Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Bakalářská práce

Zaokrouhlovací chyby v číslicových systémech

Autor práce: Tomáš Sak
Vedoucí práce: Ing. Jiří Lahoda, Ph.D.

Plzeň 2013

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš SAK**
Osobní číslo:	**E09B0417P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Název tématu:	**Zaokrouhlovací chyby v číslicových systémech**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Zpracujte přehled v současnosti používaných formátů čísel v číslicových systémech.
2. Analyzujte vliv struktury číslicového filtru a vliv formátu čísel na vzniku zaokrouhlovacích chyb.
3. Zhodnoťte dosažené výsledky.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah pracovní zprávy:	**20 - 30 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**
Seznam odborné literatury:	

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí bakalářské práce:	**Ing. Jiří Lahoda, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Konzultant bakalářské práce:	**Ing. Jiří Lahoda, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Datum zadání bakalářské práce:	**15. října 2012**
Termín odevzdání bakalářské práce:	**7. června 2013**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. října 2012

Abstrakt

Tato práce se zabývá vlivem struktury číslicového filtru a použitého formátu čísel na vznik zaokrouhlovacích chyb. Strukturální vliv je porovnáván pro čtyři typy struktur, a to strukturu přímou, paralelní, kaskádní a lattice-ladder. Dále zpracovává přehled v dnešní době nejvíce používaných formátů čísel v číslicových systémech, nejvíce je pak zaměřena na popis dvou těchto formátů a to s pevnou a plovoucí řádovou čárkou. Cílem práce je navrhnout jednotlivé struktury a zhodnotit výsledky, dosažené výpočetním programem MATLAB. Porovnat vliv jednotlivých struktur a formátů čísel, na základě rozdílnosti energie chybového signálu.

Klíčová slova

formát s pevnou řádovou čárkou, formát s plovoucí řádovou čárkou, číslicový filtr, struktura filtru

Abstract

Sak, Tomáš. *Rounding Errors in Digital Systems* [*Zaokrouhlovací chyby v číslicových systémech*]. Pilsen, 2013. Bachelor thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Jiří Lahoda

---

This thesis describes the effect of the structure of a digital filter, and used number representation on the creation of rounding errors. The structural effect is compared for four types of structures, direct, parallel, cascade and lattice-ladder structure. The main attention is paid to the most used number representation in digital systems nowadays. These two formats represent fixed and floating point arithmetic. The goal of this thesis is to design various structures and to evaluate the results achieved by computing program MATLAB and to compare the effect of the structures and number representation, based on divergence in energy of error signal.

Keywords

fixed point, floating point, digital filter, structure of filter

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

V Plzni dne 5. června 2013

Tomáš Sak

.....

Podpis

Poděkování

Tímto bych chtěl předně poděkovat vedoucímu bakalářské práce panu Ing. Jiřímu Lahodovi Ph.D. za jeho cenné profesionální rady, odbornou pomoc a metodické vedení práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

CPU	Central Processing Unit. Procesor/Mikroprocesor.
FP	Floating Point. Pohyblivá řádová čárka.
FPGA	Field Programmable Gate Array. Programovatelná hradlová pole.
FPU	Floating-Point Unit. Matematický koprocesor.
FX/FXP	Fixed Point. Pevná řádová čárka.
GPU	Graphic Processing Unit. Grafický procesor.
LSB	Least Significant Bit. Nejméně významný bit.
MSB	Most Significant Bit. Nejvýznamnější bit.

1

Úvod

Návrh číslicových filtrů tvoří v dnešní době společně s diskrétní Fourierovou a spektrální analýzou základ klasického číslicového zpracování signálu. Po masivním nástupu číslicové techniky a počítačů se postupem času začalo i v tomto odvětví přecházet od analogového zpracování filtru k číslicovému. Důvody lze snadno hledat ve větší přesnosti a snadnějším návrhu číslicových filtrů. Avšak od analogových filtrů se úplně neupustilo, vzhledem k jejich vhodnosti filtrace vysokofrekvenčních signálů.

Číslicový filtr může být algoritmus nebo obvod, jehož činností je zapříčiněna změna spektra výstupního diskrétního signálu. Pomocí filtru lze realizovat modely diskrétních systémů za použití algoritmických a aritmetických operací. Číslicové filtry se liší strukturou, tedy různou skladbou základních funkčních bloků: sčítačky, násobičky a jednotkového zpoždění. Právě vliv těchto struktur na vznik chyb při kvantování na určitý počet bitů je základem zkoumání této práce.

Každý číslicový filtr provádí svoje operace v určeném formátu čísel. Za nejzákladnější formáty čísel považujeme formáty s pevnou a plovoucí řádovou čárkou. Tyto formáty také více rozebírá právě tato práce a snaží se určit, který formát je vhodnější pro jakou strukturu filtru. Každý z těchto formátu disponuje výhodami a nevýhodami pro určité vlastnosti použitého číslicového filtru.

Pro ověření těchto vlivů je potřeba navrhnout každou strukturu zvlášť a pomocí výpočetního programu MATLAB provést jednotlivé výpočty a vykreslení grafů s důrazem na rozdíly mezi filtrem v plné přesnosti a jeho kvantovaným ekvivalentem. Tyto výsledky a jednotlivé formáty čísel jsou porovnány. Měřítkem citlivosti chyby je zvolena energie chybového signálu, kterou získáme součtem kvadrátů rozdílů impulsní odezvy v plné přesnosti a přesnosti odezvy omezené na požadovaný počet bitů.

2

Reprezentace numerických hodnot

V této kapitole nastíním některé ze způsobů reprezentace numerických hodnot v operační paměti počítače, registrech mikroprocesoru **(CPU)** či v matematickém koprocesoru **(FPU)**. Tyto hodnoty se pro výše uvedené aplikace ukládají v binárním (dvojkovém) formátu. Což znamená, že pro vyjádření takovéto hodnoty využijeme N bitů. Pomocí této N-tice bitů je možné vyjádřit 2^N navzájem odlišných stavů. Jeden bit je základní a zároveň nejmenší jednotkou informace, která se používá především v číslicové technice. Jeden bit reprezentuje informaci, získanou odpovědí na jednu otázku typu ano/ne, u které je předem dána pravděpodobnost obou odpovědí stejná (jinými slovy, u které nemáme žádnou předchozí informaci, která by jednu z odpovědí upřednostňovala). Tyto odpovědi můžeme označit binárními číslicemi 0 a 1. Jako alternativa k značení binárních číslic 0 a 1 jsou například symboly ano/ne *(yes/no)* nebo pravda/lež *(true/false)*.

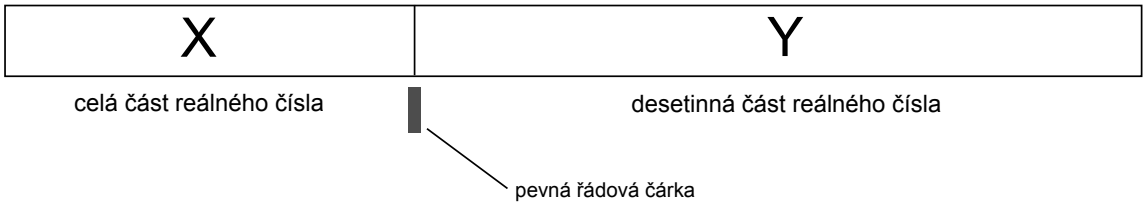
Nejpoužívanější zobrazení binárních stavů je zobrazení na interval celých kladných čísel *(Unsigned Integers)*, popřípadě na interval celých čísel *(Signed Integers)*. Nejčastějšími formami reprezentace číselných hodnot jsou pak formát pevné řádové čárky a formát plovoucí řádové čárky. Z anglické literatury jsou zavedeny zkratky **FX** nebo **FXP** *(Fixed Point)* pro formát pevné řádové čárky a **FP** *(Floating Point)* pro formát plovoucí řádové čárky. Princip obou formátů pro ukládání podmnožiny racionálních čísel je odlišný, ať už pro způsob reprezentace binárního čísla, tak pro aritmetické operace prováděné *CPU* nebo *FPU*.

Každá operační paměť počítače má konečný počet bitů a je jen na nás, jak je použijeme. To se zdá býti jedním z největších problémů dnešních programátorů, protože nedokáží efektivně využít všechny dostupné bity a některé promrhají ukládáním nepodstatných informací. Většina mikroprocesorů je však navržena na zpracovávání konstantního počtu bitů. Jako příklad může posloužit řada procesorů *x86*, která je nejlepší pro práci s 32 bity. Avšak při výpočtu s odlišným počtem bitů její efektivita klesá. Řešením tohoto problému je využití programovatelných polí **FPGA**, na kterých je možno vytvořit obvody pracující s různým počtem bitů.

2.1 Formát s pevnou řádovou čárkou *(Fixed point)*

Tato podkapitola se bude věnovat popisu méně používaného formátu a to formátu s pevnou řádovou čárkou. Pokusím se objasnit principy uchování dat v tomto formátu, aritmetické operace s těmito čísly a nakonec se pokusím porovnat výhody a nevýhody tohoto formátu oproti formátu s plovoucí řádovou čárkou.

Jak již napovídá jméno formátu, řádová čárka má přesně definovanou, pro všechna čísla neměnnou polohu. Ta poté rozděluje binární slovo na dvě různé části a to na část X, která obsahuje celou část reálného čísla a na část Y, která vyjadřuje desetinnou část reálného čísla (Obr. 2.1).



Obr. 2.1: Rozdělení na desetinnou a celou část binárního slova ve formátu pevné řádové čárky

Pomocí celé části X lze měnit rozsah čísel, protože každý bit této části reprezentuje kladnou mocninu dvojkového základu, naopak pomocí desetinné části Y lze měnit přesnost, protože každý bit této části reprezentuje zápornou mocninu dvojkového základu. Nejdůležitější při zvolení tohoto formátu je předešlé důkladné rozmyšlení potřebné přesnosti a rozsahu. Pokud chce programátor plně využít dostupnou délku slova ve formátu pevné řádové čárky, tak musí udělat jedno z těchto rozhodnutí. Při posunu o jeden bit doprava ztrácíme nejméně významný bit z angličtiny známy jako **LSB** *(Least Significant Bit)*. Pro opačný případ lze posunout všechny bity směrem doleva, aby nejvyšší bit byl na pozici **MSB** *(Most Significant Bit)*. Pro obě tyto operace je důležité zapamatovat si o kolik bitů došlo k posunu, aby bylo možné obnovit všechna čísla ve stejném rozsahu v pozdějším stádiu.

Přesnost a rozsah hodnot ve formátu pevné řádové čárky, lze reprezentovat například na bitovém slově o délce 8 bitů, kde celá část X zabíra 5 bitů a desetinná část Y pak zbylé 3 bity. Poloha řádové čárky v tomto uvedeném příkladu je tedy mezi 3. a 4. bitem slova (Tab. 2.1).

Číslo bitu	8	7	6	5	4	3	2	1
Váha bitu	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
Dekadická váha bitu	16	8	4	2	1	$0,5$	$0,25$	$0,125$

Tab. 2.1: Příklad rozsahu hodnot ve formátu pevné řádové čárky

2.1.1 Základní matematické operace

Zde bych se chtěl pokusit popsat základní matematické operace používané ve formátu s pevnou řádovou čárkou a to zejména ty nejpoužívanější jako sčítání, odčítání, násobení a dělení.

- Součet a rozdíl dvou hodnot A a B

  Jako první jsem zvolil popis dvou jednodušších operací prováděných ve formátu pevné řádové čárky a to sčítání a odčítání. Záměrně jsem tyto dvě operace spojil do společného textu, neboť pro ně platí stejná pravidla. Nejprve je nutné zjistit, zda se operace neprovádí se speciálními typy čísel jako jsou nekonečna. Poté je nutné zajistit, aby obě čísla byly ve stejném FX formátu, tedy aby u nich byla stejná velikost jak celé části X, tak i desetinné části Y.

- Součin dvou hodnot A a B

  Od jednodušších operací přecházíme ke složitějším, konkrétně součinu dvou hodnot. Zde není nutná podmínka rovnosti celé části X a desetinné části Y u obou čísel jako tomu bylo u předešlých operací. Je zde ale jiný problém, pakliže násobíme dvě čísla o N bitech, výsledné číslo pak má velikost 2N bitů (pokud násobíme dvě osmibitová čísla, výsledek je šestnáctibitový). Toto řešení je však značně nevýhodné, neboť budeme potřebovat dvakrát více bitů.

  Řešením je zmenšení desetinné oblasti Y u obou čísel o polovinu a posun o tuto zmenšenou oblast směrem doprava. Přebývající bity v celé části X se pak ořízno na požadovanou velikost formátu. To má však logicky za následek zmenšení přesnosti jak obou počátečních hodnot A a B, tak i výsledku vzniklého násobením takto upravených čísel. Dalším problémem, se kterým se budeme při této operaci často potýkat a na který si musíme dát obzvlášť velký pozor, je přetečení a podtečení.

- Podíl dvou hodnot A a B

  Stejně jako součet a rozdíl mají podobné vlastnosti i součin s podílem. Není tedy vyžadována stejná velikost celé X a desetinné Y části. Obdobný je i problém s velikostí výsledku, kde se dostaneme k číslu s větším počtem bitů, než měly hodnoty počáteční (A a B). Řešením je opět zmenšení desetinné části Y a posun směrem doprava. Logicky stejně dochází i ke zmenšení přesnosti výsledku.

2.2 Formát s plovoucí řádovou čárkou *(Floating point)*

Jako druhý představím nejpoužívanější formát v dnešní době a to formát s plovoucí řádovou čárkou (FP). Jak již název napovídá největší odlišností od formátu s pevnou řádovou čárkou (FX), je to, že řádová čárka není pevně určená a může se měnit její poloha.

