++ compiler

B. Seznam funkcí implementovaných v prostředí MATLAB

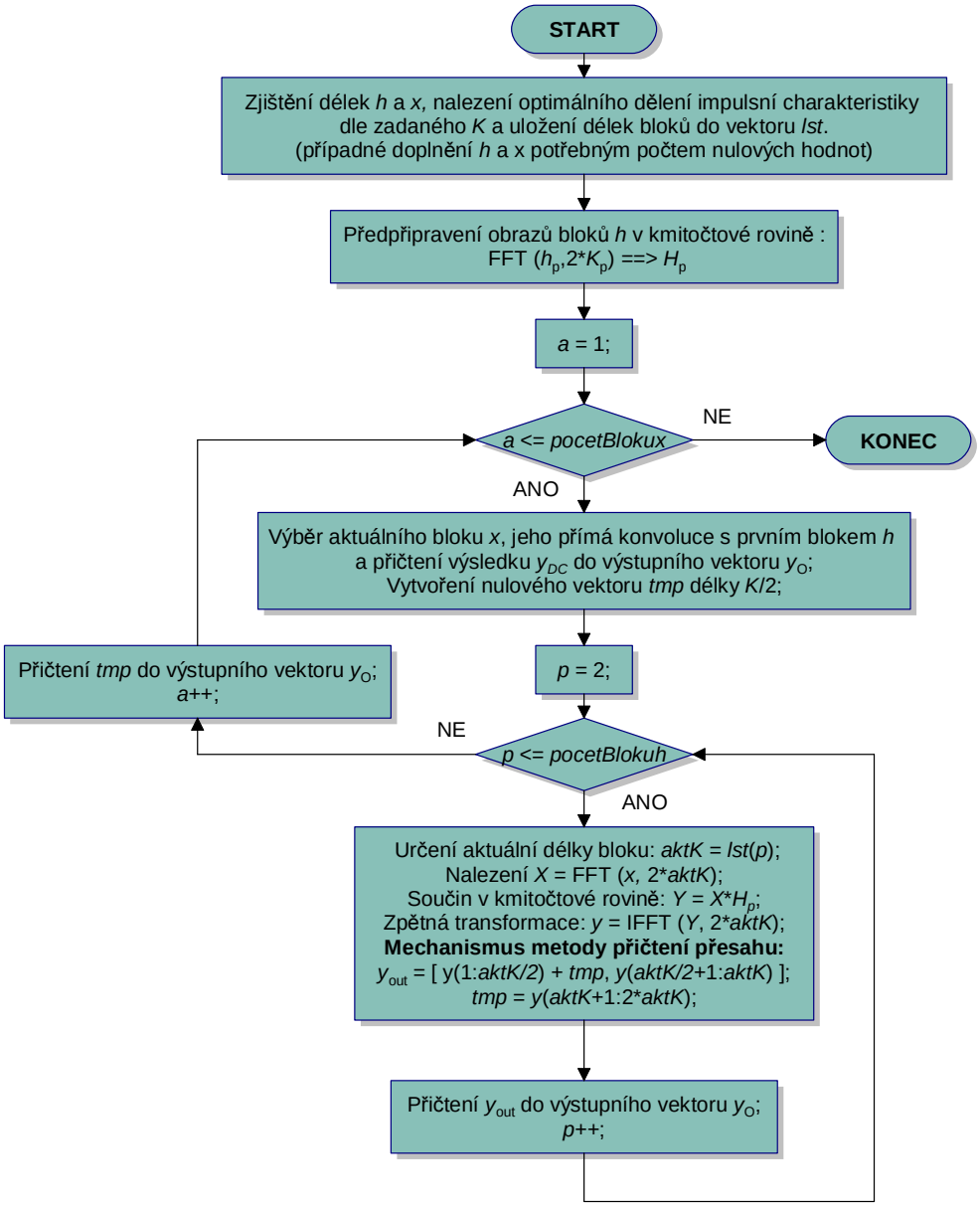
`T_MBUP`	Funkce pro výpočet lineární konvoluce dvou signálů pomocí algoritmu s dělením impulsní charakteristiky na bloky stejné délky
`T_MBUL`	Funkce pro výpočet lineární konvoluce dvou signálů pomocí algoritmu s dělením impulsní charakteristiky na bloky různých délek, verze s rovnoměrným rozložením výpočetní náročnosti
`T_MBMC`	Funkce pro výpočet lineární konvoluce dvou signálů pomocí algoritmu s dělením impulsní charakteristiky na bloky různých délek, verze s minimální výpočetní náročností
`singleFDL`	Výpočetní funkce využívající algoritmus s dělením impulsní charakteristiky na bloky stejné délky s využitím jednoduché FDL
`singleFDL_skript`	Skript pro testování funkce `singleFDL`
`doubleFDL`	Výpočetní funkce využívající algoritmus s dělením impulsní charakteristiky na bloky dvou různých délek s využitím dvojité FDL
`doubleFDL_skript`	Skript pro testování funkce `doubleFDL`