Formát se skládá ze dvou částí a to **mantisy** a **exponentu**, obě tyto části mohou být jak kladné tak záporné. Mantisa má v sobě obsaženou informaci o významných číslicích, naopak exponent udává mocninu o určitém základu (nejčastěji 2,8,10 a 16), kterou jsou významné číslice v mantise děleny či násobeny. Původní hodnotu reálného čísla dopočítáme podle (rov. 2.1).

$$R = b^e \times m \qquad [-] \tag{2.1}$$

kde:

R - původní hodnota reálného čísla
b - báze, neboli radix
e - exponent
m - mantisa

Různé formáty hodnot, které reprezentují čísla ve formátu plovoucí řádové čárky FP se od sebe liší volbou báze **(radixu)**, ale i počtem bitů vyhrazených pro uložení mantisy či exponentu. Přehled těchto formátů specifikuje mezinárodní norma IEEE 754. Tato norma je v platnosti od roku 1985 a kromě specifikací uložení numerických hodnot obsahuje také způsoby provádění základních matematických operací, pravidla pro konverzi mezi jednotlivými formáty a pravidla pro práci se specifickými hodnotami jako je nekonečno.

2.2.1 Norma IEEE 754

V následujícím textu bych chtěl objasnit nejčastěji používané formáty pro uložení numerických hodnot ve formátu s plovoucí řádovou čárkou *FP*. Základní popis podmnožiny racionálních čísel může vyjadřovat vztah (rov. 2.2).

$$R = (-1)^s \times 2^{e-bias} \times m \qquad [-] \tag{2.2}$$

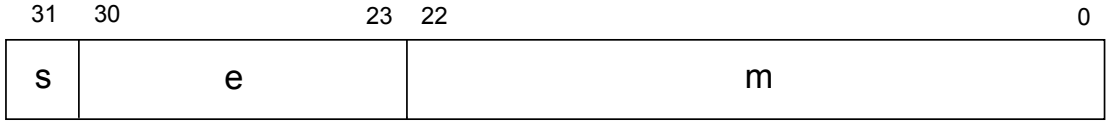
kde:

R - numerická hodnota racionálního čísla, umožňuje rozlišení speciálních typu čísel
2 - báze, neboli radix, u normy IEEE 754 je báze vždy dvojka, pro zjednodušení výpočtu u číslicových obvodů
e - exponent, kladný posunutý o hodnotu **bias**
bias - taková hodnota, která umožňuje zobrazení záporných čísel ve formátu s posunutou nulou
m - mantisa, vždy kladná
s - bit určující znaménko, pokud se rovná nule, je výsledná hodnota kladná a naopak

Podle těchto parametrů rozlišuje norma IEEE 754 na dva základní formáty a to na formát jednoduché přesnosti *(single)* a na formát s dvojitou přesností *(double)*.

- Formát jednoduché přesnosti *(single)*

  Tento formát je možná známější pod názvem **float**, což je zapříčiněno hojným rozšířením programovacích jazyků Java a C++. Charakteristickým znakem tohoto formátu je uložení hodnoty na 32 bitech. Ty jsou dále rozloženy na 3 části. Znaménko je uloženo na nejvyšším bitu *(MSB)*, dalších osm bitů napravo od *MSB* je vyhrazeno pro uložení exponentu v kódu s posunutou nulou a zbylých 23 bitů je využito pro uložení mantisy (Obr. 2.2). Velikost 32 bitů je zvolena pro svůj dobrý poměr mezi přesností a rozsahem hodnot, nehledě na to, že má optimální nároky na úložný prostor a většina v dnešní době používaných architektur má právě 32-bitovou sběrnici. Proto se s tímto formátem můžeme setkat u většiny **FPU** a **GPU**. Hodnota *bias* je v tomto formátu rovna 127 (rov. 2.3).



Obr. 2.2: Struktura formátu s plovoucí řádovou čárkou, jednoduchá přesnost

$$\begin{aligned} bias &= 2^{eb-1} - 1 \\ &= 2^{8-1} - 1 \\ &= 128 - 1 \\ &= 127 \qquad [-] \end{aligned} \tag{2.3}$$

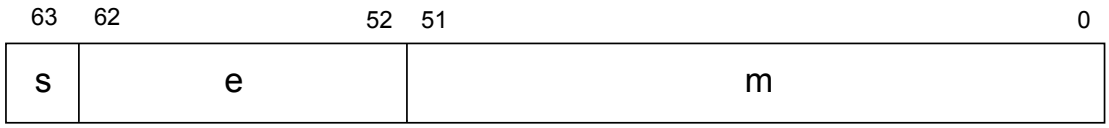
kde: *eb* - počet bitů určených pro exponent

Vztah vyjadřující popis podmnožiny racionálních čísel ve formátu jednoduché přesnosti pak můžeme popsat jako (rov. 2.4).

$$R = (-1)^s \times 2^{e-127} \times m \qquad [-] \tag{2.4}$$

- Formát dvojité přesnosti *(double)*

  Druhým ze základních formátů, které specifikuje norma IEEE 754 je formát s dvojitou přesností **(double)**. Jak již název napovídá, počet bitů do kterých je možné hodnotu uložit se zdvojnásobil. Takže tento formát operuje s 64 bity, narozdíl od 32 bitů formátu **single**. Bity jsou pak obdobně rozloženy na tři části. *MSB* je opět vyhrazen pro znaménko, dalších 11 bitů pro exponent a zbylých 52 pro uložení mantisy (Obr. 2.3). Hodnota *bias*, která určuje o kolik je exponent posunutý je v tomto případě 1023 (rov. 2.5).



Obr. 2.3: Struktura formátu s plovoucí řádovou čárkou, dvojitá přesnost

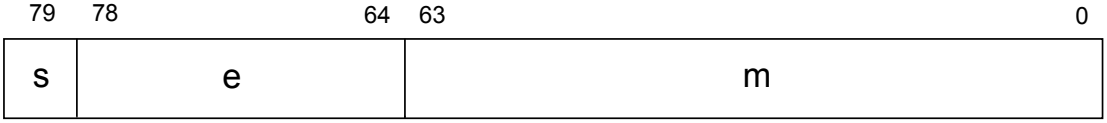
$$\begin{aligned} bias &= 2^{eb-1} - 1 \\ &= 2^{11-1} - 1 \\ &= 1024 - 1 \\ &= 1023 \qquad [-] \end{aligned} \tag{2.5}$$

Poté vztah, popisující podmnožinu racionálních čísel u formátu dvojité přesnosti je následující (rov. 2.6).

$$R = (-1)^s \times 2^{e-1023} \times m \qquad [-] \tag{2.6}$$

- Rozšířený formát *(extended)*

  Pro doplnění základních formátu *(single a double)* z normy IEEE 754, představím ještě rozšířeny formát *(extended)*, který norma také definuje. Ze všech tří formátu má tento největší kapacitu pro práci s hodnotami a to celých 80 bitů. Bity jsou opět analogicky rozloženy na tři části. *MSB* pro znaménko, 15 bitů pro exponent a 64 bitů pro mantisu (Obr. 2.4). Jeho hlavní určení je pro výpočty na *FPU*. Hodnotu *bias* spočteme podle již známého vzorečku (rov. 2.7) a vyjde nám 16383.



Obr. 2.4: Struktura formátu s plovoucí řádovou čárkou, rozšířená přesnost

$$\begin{aligned} bias &= 2^{eb-1} - 1 \\ &= 2^{15-1} - 1 \\ &= 16384 - 1 \\ &= 16383 \qquad [-] \end{aligned} \tag{2.7}$$

Obdobně jako v předešlých případech je vztah pro popis podmnožiny racionálních čísel vyjádřen (rov. 2.8).

$$R = (-1)^s \times 2^{e-16383} \times m \qquad [-] \tag{2.8}$$

Na konec této kapitoly bych se ještě rád zmínil o dalších formátech popsaných v normě IEEE 754. Těmito jsou například formát **ZX Spectrum**, jež byl používám na stejnojmenném osmibitovém počítači. Tento velmi populární počítač vyrobený v roce 1982, jsme v České republice znali spíše pod názvem jednoho z jeho klonů - Didaktik. Dalším formátem stvořeným firmou Borland pro programovací jazyk Turbo Pascal byl šestnáctibitový formát **real**. Jiným šestnáctibitovým formátem je **Minifloat**, používaný například na grafických procesorech firmy nVidia. Posledním a nejmenším formátem pro praktické využití je formát **Microfloat**, u kterého jsou hodnoty uloženy na pouhých osmi bitech.

2.2.2 Základní matematické operace

Stejně jako u formátu s pevnou řádovou čárkou, i zde se pokusím vypracovat přehled základních aritmetických operací a jejich realizaci u formátu s plovoucí řádovou čárkou.

- Součet a rozdíl dvou hodnot A a B

  Obdobně jako u formátu s pevnou řádovou čárkou začnu s popisem dvou jednodušších operací a to sčítáním a odčítáním. Záměrně jsem tyto dvě operace spojil do

společného textu, neboť pro ně platí stejná pravidla. Opět je nejprve nutné zjistit, zda se operace neprovádí se speciálními typy čísel. Poté je potřeba zjistit, které ze sčítaných hodnot A a B je větší. To se provede porovnáním hodnot exponentů u obou čísel. Pokud se exponenty nerovnají, je nutné vypočítat číselný rozdíl exponentů (rov. 2.9). Mantisu menšího z obou porovnávaných čísel pak přesuneme doprava právě o tolik bitů, kolik jsme vypočítali rozdílem obou exponentů. Poté je nutné zjistit hodnoty bitů určujících znaménka obou čísel a podle nich je rozhodnuto zda půjde o operaci sčítání či odčítání. Výsledek této operace má pak znaménkový bit rovný znaménkovému bitu většího z čísel (tzn. pokud bude větší číslo kladné, výsledek bude kladný a naopak).

Rizikem těchto operaci je přetečení či podtečení výsledku. K tomuto dochází při dosažení maximální respektive minimální hodnoty exponentu.

$$difference = e_1 - e_2 \qquad [-] \tag{2.9}$$

Pro názorné předvedení operace součtu u formátu s plovoucí řádovou čárkou poslouží pár příkladů. Jako první příklad jsem zvolil součet 2 hodnot, na kterém vidíme, že poslední tři hodnoty čísla B jsou nenávratně ztraceny díky závěrečnému zaokrouhlení. (rov. 2.10).

Př.1 A = 538623,6 ; B = 105,4239

$$\begin{aligned} 538623,6 + 105,4239 &= (5,386236 \times 10^5) + (1,054239 \times 10^2) \\ &= (5,386236 \times 10^5) + (0,001054239 \times 10^5) \\ &= (5,386236 + 0,001054239) \times 10^5 \\ &= 5,387290239 \times 10^5 \\ &\doteq 5,387290 \times 10^5 \qquad [-] \end{aligned} \tag{2.10}$$

Extrémním případem však může být, že menší z čísel nemá kvůli konečnému zaokrouhlení na součet vůbec žádný vliv. Což demonstruje druhý příklad (rov. 2.11).

Př.2 A = 538623,6 ; B = 0,0001054239

$$\begin{aligned} 538623,6 + 0,0001054239 &= (5,386236 \times 10^5) + (0,00000001054239 \times 10^5) \\ &= (5,386236 \times 10^5 + 0,00000001054239) \times 10^5 \\ &= 5,38623601054239 \times 10^5 \\ &\doteq 5,386236 \times 10^5 \qquad [-] \end{aligned} \tag{2.11}$$

- Součin dvou hodnot A a B

  Další a značně složitější aritmetickou operací je součin. Je komplikovanější z toho důvodu, že jsou hodnoty rozděleny na mantisu a exponent. Na začátku této operace budeme postupovat stejně jako sčítání či odčítání. Musíme zkontrolovat zda nebudeme operovat se speciálními hodnotami jako jsou nekonečna a tzv. **NaN** (Not a Number), které vznikají například po dělení nuly nulou. Výsledný exponent poté získáme tak, že sečteme exponenty obou hodnot A i B (rov. 2.12). Takto vzniklý výsledek pak ještě musíme ponížit o hodnotu **bias**, z důvodu zabránění posunu výsledku směrem doleva, který by mohl vzniknout kvůli umístění binární čárky v mantise za prvním bitem. Pokud dojde při součtu exponentů k přetečení (překročí-li se maximální hodnota) je výsledkem nekonečno. O znaménku pak rozhoduje znaménkový bit. Naopak jestliže dojde při součtu exponentů k podtečení (překročí-li se minimální hodnota) výsledkem je nula a o znaménku opět rozhoduje znaménkový bit. Výslednou mantisu pak spočítáme jako součin mantis hodnot A a B (rov. 2.13). Pokud dojde k přetečení výsledné mantisy o určitou hodnotu, je právě o tuto hodnotu exponent navýšen. Pokud se *MSB* mantisy nerovná jedné, posouvají se bity směrem doleva a exponent naopak klesá. Nakonec spočítáme výsledný znaménkový bit pomocí bitové operace ecxlusive-or obou znaménkových bitů hodnot A a B (rov. 2.14).

$$e = e_1 + e_2 \qquad [-] \tag{2.12}$$

$$m = m_1 \times m_2 \qquad [-] \tag{2.13}$$

$$s = s_1 \oplus s_2 \qquad [-] \tag{2.14}$$

  I u této operace musíme brát v potaz rizika s ní spjatá. A to hlavně riziko přetečení, podtečení a problémy spjaté se zaokrouhlováním hodnot.

- Podíl dvou hodnot A a B

  Poslední ze základních aritmetických operací je dělení. U této operace si musíme dávat obzvlášť velký pozor na situace, kdy budeme dělit nulou. Při tomto úkonu musí být zachováno znaménko, aby bylo možné detekovat, zda-li je výsledek kladné nebo záporné nekonečno. Samotná operace dělění lze provést třemi různými způsoby. Za prvé můžeme využít hardwarové děličky, kterou hojně využívají v dnešní době *FPU*. Druhým způsobem může být postupné odečítání dělence od dělitele, postup s kterým jsme se seznámili na základní škole, při jednoduchých výpočtech. A do třetice postup, který využívá k výpočtu převrácené hodnoty dělitele, se kterou je následně vynásoben dělenec.

2.3 Porovnání FX a FP

Jak již vyplývá z předchozího textu, používanějším z obou formátu je Floating point. Jednou z hlavních předností tohoto formátu je podpora FP operací na hardwarových jednotkách FPU. Ať už ve formě samostatného matematického koprocesoru nebo jako přímá součást modernějších mikroprocesorů. Další z nezanedbatelných výhod je určitě samotná existence normy IEEE 754, jež jasně definuje pravidla pro používání formátu s plovoucí řádovou čárkou. Tyto skutečnosti pak logicky vedly k tomu, že tento formát je primárně implementován jako datový typ v drtivé většině dnes používaných programovacích jazyků. Toto je obrovský rozdíl oproti formátu s pevnou řádovou čárkou, který je podporován pouze malým množstvím programovacích jazyků.

Kde však FX formát před svým používanějším protějškem vyhrává na celé čáře, jsou situace, kdy dopředu známe rozsah zpracovávaných hodnot a také požadovanou přesnost. Další nespornou výhodou je předem známá pozice řádové čárky, není totiž nutné společně s číselnou hodnotou uchovávat i pozici řádové čárky, což vede ke značné úspoře použitých bitů. Matematické operace prováděné v FX formátu jsou mnohdy rychlejší a jednodušší, oproti počítání jednoduchých aritmetických operací ve formátu FP bez použití matematického koprocesoru. Asi největší výhodou formátu s pevnou řádovou čárkou je dodržení požadované přesnosti u všech prováděných operací. Jak jsem naznačil dříve, součet nebo rozdíl dvou od sebe se lišících hodnot v mnoha řádech, může skončit naprostou ignorací menšího z čísel, nebo ještě hůře zacyklením výpočtů v nekonečných smyčkách.

Nevýhodou FX formátu jsou logicky případy, kde předem nevíme s jakými hodnotami budeme pracovat. Pokud bychom zpracovávali hodnoty s velkým poměrem mezi nejvyšší a nejnižší absolutní hodnotou, musíme počítat s alokováním velkého počtu bitů.

Nevýhody FP formátu můžeme hledat paradoxně v komplexnosti tohoto formátu. Neboť rozložení na exponent a mantisu nepřináší jenom výhody. Kvůli tomuto rozdělení je ztrátový i jednoduchý převod mezi formátem **int** a **single**, protože dochází k nenávratnému ztracení hodnoty na osmi nejnižších bitech, které slouží pro uložení exponentu. Proto jsou například signálové procesory vyráběny pouze s podporou FX formátu, protože u nich víme, že na vstupu i výstupu jsou prakticky vždy celá čísla. Další nevýhodou, jež vyplývá z komplexnosti formátu s plovoucí řádovou čárkou, je prakticky nemožné využití v tzv. **embedded** zařízeních, jež jsou v dnešní době využívány vice než klasické osobní počítače, neboť jak jsem psal již dříve matematické koprocesory jsou velmi komplikované.

3

Analýza vlivu struktury číslicového filtru a vlivu použitého formátu čísel na vznik zaokrouhlovacích chyb

V následující kapitole se přesunem od roviny teoretické, tedy popisu jednotlivých formátů, k rovině praktické, tedy porovnání vlivu struktury číslicového filtru a vlivu použitého formátu čísel na vznik zaokrouhlovacích chyb.

3.1 Kvantování

Hlavním důvodem vzniku zaokrouhlovacích chyb při číslicovém zpracování signálu je fakt, že při použití filtru navrženého v plné přesnosti, je nutná kvantizace vypočítaných koeficientů. K dispozici však máme jen omezený počet bitů. Tímto dochází k ztrátě přesnosti čísel v jednotlivých formátech. Chybu kvantování můžeme vyjádřit pomocí rozdílu hodnot kvantovaných koeficientů filtru a hodnoty vypočítané v plné přesnosti filtru (rov. 3.1), (rov. 3.2).

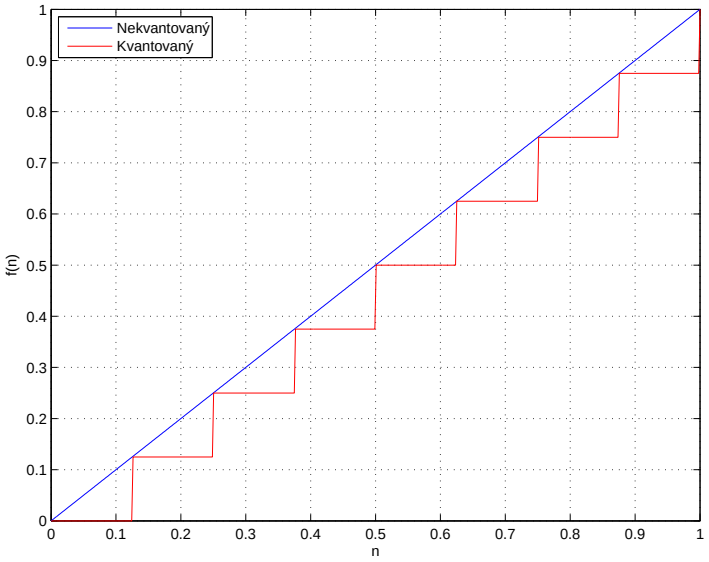
$$\Delta a_i = \acute{a_i} - a_i \qquad [-] \tag{3.1}$$

$$\Delta b_i = \acute{b_i} - b_i \qquad [-] \tag{3.2}$$

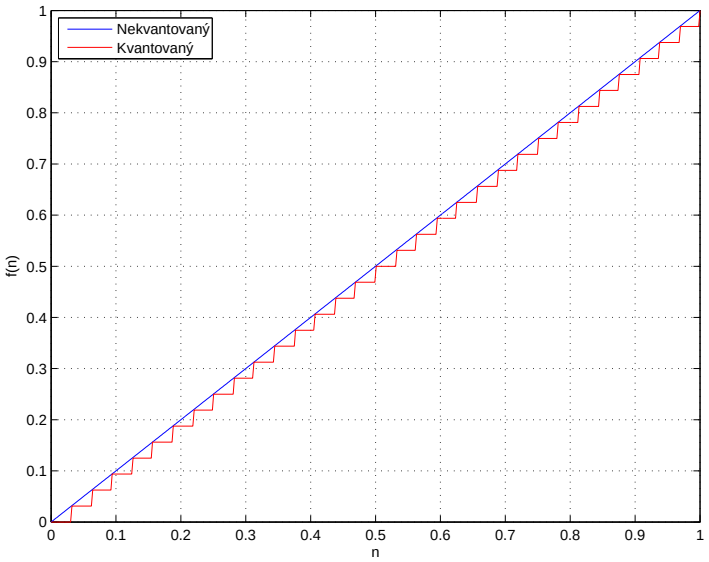
kde:

Δa_i, Δb_i - chyba vzniklá kvantováním
$\acute{a_i}$, $\acute{b_i}$ - kvantované koeficienty filtru
a_i, b_i - koeficienty v plné přesnosti

Pro demonstraci chyb způsobených kvantováním jsem zvolil základní jednorozměrný útvar, přímku. Na tomto útvaru lze nejsnáze a nejnázorněji ukázat jak rozdíl ve volbě formátu čísel, tak rozdíl v počtu bitů, jež máme k dispozici ke kvantování. Pro formát s pevnou řádovou čárkou jsem zvolil formát s délkou zlomkové části 3 bity, (Obr. 3.1) a formát s délkou zlomkové části 5 bitů, (Obr. 3.2). Tyto formáty jsem zvolil záměrně pro jejich názornost a pro zachování totožného počtu bitů na který kvantujeme, jež se shoduje s počtem bitů použitých pro kvantování u následujícího formátu s plovoucí řádovou čárkou. Z grafů je jasně vidět, že čím je zlomková část větší, tím je kvantovací krok menší a tím pádem kvantování jemnější a přesnější.

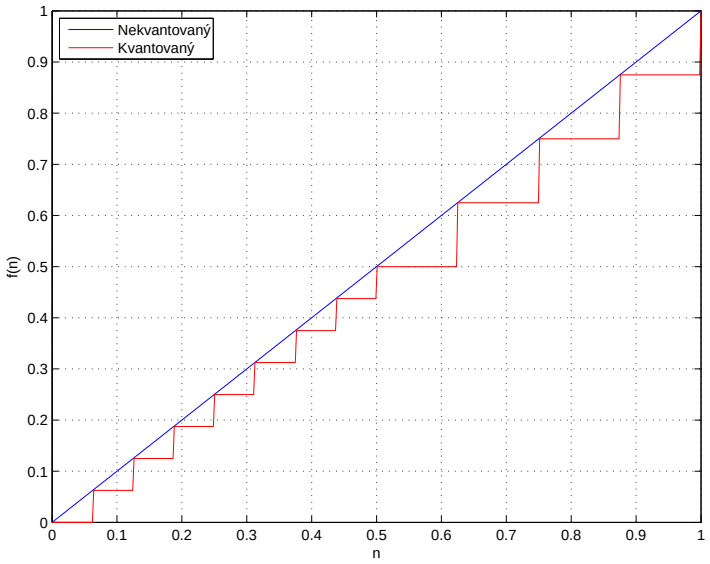


Obr. 3.1: Formát s pevnou řádovou čárkou, délka zlomkové části 3 bity

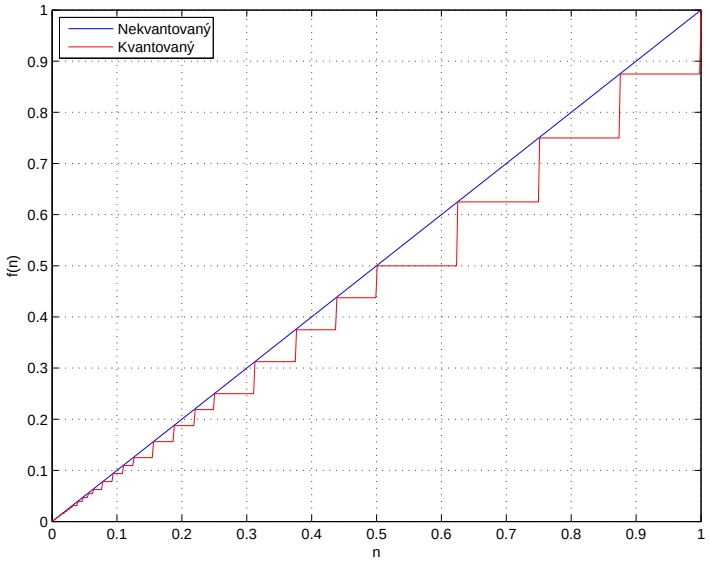


Obr. 3.2: Formát s pevnou řádovou čárkou, délka zlomkové části 5 bitů

Pro druhý formát s plovoucí řádovou čárkou jsem zvolil stejně dlouhé bitové délky jako pro formát předešlý, avšak v tomto se liší délkou exponentů narozdíl od délky zlomkové části, což vyplývá z předešlého textu a popisu základních rozdílů těchto dvou formátů. Testy jsem tedy provedl na formátu s délkou exponentu 3 bity (Obr. 3.4) a formátu s délkou exponentu 5 bitů (Obr. 3.4). I zde názorně vidíme, že použití více-bitového formátu je přesnější. Pro větší hodnoty je kvantování hrubší, což je dáno z podstaty formátu s plovoucí řádovou čárkou a jeho rozdělením na mantisu a exponent.



Obr. 3.3: Formát s plovoucí řádovou čárkou, délka exponentu 3 bity



Obr. 3.4: Formát s pevnou řádovou čárkou, délka exponentu 5 bitů

3.2 Číslicové filtry

Pro pochopení funkce číslicových filtrů se nejdříve pokusím vysvětlit pár nezbytných pojmů. Lineárně časově invariantní systémy jsou základem práce s číslicovými filtry. Časová invariantnost znamená, že systém odpovídá stále stejným výstupním signálem y(t) na určitý vstupní signál x(t). Pokud budíme vstup systému signálem x(t) posunutým v čase x(t-t_0), potom systém odpoví odezvou y(t) stejně posunutou v čase y(t-t_0) (rov. 3.3).

$$a_0y(t) - a_1y(t-1) - a_2y(t-2) - ... = b_0x(t) + b_1x(t-1) + b_2x(t-2) + ... \quad (3.3)$$

kde:

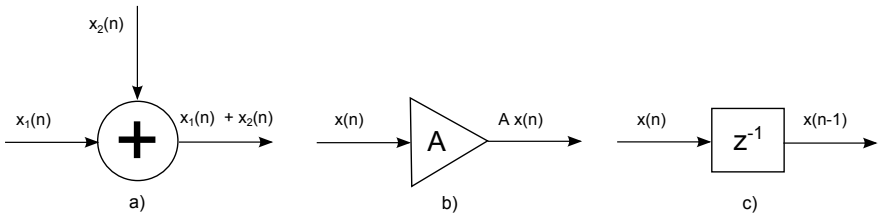
a_0, a_1, a_2, a_m, b_0, b_1, b_2, b_n - koeficienty filtru

Lineární systém je pak takový, který na násobky vstupního signálu x(t), odpovídá násobky výstupního signálu k y(t) a na sumu vstupních signálů odpovídá sumou signálů výstupních. Hodnoty výstupu realizované hodnotami v dřívějších okamžicích pak popisuje rekurentní tvar rovnice (rov. 3.4). Z tohoto tvaru rovnice pak Z transformací získáme obecný tvar přenosové funkce číslicového filtru (rov. 3.5).