C. Vývojové diagramy algoritmů uplatňujících metodu přičtení přesahu v časové rovině

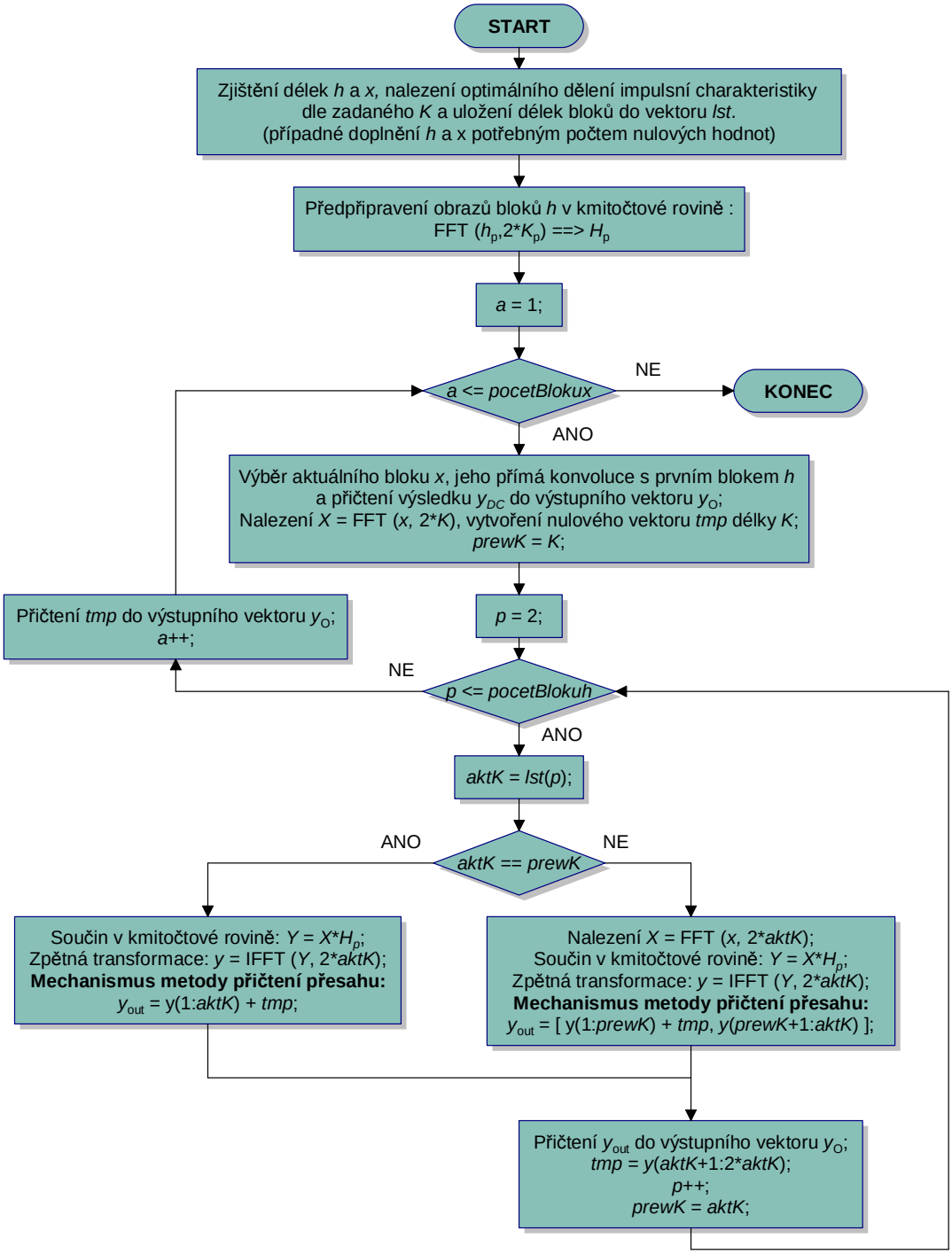
C.1 Vývojový diagram algoritmu MBUP



C.2 Vývojový diagram algoritmu MBMC



C.3 Vývojový diagram algoritmu MBUL



D. Diagram třídy myFDL

myFDL

Variables:

```
-pocetBlokuh:int    // pocet bloku IR singleFDL
-N:int              // delka bloku FDL1
-N2:int             // delka bloku FDL1 v km. oblasti
-delkaIR:int        // delka originalu IR
-BUFFsize:int       // delka vypoctoveho bufferu FDL1
-DpocetFDL1:int     // pocet bloku FDL1
-DpocetFDL2:int     // pocet bloku FDL2
-M:int              // delka bloku FDL2
-M2:int             // delka bloku FDL2 v km. oblasti
-BUFF2size:int      // delka vypoctoveho bufferu FDL2
-p:int              // index pro pohyb ve vypocetnim bufferu FDL1
-p2:int             // index pro pohyb ve vypoctovem bufferu FDL2
-indexFDL:int       // pomocna pro zjisteni stavu naplneni inputBuffer
-ratio:int          // pomer velikosti bloku FDL2/FDL1
-compratio:int      // pocet casti, na ktere bude rozdelen vypocet FDL2
-px[2]:int          // index pro kontrolu plneni FDL2inbuffer
-zac2[2]:int        // indexace prace s BUFF2
-FDL2part:int       // pocet bloku FDL2 zpracovanych v jednom cyklu vypoctu

-nn:float           // pomocna pro prevod deleni na soucin
-mm:float           // pomocna pro prevod deleni na soucin
-REAL:float*        // vstup FFT / vystup IFFT FDL1
-REAL2:float*       // vstup FFT / vystup IFFT FDL2
-IRoriginal:float*  // promenna pro ulozeni originalu IR
-OLtmp:float**              // pomocna pro realizaci metody pricteni presahu FDL1
-OLtmp2:float**             // pomocna pro realizaci metody pricteni presahu FDL2
-inputBuffer:float**        // vstupni buffer cele  FDL
-outputBuffer:float**       // vystupni buffer cele FDL
-FDL2inbuffer:float**       // buffer na vstupu FDL2