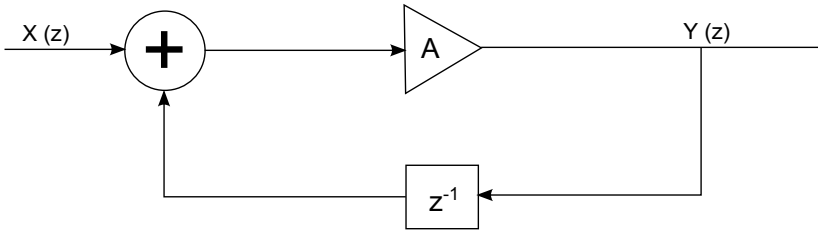
$$y(n) = \sum_{i=0}^{m} b_i x(n-i) + \sum_{i=1}^{n} a_i y(n-i) \quad (3.4)$$

$$\begin{aligned} Y(z) &= \sum_{i=0}^{m} b_i X(z) z^{-i} + \sum_{i=1}^{n} a_i Y(z) z^{-i} \\ Y(z)(1 - \sum_{i=1}^{n} a_i z^{-i}) &= X(z) \sum_{i=0}^{m} b_i z^{-i} \\ H(z) &= \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{m} b_i z^{-i}}{1 - \sum_{i=1}^{n} a_i z^{-i}} \end{aligned} \quad (3.5)$$

Přenosovou funkci pak lze nalézt Z-transformací diferenční rovnice. Lineární časově invariantní systémy lze popsat blokovými schématy. Pro názornost jsem použil jednoduché schéma s třemi základními bloky (Obr. 3.5). Rozdíl mezi časovou (Obr. **??**) a transformovanou (Obr. 3.6) oblastí je znázorněn níže.



Obr. 3.5: Základní bloky filtru: a) sčítačka, b) násobička konstantou, c) jedntokové zpoždění



Obr. 3.6: Blokové schéma systému v transformované oblasti

3.3 Návrh číslicových filtrů

Návrh číslicového filtru jako takového tvoří základ pro klasické číslicové zpracování signálu. Realizace číslicového filtru je možná buďto speciálním obvodem, nebo pomocí výpočetních programů pro stolní počítače. Velkou výhodou číslicových filtrů oproti analogovým je vysoká přesnost a absence driftu, který je zapříčiněn vlivem změny součástek. Dále je zde i značně snadnější návrh a následná simulace filtru. Na rozdíl od analogových filtrů je však zcela nevhodný pro vysokofrekvenční signály. Číslicové filtry dělíme:

1. podle délky impulsní odezvy na:
    - filtry s konečnou impulsní odezvou **(FIR)**
    - filtry s nekonečnou impulsní odezvou **(IIR)**
2. podle struktury blokového schématu na:
    - filtry, jež nemají zpětnou vazbu, tzv. nerekurzivní
    - filtry, jež mají zpětnou vazbu, tzv. rekurzivní

Filtry s konečnou impulsní odezvou (FIR)

Velká devíza filtru s konečnou odezvou je jeho stabilita za všech podmínek. Volíme ho především tam, kde je potřeba lineární fázové charakteristiky v propustném pásmu, avšak pokud tuto vlastnost přímo nevyžadujeme je lepší volba filtru typu IIR, protože pro stejný útlum v nepropustném pásmu mu stačí menší počet koeficientů. Filtr typu FIR je tedy za takových podmínek neúsporný a celkově nevhodný.

Filtry s nekonečnou impulsní odezvou (IIR)

Tomuto typu filtru se budu věnovat více dopodrobna, neboť tento typ používám i v analýze vlivu struktury na zaokrouhlovací chyby. Filtr s nekonečnou impulsní odezvou vyžaduje vždy alespoň jednu zpětnovazební smyčku, neboli má vždy rekurzivní strukturu, která bude blíže popsána v typech jednotlivých struktur. Oproti FIR filtru je zde nutné kontrolovat stabilitu, tedy polohu pólů přenosu vůči jednotkové kružnici. Další nespornou výhodou je realizace zadání za použití nižšího řádu filtru, to způsobuje aproximace požadované charakteristiky racionální lomenou funkcí. Odtud plyne menší zpoždění signálu v rámci filtrace. Řád filtru určuje vyšší ze stupňů polynomů, jimiž je reprezentována přenosová funkce analogového prototypu filtru. Fázová charakteristika není nikdy lineární.

Nejpoužívanější metodou návrhu je návrh pomocí analogového vzorového filtru. Pokud vyjdeme z tabulek koeficientů analogového filtru a zobrazení roviny P do roviny Z. Pak najdeme příslušné koeficienty číslicového filtru. Levá polorovina P pak musí odpovídat alespoň části vnitřku jednotkové kružnice v rovině Z pro zachování stability filtru. Nejpoužívanějšími metodami pro výše zmíněnou transformaci jsou *metoda impulsní invariance, metoda náhrady derivací diferencemi a metoda bilineární transformace*, která je nejpoužívanější pro její vyloučení aliasingu. Kvůli nelineární transformaci z analogové do číslicové části je potřeba v rámci návrhu přepočítat mezní frekvenci filtru. Hlavními navrhovanými filtry typu IIR jsou pak základní kmitočtově výběrové filtry a to zejména dolní propust, ze které návrhy většinou vycházejí. Pro filtry s více propustnými či nepropustnými pásmy se používá metoda přímého návrhu, při které dochází k aproximaci požadované amplitudové frekvenční charakteristiky číslicovým filtrem za pomocí přímkových úseků.

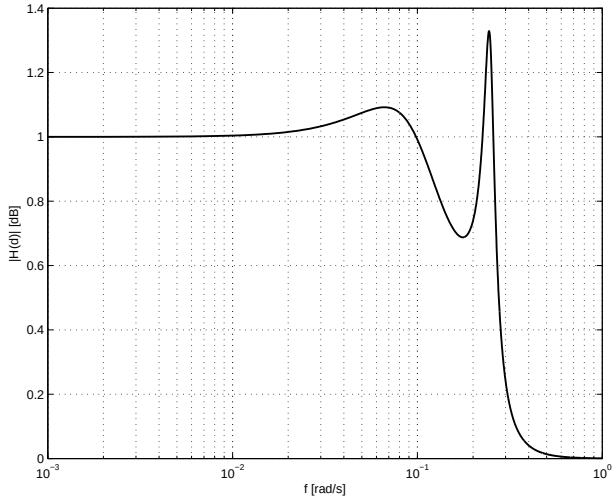
3.4 Filtr čtvrtého řádu, typu dolní propust

V tomto příkladu shrnu všechny dříve popsané skutečnosti a na jejich základě provedu analýzu pro filtr typu dolní propust. Strukturální vliv ověříme na čtyřech typech struktur. A to pro přímou, kaskádní, paralelní a lattice-ladder strukturu. Formáty čísel budou reprezentovat mnou dříve představené dva typy, formát s pevnou a plovoucí řádovou čárkou.

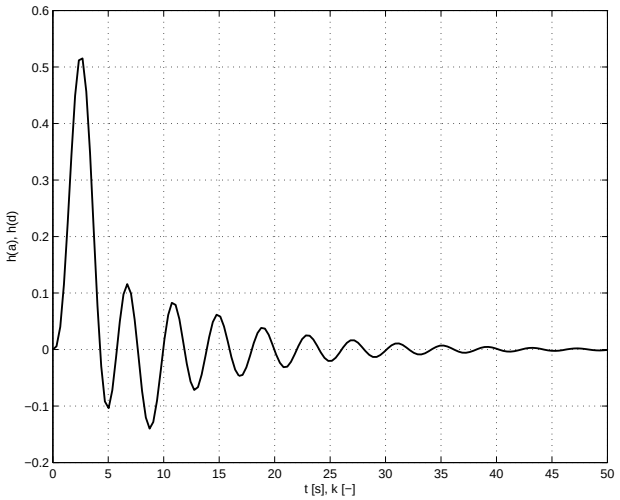
Pro svoji analýzu jsem zvolil filtr čtvrtého řádu, analogový prototyp je reprezentovaný přenosovou funkcí (rov. 3.6)

$$F_{(s)} = \frac{1}{s^4 + s^3 + 3s^2 + 2s + 1} \tag{3.6}$$

Frekvenční a impulsní charakteristiku tohoto filtru znázorňují obrázky (Obr. 3.7), (Obr. 3.8).



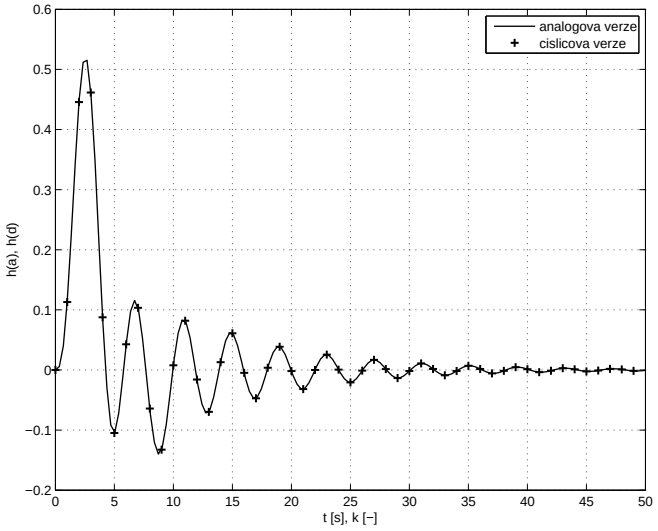
Obr. 3.7: Frekvenční charakteristika analogového prototypu filtru



Obr. 3.8: Impulsní charakteristika analogového prototypu filtru

Pro vzorkování analogového filtru jsem zvolil periodu rovnou jedné sekundě (rov. 3.7). Z obrázku (Obr. 3.9) je patrné, že impulsní charakteristika v plné přesnosti získaná impulsní invariancí a impulsní charakteristika analogového prototypu filtru se shodují.

$$T_{(s)} = 1s \tag{3.7}$$



Obr. 3.9: Impulsní charakteristika analogového prototypu filtru v porovnání s jeho diskrétní verzí

Diskrétní verzi filtru v plné přesnosti získanou impulsní invariancí popisuje přenosová funkce (rov. 3.8).

$$\begin{aligned} H(z) &= \tfrac{Y(z)}{X(z)} \\ Y(z) &= 1,1308\text{e-}1z^3 + 3,0882\text{e-}1z^2 + 6,9332\text{e-}2z \\ X(z) &= 1z^4 - 1,2108z^3 + 1,3033z^2 - 9,6992\text{e-}1z + 3,6788\text{e-}1 \end{aligned} \tag{3.8}$$

Pro porovnávání vlivu chyby vzniklé kvantováním jsem zvolil energii chybového signálu. Tuto energie získáme jako sumu kvadrátů rozdílu impulsní odezvy v plné přesnosti a přesnosti kvantované na určitý počet bitů. K výpočtu použijeme prvních 100 členů impulsní odezvy výše uvedeného filtru (rov. 3.9). Porovnání energií je pak provedeno pro všechny 4 struktury a pro oba formáty čísel.

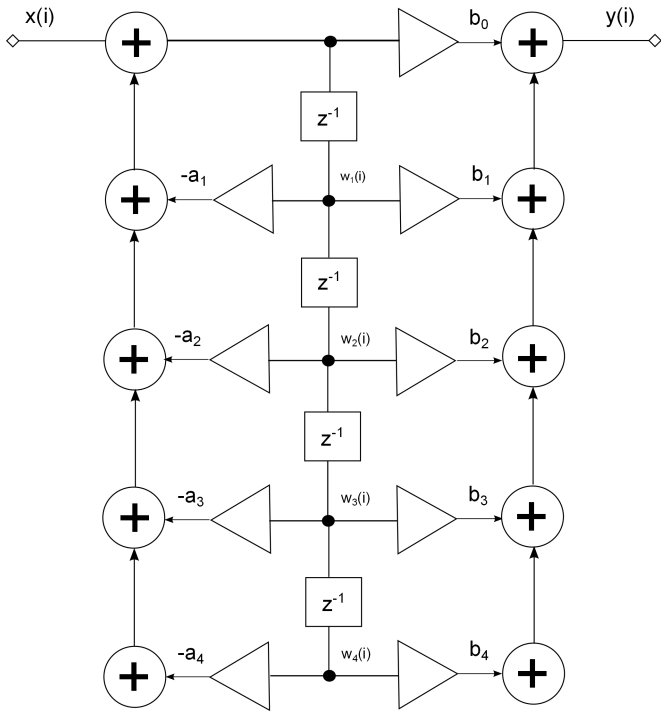
$$e(b) = \sum_{i=0}^{100} [h(i) - h_b(i)]^2 \qquad [-] \tag{3.9}$$

kde:

e - energie chybového signálu
$h(i)$ - v i-tý vzorek impulsní odezvy číslicového filtru v plné přesnosti
$h_b(i)$ - i-tý vzorek impulsní odezvy číslicového filtru v přesnosti omezené na b bitů
b - počet bitů, na které je přesnost omezena

3.4.1 Přímá struktura

Realizace této struktury (Obr. 3.10) je nejjednodušší, jelikož její parametrizace přímo vychází z přenosové funkce analogového filtru. Jako všechny mnou použité struktury, je i tato rekurzivní, což znamená, že obsahuje alespoň jednu zpětnovazební větev. Důsledek zpětné vazby je teoreticky neomezená délka impulsní odezvy. Prakticky však hodnoty získané postupnou iterací diferenční rovnice limitují k nule. Největší nevýhodou přímých struktur je velká citlivost frekvenční charakteristiky na přesnost koeficientů a_i, b_i. Zejména pak při velkém počtu koeficientů a_i, b_i dochází kvantizací ke změně polohy pólů a nul oproti vypočteným hodnotám. Tyto vlivy lze zmenšit postupným dělením systémové funkce na menší a jednodušší systémové funkce. Každá z takto nově vzniklých funkcí má pak jen malý počet pólů a nul. Spojením těchto menších funkcí pak realizujeme paralelní nebo kaskádní strukturou.



Obr. 3.10: Filtr realizován přímou strukturou

Parametrizaci filtru, který je realizován přímou strukturou je následující (rov. 3.10)

$$\begin{array}{ll} & b_0 = 0 \\ a_1 = -1,2108 & b_1 = 1,1308\text{e-}1 \\ a_2 = 1,3033 & b_2 = 3,0882\text{e-}1 \\ a_3 = -9,6992\text{e-}1 & b_3 = 6,9332\text{e-}2 \\ a_4 = 3,6788\text{e-}1 & b_4 = 0 \end{array} \tag{3.10}$$

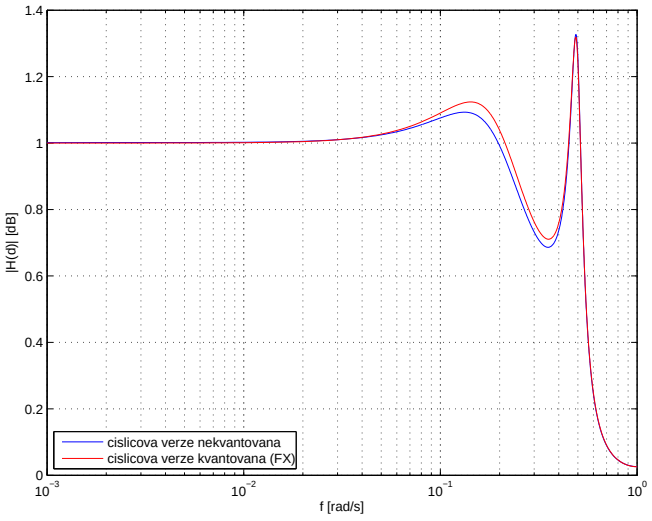
Struktura matic stavové reprezentace filtru realizovaného přímou strukturou je popsána rovnicemi (rov. 3.11), (rov. 3.12).

$$\begin{bmatrix} w_1(i+1) \\ w_2(i+1) \\ w_3(i+1) \\ w_4(i+1) \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & -a_3 & -a_4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1(i) \\ w_2(i) \\ w_3(i) \\ w_4(i) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} x(i) \tag{3.11}$$

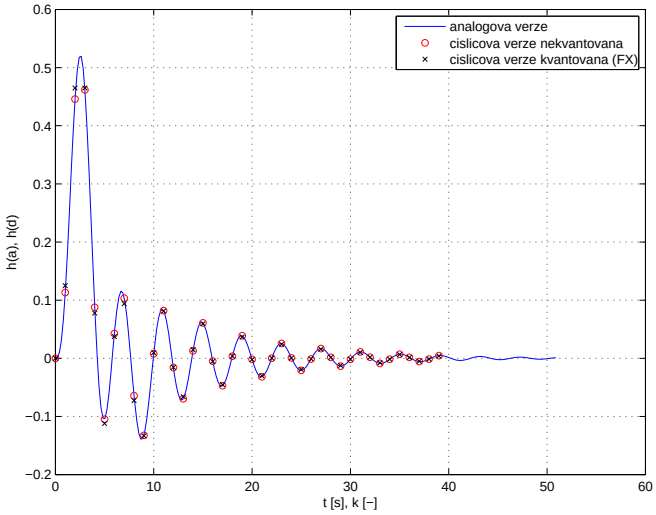
$$y(i) = \begin{bmatrix} -b_0a_1 + b_1 & -b_0a_2 + b_2 & -b_0a_3 + b_3 & -b_0a_4 + b_4 \end{bmatrix} \begin{bmatrix} w_1(i) \\ w_2(i) \\ w_3(i) \\ w_4(i) \end{bmatrix} + \begin{bmatrix} b_0 \end{bmatrix} x(i) \tag{3.12}$$

Formát s pevnou řádovou čárkou

Jako první u této struktury zhodnotím formát s pevnou řádovou čárkou. Kvantování u tohoto formátu je provedeno na 16 bitů stejně tak jako u formátu s plovoucí řádovou čárkou. Jak vidíme z obrázků (Obr. 3.11), (Obr. 3.12) není frekvenční ani impulsní charakteristika věrnou kopií charakteristik analogové verze filtru. To je zapříčiněno tím, že dochází k přetečení a následné saturaci. Proto dosáhnou všechny koeficienty filtru v tomto zobrazení maximálních hodnot.

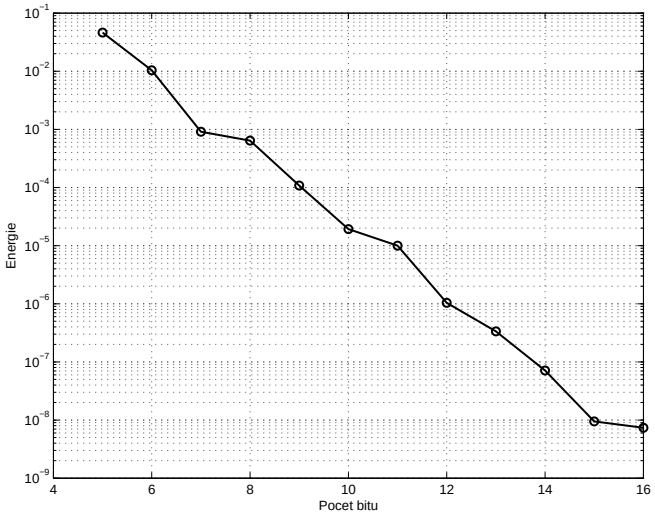


Obr. 3.11: Frekvenční charakteristika filtru realizovaná přímou strukturou, porovnání kvantované verze ve formátu s pevnou řádovou čárkou (16ti bitový formát, délka celočíselné části - 2 bity) a verze nekvantované



Obr. 3.12: Impulsní charakteristika filtru realizovaná přímou strukturou, porovnání kvantované verze ve formátu s pevnou řádovou čárkou (16ti bitový formát, délka celočíselné části - 2 bity), verze nekvantované a analogové verze filtru

Další graf (Obr. 3.13) znázorňuje změny energie chybového signálu v závislosti na počtu bitů, na něž je kvantován filtr.

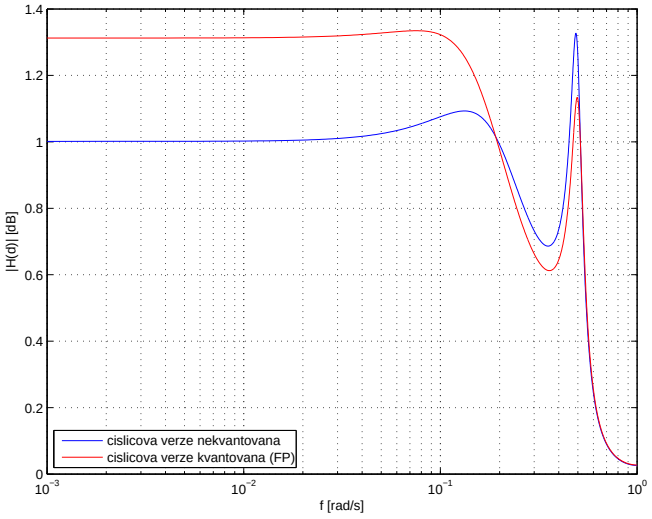


Obr. 3.13: Změny energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Fixed Point - 16ti bitový formát, délka celočíselné části - 2 bity)

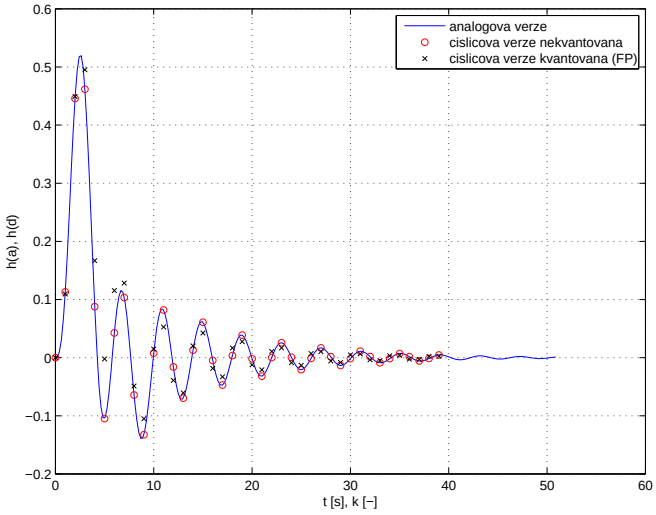
Z obrázku (Obr. 3.13) je patrné, že čím více bitu ke kvantování použijeme, tím je energie chybového signálu menší. Pro prvních několik bitů jsou zde velké rozdíly, avšak je vidno z obrázku, že již od kvantování na 9 a více bitů jsou energie skoro neměnné.

Formát s plovoucí řádovou čárkou

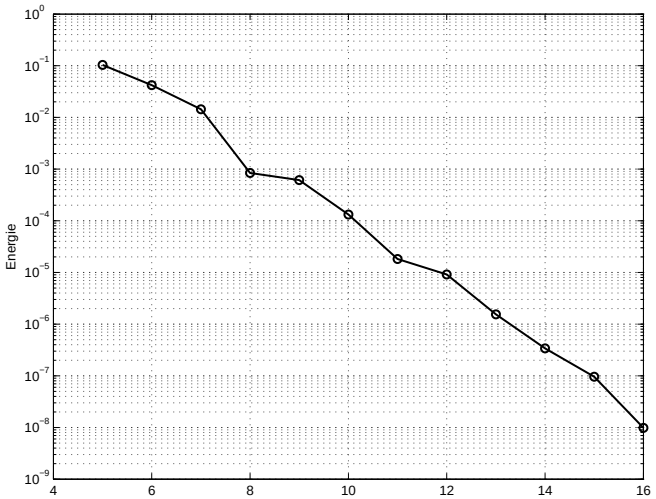
Totéž jsem provedl pro formát s plovoucí řádovou čárkou. Z grafů (Obr. 3.14), (Obr. 3.15) můžeme pozorovat, že u tohoto formátu jsou výsledky horší než u formátu předešlého. To jasně potvrzuje i obrázek (Obr. 3.16), který dokazuje, že energie chybového signálu pro stejný počet bitů na které kvantujeme, je u tohoto formátu větší. Z grafu vyplývá, že energie chybového signálu při kvantování na 5 bitů u formátu s pevnou řádovou čárkou se přibližně rovná energii chybového signálu při kvantování na 6 bitů u formátu s plovoucí řádovou čárkou. Tím můžeme prohlásit, že formát s pevnou řádovou čárkou je u této struktury vhodnější.



Obr. 3.14: Frekvenční charakteristika filtru realizovaná přímou strukturou, porovnání kvantované verze ve formátu s plovoucí řádovou čárkou (16ti bitový formát, délka exponentu - 2 bity) a verze nekvantované



Obr. 3.15: Impulsní charakteristika filtru realizovaná přímou strukturou, porovnání kvantované verze ve formátu s plovoucí řádovou čárkou (16ti bitový formát, délka exponentu - 2 bity), verze nekvantované a analogové verze filtru

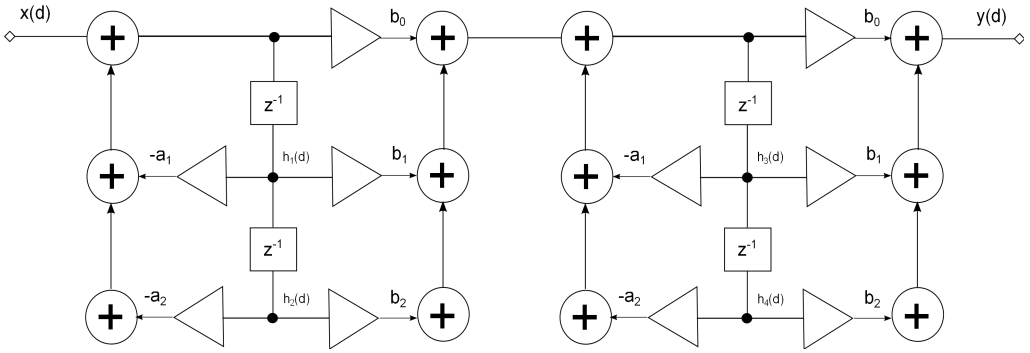


Obr. 3.16: Změny energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Floating Point - 16ti bitový formát, délka exponentu - 2 bity)

3.4.2 Kaskádní struktura

Kaskádní struktura je charakterizována jako součin jednodušších, dílčích systémových funkcí (rov. 3.13). Jako předešlá struktura i tato obsahuje alespoň jednu zpětnovazební větev což je názorně vidět ze schématu této struktury (Obr. 3.17).

$$H(z) = \frac{Y(z)}{X(z)} = H_1(z)H_2(z)\dots H_n(z) \qquad [-] \tag{3.13}$$



Obr. 3.17: Filtr realizován kaskádní strukturou

Parametrizaci filtru, který je realizován kaskádní strukturou vyjadřuje rovnice (rov. 3.14)

$$\begin{array}{ll} & b_0 = -1,1308\text{e-}1 \\ a_1 = -1,1778 & b_1 = 2,8091\text{e-}1 \\ a_2 = 4,5373\text{e-}1 & b_2 = 0 \\ a_3 = -3,2962\text{e-}1 & b_3 = 0 \\ a_4 = 8,1078\text{e-}1 & b_4 = 1 \\ & b_5 = 2,4682\text{e-}1 \end{array} \tag{3.14}$$

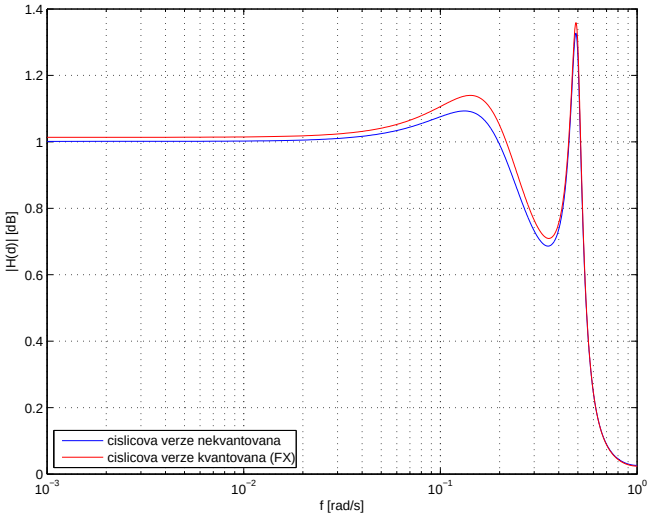
Struktura matic stavové reprezentace filtru realizovaného kaskádní strukturou popisují matice (rov. 3.15), (rov. 3.16).