-FDL2outbuffer:float**      // buffer na vystupu FDL2

-COMP:fftwf_complex*        // vstup IFFT / vystup FFT FDL1
-COMP2:fftwf_complex*       // vstup IFFT / vystup FFT FDL2
-tCOMP2:fftwf_complex*      // pomocna pro rozlozeni vypoctu FDL2part
-HIR:fftwf_complex*         // promenna pro ulozeni obrazu IR pro FDL1
-HIR2:fftwf_complex*        // promenna pro ulozeni obrazu IR pro FDL2
-BUFF:fftwf_complex*        // vypoctovy buffer FDL1
-BUFF2:fftwf_complex*       // vypoctovy buffer FDL2

-p_forward:fftwf_plan       // plan pro FFT FDL1
-p_backward:fftwf_plan      // plan pro IFFT FDL1
-Dp_forward:fftwf_plan      // plan pro FFT FDL2
-Dp_backward:fftwf_plan     // plan pro IFFT FDL2
```

Methods:

```
    // inline funkce pro komplexni nasobeni a pricitani
-multAddArray(fftwf_complex *C,fftwf_complex *A,fftwf_complex *B,int delka):void
-initBuffer(void):void      // nastaveni vypocetnich bufferu
-zeroVariables(void):void   // vynulovani promennych tridy
-setN(int Nin):void         // nastaveni promennych singleFDL
-prepareHIR(void):void      // pripraveni obrazu IR pro singleFDL
-findM(void):void           // nalezeni optimalni delky FDL2
-DprepareHIR(void):void     // pripraveni obrazu IR pro doubleFDL
-DsetN(int Nin):void        // nastaveni promennych doubleFDL

+myFDL(void);               // konstruktor tridy
+~myFDL(void);              // destruktor tridy
    // funkce pro nacteni impulsni charakteristiky
+loadIR(float irR[],float irL[], int delkaIRin):void
+changeN(int Nin):void      // nastaveni delky bloku singleFDL
+DchangeN(int Nin):void     // nastaveni delky bloku FDL1 u doubleFDL
+isSetBypass(void):void     // osetreni vnitrnich promennych pri zapnuti bypass
    // samotna vypocetni funkce singleFDL
+singleFDL(float** in, float** out,int sampleFrames):void
    // samotna vypocetni funkce doubleFDL
+doubleFDL(float** in, float** out,int sampleFrames):void
```