$$\begin{bmatrix} w_1(i+1) \\ w_2(i+1) \\ w_3(i+1) \\ w_4(i+1) \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -b_0a_1+b_1 & -b_0a_2+b_2 & -a_3 & -a_4 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1(i) \\ w_2(i) \\ w_3(i) \\ w_4(i) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ b_0 \\ 0 \end{bmatrix} x(i) \tag{3.15}$$

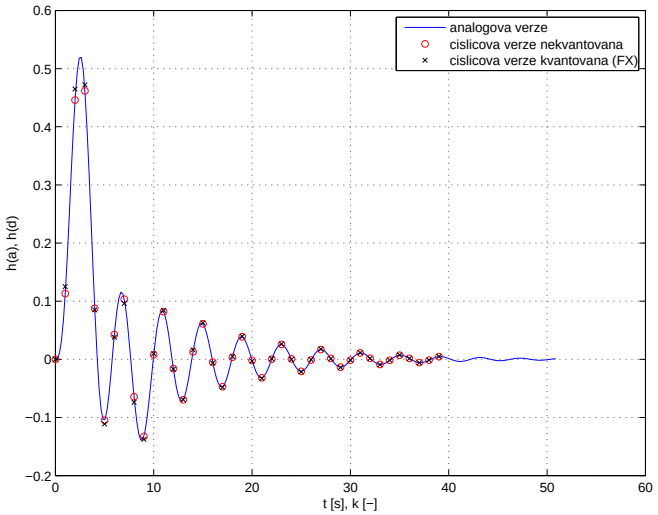
$$y(i) = \begin{bmatrix} b_0b_3a_1+b_1b_3 & b_0b_3a_2+b_2b_3 & b_3a_3+b_4 & b_3a_4+b_5 \end{bmatrix} \begin{bmatrix} w_1(i) \\ w_2(i) \\ w_3(i) \\ w_4(i) \end{bmatrix} + \begin{bmatrix} b_0b_3 \end{bmatrix} x(i) \tag{3.16}$$

Formát s pevnou řádovou čárkou

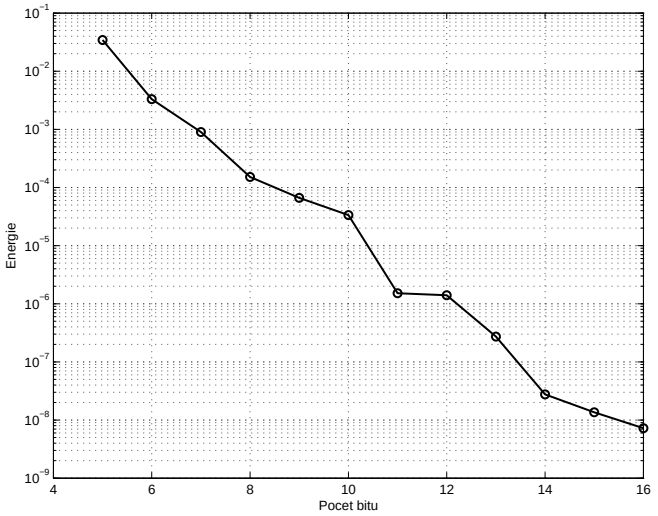
Stejně jako u struktury přímé i zde budu porovnávat kvantované verze filtru s nekvantovanými, u frekvenčních a impulsních charakteristik filtru (Obr. 3.18), (Obr. 3.19). Tato struktura dosahuje lepších výsledků, než struktura přímá, což dokumentuje i graf změn energie chybového signálu v závislosti na počet bitů na něž je kvantován (Obr. 3.20). I zde pozorujeme, že se energie od určitého počtu bitů na něž kvantujeme již nemění, ale tento jev nastává o jeden bit dříve, než u struktury přímé. Při kvantování na 6 bitů, je energie chybového signálu poloviční než tomu bylo u struktury přímé. To je zapříčiněno právě rozložením systémové funkce na součin jednodušších, dílčích systémových funkcí.



Obr. 3.18: Frekvenční charakteristika filtru realizovaná kaskádní strukturou, porovnání kvantované verze ve formátu s pevnou řádovou čárkou (16ti bitový formát, délka celočíselné části - 2 bity) a verze nekvantované



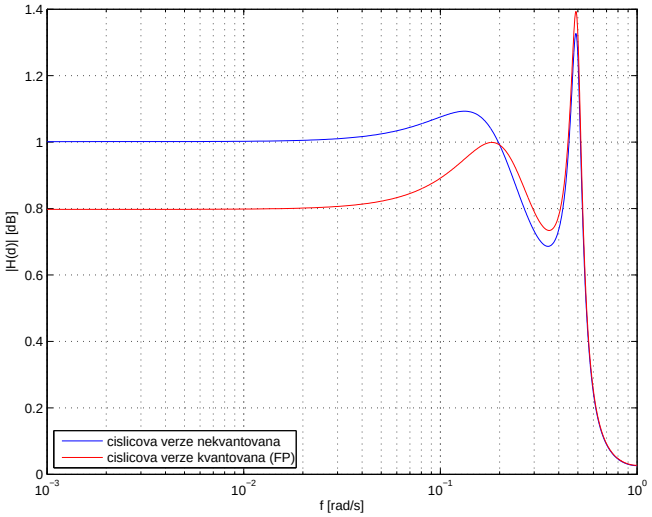
Obr. 3.19: Impulsní charakteristika filtru realizovaná kaskádní strukturou, porovnání kvantované verze ve formátu s pevnou řádovou čárkou (16ti bitový formát, délka celočíselné části - 2 bity), verze nekvantované a analogové verze filtru



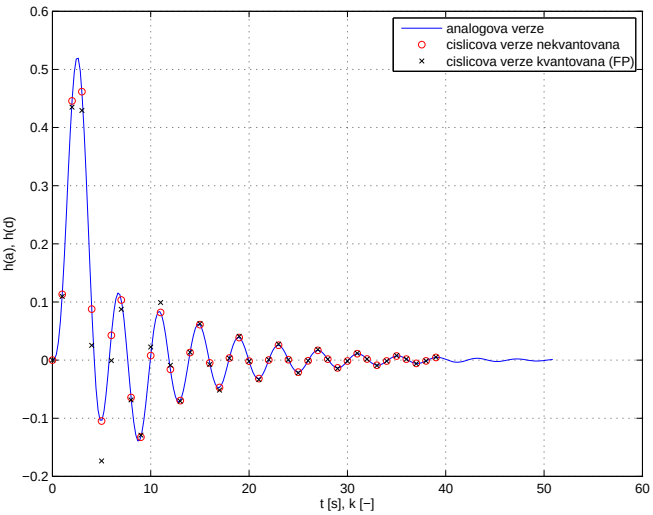
Obr. 3.20: Změny energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Fixed Point - 16ti bitový formát, délka celočíselné části - 2 bity)

Formát s plovoucí řádovou čárkou

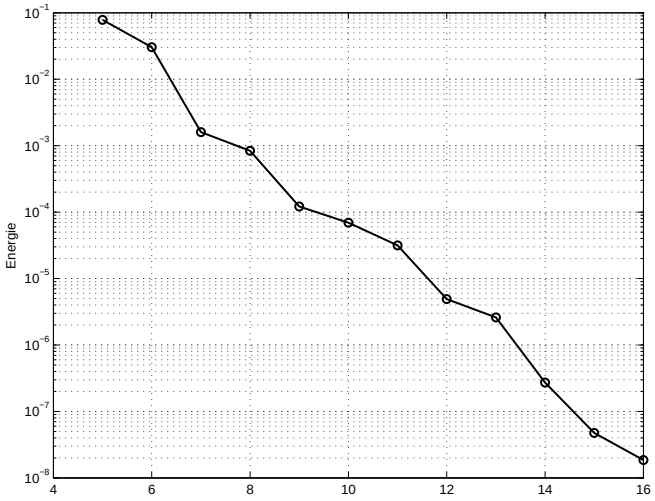
I u této struktury se formát s plovoucí řádovou čárkou jeví jako ten horší, což můžeme vypozorovat z frekvenční a impulsní charakteristiky (Obr. 3.21), (Obr. 3.22). V porovnání se strukturou přímou však vykazuje zlepšení, tedy energie chybového signálu je pro jakýkoliv počet bitů na něž je kvantováno nižší (Obr. 3.23). Ustálení změny energie chybového signálu, je i zde od kvantování na 8 bitů



Obr. 3.21: Frekvenční charakteristika filtru realizovaná kaskádní strukturou, porovnání kvantované verze ve formátu s plovoucí řádovou čárkou (16ti bitový formát, délka exponentu - 2 bity) a verze nekvantované



Obr. 3.22: Impulsní charakteristika filtru realizovaná kaskádní strukturou, porovnání kvantované verze ve formátu s plovoucí řádovou čárkou (16ti bitový formát, délka exponentu - 2 bity), verze nekvantované a analogové verze filtru

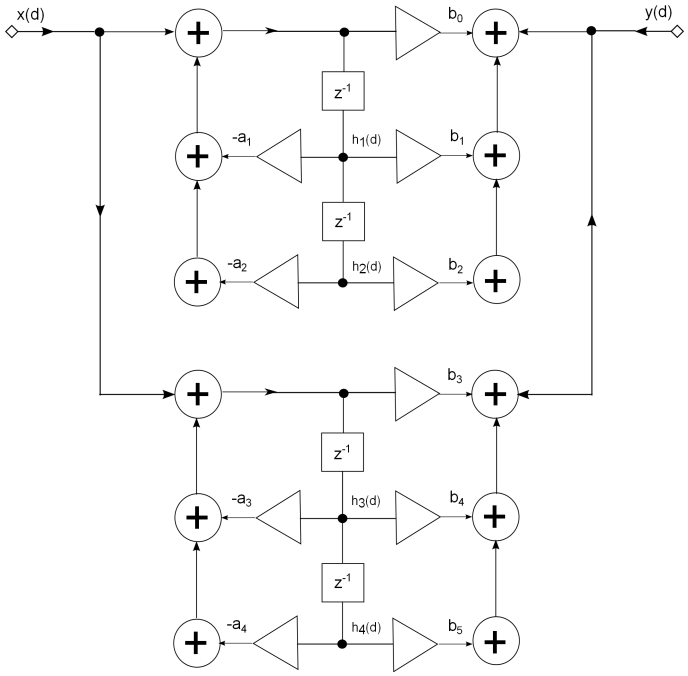


Obr. 3.23: Změny energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Floating Point - 16ti bitový formát, délka exponentu - 2 bity)

3.4.3 Paralelní struktura

Tak jako kaskádní struktura tak i paralelní je charakterizována jednoduššími, dílčími systémovými funkcemi, avšak ne jejich součinem ale součtem (rov. 3.17). I tato struktura stejně jako předešlé patří mezi rekurentní struktury (Obr. 3.24).

$$H(z) = \frac{Y(z)}{X(z)} = H_1(z) + H_1(z) + H_2 + (z) \ldots + H_n(z) \qquad [-] \tag{3.17}$$



Obr. 3.24: Filtr realizován paralelní strukturou

Parametrizaci filtru, který je realizován paralelní strukturou charakterizuje rovnice (rov. 3.18).

$$\begin{array}{ll} & b_0 = -1,1397\text{e-}1 \\ a_1 = -3,2962\text{e-}1 & b_1 = 2,3370\text{e-}1 \\ a_2 = 8,1078\text{e-}1 & b_2 = 0 \\ a_3 = -1,778 & b_3 = 1,1397\text{e-}1 \\ a_4 = 4,5373\text{e-}1 & b_4 = -2,1629\text{e-}1 \\ & b_5 = 0 \end{array} \tag{3.18}$$

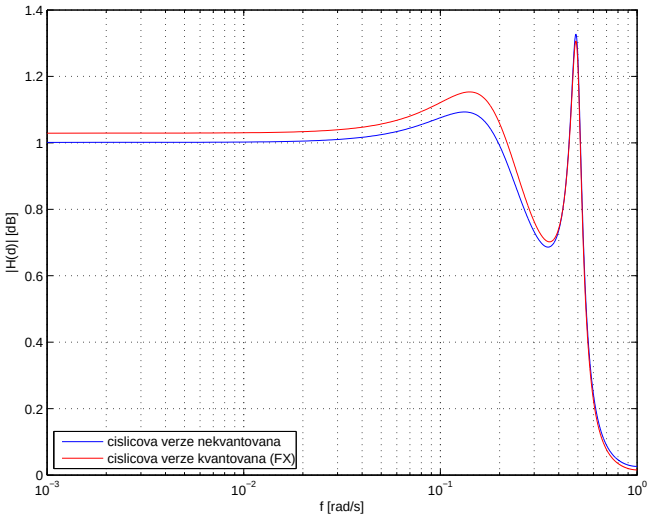
Matice, které popisuje struktura stavové reprezentace filtru realizovaného paralelní strukturou jsou pak následující (rov. 3.19), (rov. 3.20).

$$\begin{bmatrix} w_1(i+1) \\ w_2(i+1) \\ w_3(i+1) \\ w_4(i+1) \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -a_3 & -a_4 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1(i) \\ w_2(i) \\ w_3(i) \\ w_4(i) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} x(i) \tag{3.19}$$

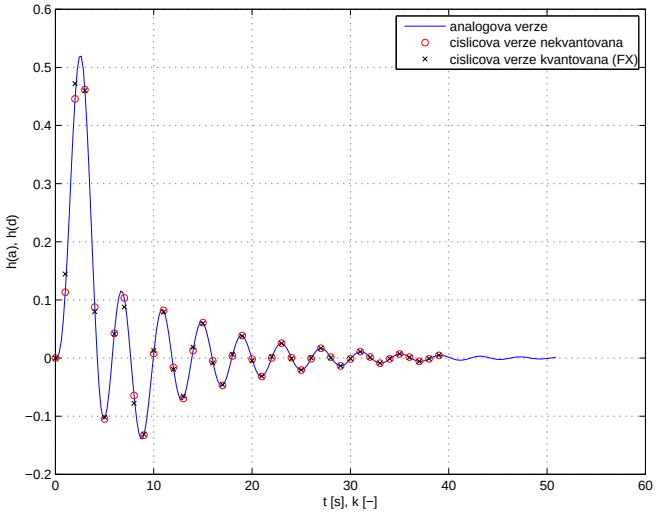
$$y(i) = -\begin{bmatrix} b_0a_1 + b_1 & b_0a_2 + b_2 & b_3a_3 + b_4 & b_3a_4 + b_5 \end{bmatrix} \begin{bmatrix} w_1(i) \\ w_2(i) \\ w_3(i) \\ w_4(i) \end{bmatrix} + \begin{bmatrix} b_0 \end{bmatrix} x(i) \tag{3.20}$$

Formát s pevnou řádovou čárkou

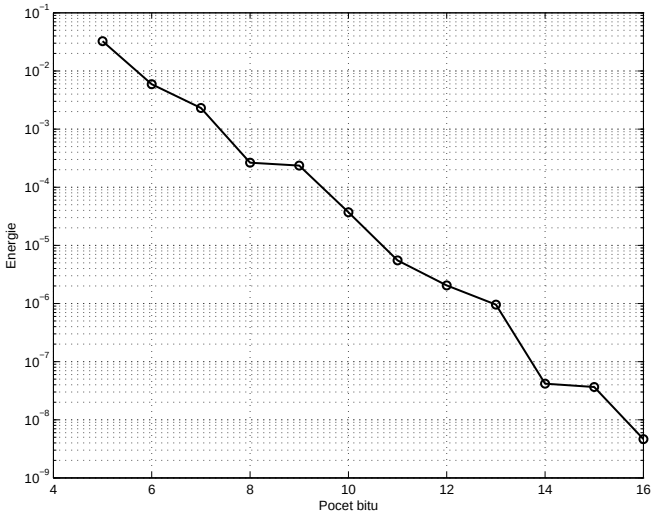
Tato struktura dosahuje zdánlivě nejlepších výsledků podle grafu změny energie chybového signálu (Obr. 3.27). Avšak při bližším prozkoumání zjistíme, že při kvantování na 5 bitů sice vykazuje nejmenší chybovost, avšak při zvyšování počtu bitů na něž kvantujeme, energie neklesá tak strmě jako u ostatních struktur a při kvantování na 9 bitů již dosahuje výsledku nejhorších ze všech struktur. To je dáno právě sčítáním dílčích systémových funkcí.



Obr. 3.25: Frekvenční charakteristika filtru realizovaná paralelní strukturou, porovnání kvantované verze ve formátu s pevnou řádovou čárkou (16ti bitový formát, délka celočíselné části - 2 bity) a verze nekvantované



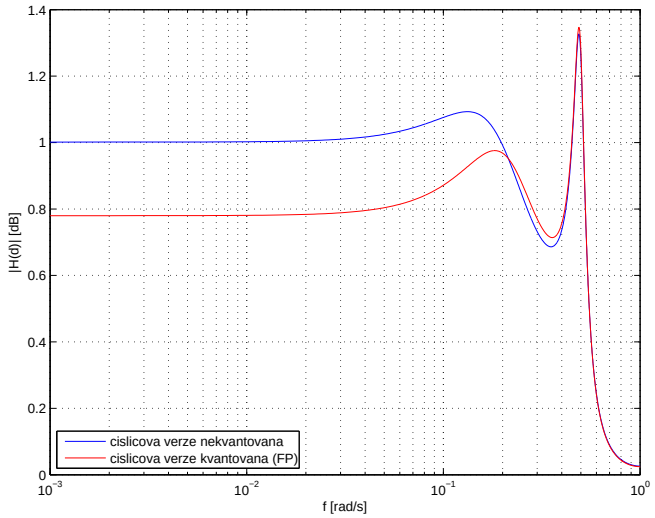
Obr. 3.26: Impulsní charakteristika filtru realizovaná paralelní strukturou, porovnání kvantované verze ve formátu s pevnou řádovou čárkou (16ti bitový formát, délka celočíselné části - 2 bity), verze nekvantované a analogové verze filtru



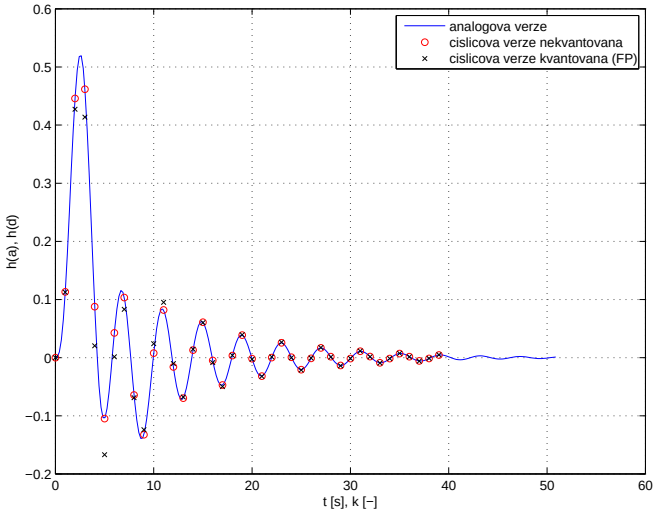
Obr. 3.27: Změny energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Fixed Point - 16ti bitový formát, délka celočíselné části - 2 bity)

Formát s plovoucí řádovou čárkou

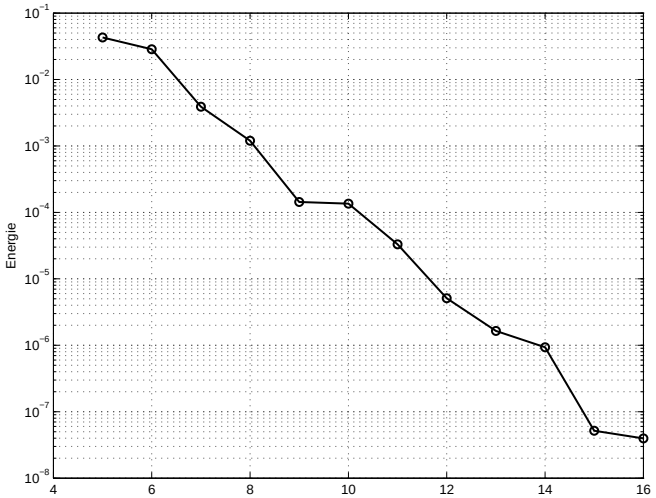
Ani u této struktury nevykazuje formát s plovoucí řádovou čárkou lepších vysledků než ten s pevnou. Co však stojí za povšimnutí je, že při kvantování na 5 bitů se energie chybového signálu téměř rovná hodnotě, kterou dosáhl formát s pevnou řádovou čárkou u přímé struktury (Obr. 3.30). To naznačuje výhodnost této struktury při kvantování na malý počet bitů.



Obr. 3.28: Frekvenční charakteristika filtru realizovaná paralelní strukturou, porovnání kvantované verze ve formátu s plovoucí řádovou čárkou (16ti bitový formát, délka exponentu - 2 bity) a verze nekvantované



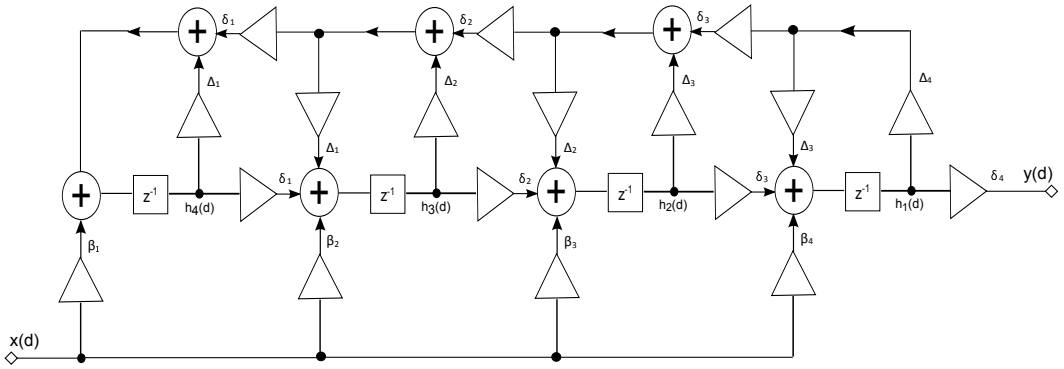
Obr. 3.29: Impulsní charakteristika filtru realizovaná paralelní strukturou, porovnání kvantované verze ve formátu s plovoucí řádovou čárkou (16ti bitový formát, délka exponentu - 2 bity), verze nekvantované a analogové verze filtru



Obr. 3.30: Změny energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Floating Point - 16ti bitový formát, délka exponentu - 2 bity)

3.4.4 Lattice-ladder struktura

Poslední strukturou jejíž vliv budu ověřovat, je Lattice-ladder struktura (Obr. 3.31). Tato struktura má obtížnější parametrizaci než ostatní (rov. 3.21). Stavová reprezentace filtru, pak poskytuje informace o struktuře jednotlivých matic (rov. 3.22), (rov. 3.23).



Obr. 3.31: Filtr realizován lattice-ladder strukturou

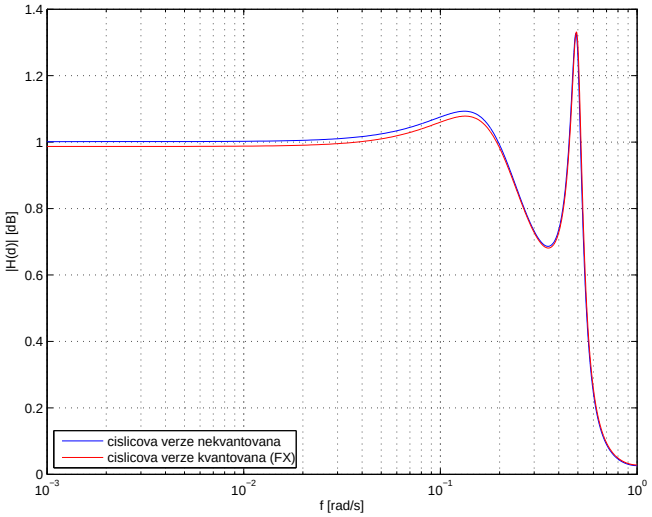
$$\begin{array}{lll} \Delta_1 = 4,1558\text{e-}1 & \delta_1 = \sqrt{1-\Delta_1^2} & \beta_1 = 0 \\ \Delta_2 = -5,5965\text{e-}1 & \delta_2 = \sqrt{1-\Delta_2^2} & \beta_2 = 1,5296\text{e-}1 \\ \Delta_3 = 6,0659\text{e-}1 & \delta_3 = \sqrt{1-\Delta_3^2} & \beta_3 = 6,2370\text{e-}1 \\ \Delta_4 = -3,6788\text{e-}1 & \delta_4 = \sqrt{1-\Delta_4^2} & \beta_4 = 2,9528\text{e-}1 \end{array} \tag{3.21}$$

$$\begin{bmatrix} w_1(i+1) \\ w_2(i+1) \\ w_3(i+1) \\ w_4(i+1) \end{bmatrix} = \begin{bmatrix} -\Delta_3\Delta_4 & \delta_3 & 0 & 0 \\ -\Delta_2\delta_3\Delta_4 & -\Delta_2\Delta_3 & \delta_2 & 0 \\ -\Delta_1\delta_2\delta_3\Delta_4 & -\Delta_1\delta_2\Delta_3 & -\Delta_1\Delta_2 & \delta_1 \\ \delta_1\delta_2\delta_3\Delta_4 & \delta_1\delta_2\Delta_3 & \delta_1\Delta_2 & \Delta_1 \end{bmatrix} \begin{bmatrix} w_1(i) \\ w_2(i) \\ w_3(i) \\ w_4(i) \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} x(i) \tag{3.22}$$

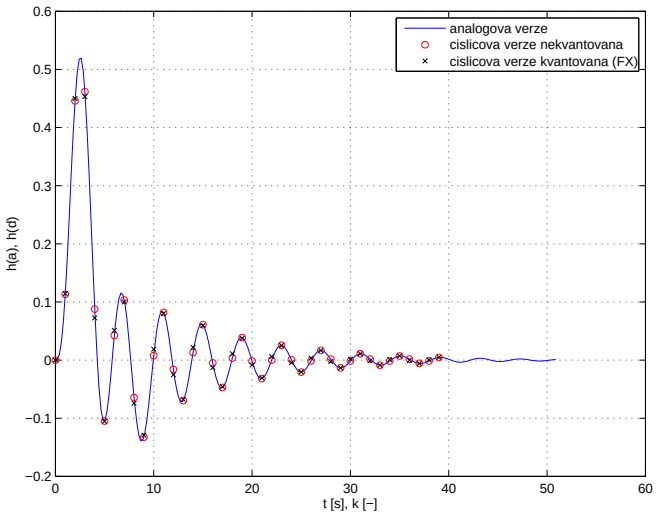
$$y(i) = \begin{bmatrix} \delta_4 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1(i) \\ w_2(i) \\ w_3(i) \\ w_4(i) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} x(i) \tag{3.23}$$

Formát s pevnou řádovou čárkou

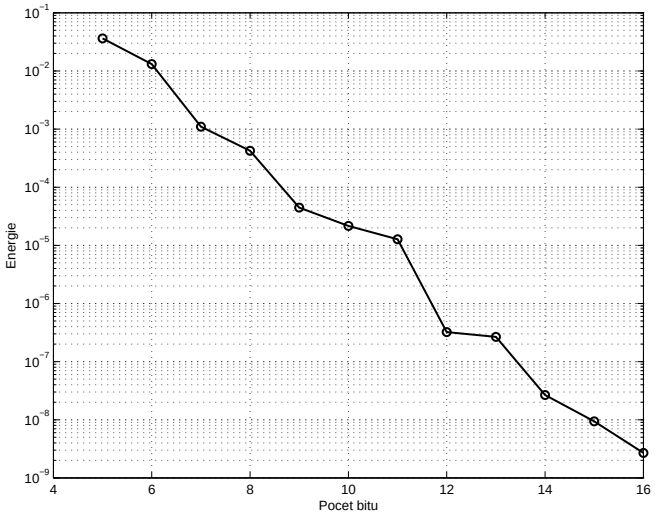
Jak dokazuje obrázek, frekvenční charakteristika kvantovaná u tohoto formátu nejvěrněji kopíruje charakteristiku nekvantovanou (Obr. 3.32). To je zapříčiněno použitou strukturou filtru, neboť u této struktury nedochází k přetečení koeficientů. Energie chybového signálu však nedosahuje nejlepších hodnot (Obr. 3.34). To může být zapříčiněno právě složitou parametrizací této struktury.



Obr. 3.32: Frekvenční charakteristika filtru realizovaná lattice-ladder strukturou, porovnání kvantované verze ve formátu s pevnou řádovou čárkou (16ti bitový formát, délka celočíselné části - 2 bity) a verze nekvantované



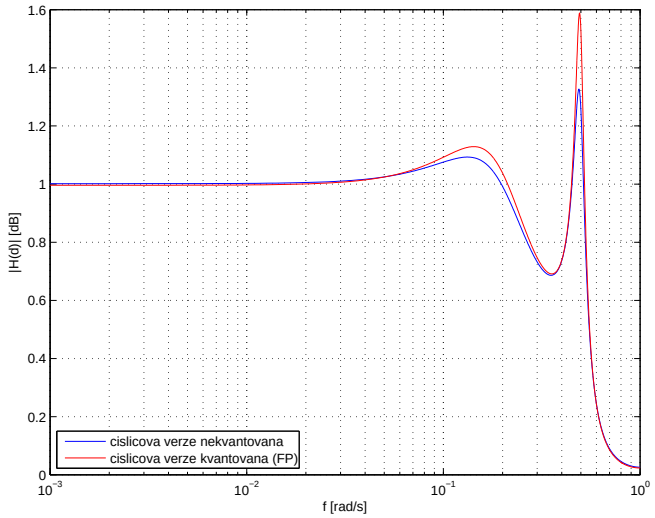
Obr. 3.33: Impulsní charakteristika filtru realizovaná lattice-ladder strukturou, porovnání kvantované verze ve formátu s pevnou řádovou čárkou (16ti bitový formát, délka celočíselné části - 2 bity), verze nekvantované a analogové verze filtru



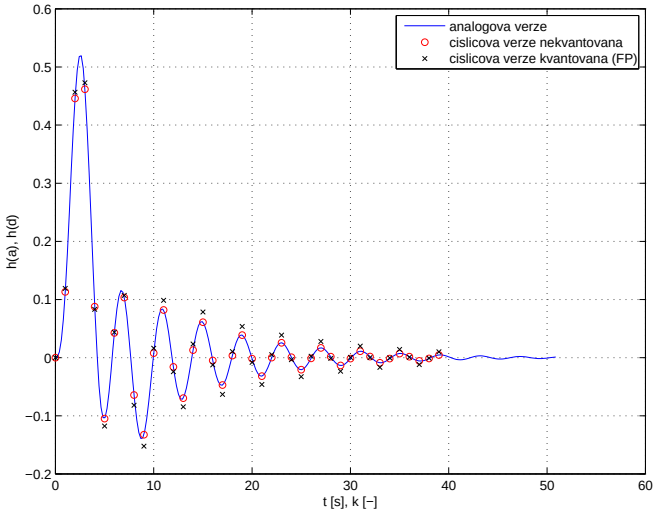
Obr. 3.34: Změny energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Fixed Point - 16ti bitový formát, délka celočíselné části - 2 bity)

Formát s plovoucí řádovou čárkou

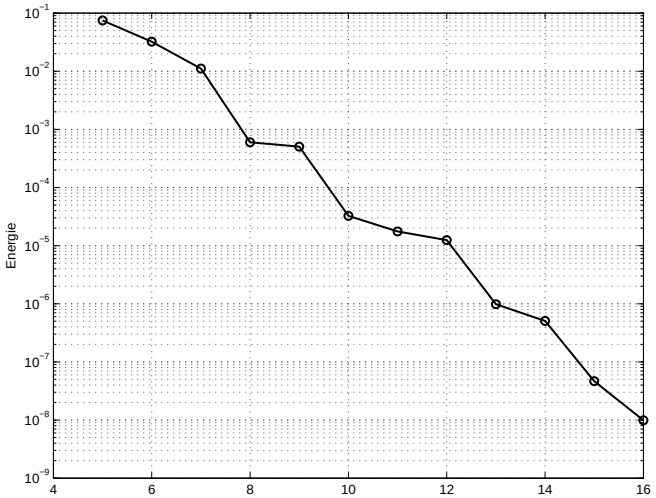
I u této struktury prokazuje formát s plovoucí řádovou čárkou horší výsledky, než ten s pevnou. Obdobně jako u Fixed Pointu, ani v tomto formátu není tato struktura nejlepší co se týče energie chybového signálu (Obr. 3.37). Důsledky lze opět hledat ve velkém počtu parametrů lattice-ladder struktury.



Obr. 3.35: Frekvenční charakteristika filtru realizovaná lattice-ladder strukturou, porovnání kvantované verze ve formátu s plovoucí řádovou čárkou (16ti bitový formát, délka exponentu - 2 bity) a verze nekvantované



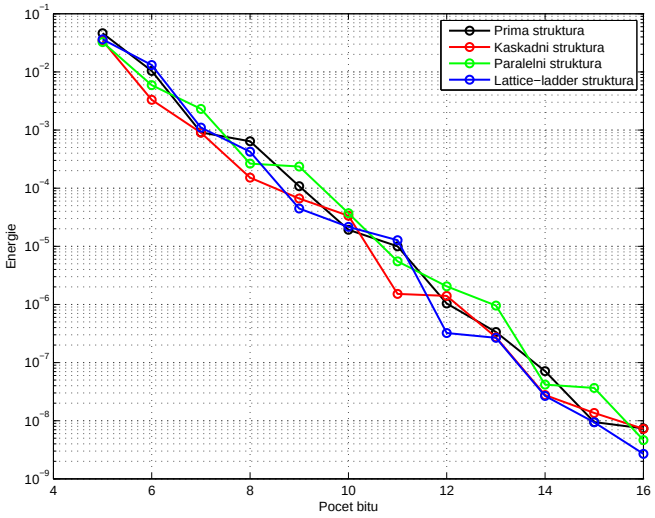
Obr. 3.36: Impulsní charakteristika filtru realizovaná lattice-ladder strukturou, porovnání kvantované verze ve formátu s plovoucí řádovou čárkou (16ti bitový formát, délka exponentu - 2 bity), verze nekvantované a analogové verze filtru



Obr. 3.37: Změny energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Floating Point - 16ti bitový formát, délka exponentu - 2 bity)

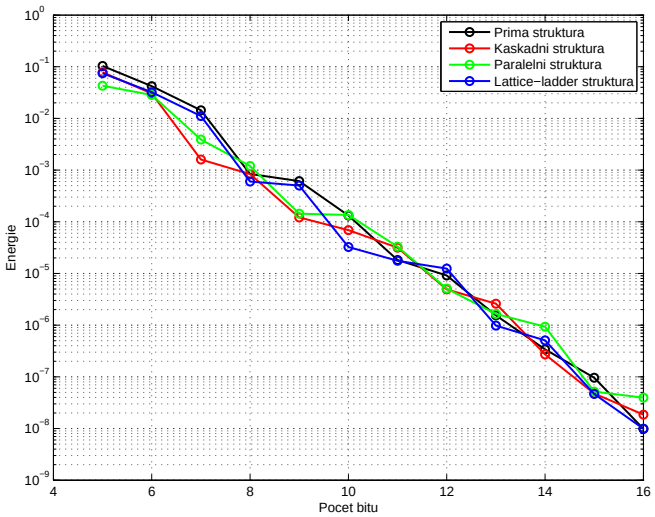
3.4.5 Shrnutí výsledků

Jako první zhodnotíme formát s pevnou řádovou čárkou. Z grafu (Obr. 3.38) je zřejmé, že pro kvantování na velmi malý počet bitů je nejlepší paralelní struktura. Se zvyšujícím se počtem bitů však prokazuje nejlepší výsledky struktura kaskádní. Nejlepší aproximaci frekvenční charakteristiky v plné přesnosti dosahuje zcela určitě lattice-ladder struktura. Což je zapříčiněno její parametrizací, nedochází zde k přetečení koeficientů.



Obr. 3.38: Srovnání změn energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Fixed Point - 16ti bitový formát, délka celočíselné části - 2 bity)

U formátu s plovoucí řádovou čárkou jsme u všech struktur došli k horším výsledkům než u formátu předešlého (Obr. 3.39). Avšak rozdíly nejsou nikterak velké a při kvantování na 5 bitů se nejlepší ze struktur (kaskádní) blíží hodnotami energie chybového signálu k nejhorší (přímé) struktuře kvantované ve formátu s pevnou řádovou čárkou.



Obr. 3.39: Srovnání změn energie chybového signálu v závislosti na počtu bitů, na něž je filtr kvantován (Floating Point - 16ti bitový formát, délka exponentu - 2 bity)

4

Závěr

Hlavními cíli této bakalářské práce bylo zpracování přehledu v současnosti používaných formátů čísel v číslicových systémech a analyzování vlivu struktury číslicového filtru a vlivu použitého formátu čísel na vznik zaokrouhlovacích chyb.

Základem práce bylo navrhnout funkční algoritmy pro jednotlivé typy struktur (přímá, kaskádní, paralelní, lattice-ladder) a jejich následná analýza za použití výpočetního programu MATLAB. Hlavní důraz byl kladen na porovnání frekvenčních charakteristik filtru v plné přesnosti a filtru kvantovaného na 16 bitů. Dále pak porovnání impulsních charakteristik analogového prototypu filtru s kvantovanaou verzí číslicového filtru a nakonec porovnání energie chybového signálu v závislosti na počtu bitů, na než byl filtr kvantován. Toto porovnání bylo provedeno pro všechny struktury a pro dva formáty čísel, formát s pevnou a plovoucí řádovou čárkou.

Na základě mnou získaných výsledků mohu prohlásit, že formát s pevnou řádovou čárkou dosáhl lepších výsledků u všech čtyřech testovaných struktur. Rozdíly nejsou však nikterak veliké a u paralelní struktury se energie chybového signálu za použití formátu s plovoucí řádovou čárkou téměř rovná hodnotě, kterou dosáhl formát s pevnou řádovou čárkou u přímé struktury.

Z hlediska strukturálního vlivu dosahuje nejlepších výsledků v měření energie chybového signálu pro kvantování na velmi malý počet bitů paralelní struktura. Se zvyšujícím se počtem bitů však prokazuje nejlepší výsledky struktura kaskádní. Nejlepší aproximaci frekvenční charakteristiky v plné přesnosti pak dosahuje lattice-ladder struktura, což je zapříčiněno její parametrizací, nedochází zde totiž k přetečení koeficientů. Podle předpokladů vykazovala pro kvantování na malý počet bitů nejhorší výsledky struktura přímá.

Dalším námětem k hlubšímu zpracování této problematiky je testování těchto struktur a formátů čísel v praxi pro různé účely filtrování a porovnání jejich vlivu na konkrétních příkladech.

Literatura

[1] YATES, Randy. *Technical Reference Fixed-Point Arithmetic : An Introduction.*, 2013. [Cit. 26. 3. 2013]. Dostupné z: http://www.digitalsignallabs.com/fp.pdf

[2] TIŠNOVSKÝ, Pavel. *Seriál Fixed Point Arithmetic. Root.cz [online]*, 2006. [Cit. 4. 5. 2013]. Dostupné z: http://www.root.cz/serialy/fixed-point-arithmetic

[3] SOVKA, Pavel a POLLÁK, Petr. *Vybrané metody číslicového zpracování signálů.* Vyd. 1. Praha: Vydavatelství ČVUT, 2001. 206 s. ISBN 80-01-02416-4.

[4] BRTNÍK, Bohumil. *Algoritmy číslicového zpracování signálů.* Praha: BEN - technická literatura, 2011. 1 sv. (v různém stránkování). ISBN 978-80-7300-400-2.

[5] HAASZ, Vladimír, NOVÁK, Jiří a ROZTOČIL, Jaroslav. *Číslicové měřicí systémy.* Vyd. 2., přeprac. Praha: Vydavatelství ČVUT, 2000. 315 s. ISBN 80-01-02245-6.

[6] DAVÍDEK, Vratislav, LAIPERT, Miloš a VLČEK, Miroslav. *Analogové a číslicové filtry.* Vyd. 2. Praha: Vydavatelství ČVUT, 2006. 345 s. ISBN 80-01-03026-1.

[7] *Matlab: Fixed-Point Design for MATLAB Code*, [online], 2013. [Cit. 12. 2. 2013]. Dostupné z: http://www.mathworks.com/help/fixedpoint/

[8] *Matlab: Fixed-Point*, [online], 2013. [Cit. 12. 2. 2013]. Dostupné z: http://www.mathworks.com/help/vision/fixed-point.html

[9] *Matlab: Floating-Point Numbers*, [online], 2013. [Cit. 12. 2. 2013]. Dostupné z: http://www.mathworks.com/help/matlab/matlab˙prog/floating-point-numbers.html