
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# SADA POČÍTAČOVÝCH CVIČENÍ PRO SIGNÁLY A SYSTÉMY

**SET OF COMPUTER EXERCISES FOR SIGNALS AND SYSTEMS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **PETR PÁLKA**
**AUTHOR**

**VEDOUCÍ PRÁCE** **prof. Dr. Ing. JAN ČERNOCKÝ**
**SUPERVISOR**



**BRNO 2022**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

# Zadání bakalářské práce


Student: **Pálka Petr**
Program: Informační technologie
Název: **Sada počítačových cvičení pro Signály a systémy**
**Set of Computer Exercises for Signals and Systems**
Kategorie: Zpracování signálů

Zadání:

1. Seznamte se s dosavadními materiály pro počítačová cvičení kursu ISS.
2. Seznamte se s relevantní teorií ke spektrální analýze a číslicové filtraci.
3. Implementujte dosavadní cvičení v Pythonu s relevantními knihovnami (numpy, atd).
4. Navrhněte vhodný formát dokumentace ke cvičením (např. Python notebook) a dokumentaci realizujte.
5. Navrhněte rozšíření podporující interaktivitu a porozumění signálům a systémům, návrhy konzultujte s vedoucím. Vybrané návrhy realizujte.
6. Vytvořte metodiku hodnocení, vyberte skupinu studentů, hodnocení realizujte a vyhodnoťte.

Literatura:

- materiály na https://www.fit.vutbr.cz/study/courses/ISS/public/

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání, podstatný pokrok v bodě 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Černocký Jan, prof. Dr. Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 29. července 2022
Datum schválení: 25. července 2022

# Abstrakt

Cílem této práce je transformovat současný formát materiálů k počítačovým cvičení z předmětu Signály a Systémy do prostředí Jupyter. Dosavadní materiály jsou členěny podle témat do jednotlivých PDF souborů, které obsahují relevantní teorii a ukázky MATLAB kódu. Tematické okruhy se zabývají základy digitálního zpracování signálů na počítači. Vytvořené Jupyter notebooky obsahují teoretickou oporu z originální předlohy ve formátu Markdown + LaTeX a původní implementace pro prostředí MATLAB je zde převedena do jazyka Python. Tyto materiály navíc přímo prezentují výstupy Python kódu typu graf, přehratelné audio, text, obrázek nebo jejich interaktivní kombinace.



# Abstract

The goal of this bachelor thesis is to transform the current format of materials for computer exercises from the Signals and Systems course into Jupyter environment. Current materials are divided by topics into individual PDF files that contain relevant theory and MATLAB code examples. These PDF materials cover basics of digital signal processing on computer. Produced Jupyter notebooks contain theory from original materials in Markdown + LaTeX, and MATLAB code examples are replaced by Python examples. In addition to current materials, they display output of the Python code which may include graphs, playable audio, text, image or some interactive combination of listed.



# Klíčová slova

zpracování signálu, Jupyter notebook, číslicová filtrace, spektrální analýza, zpracování obrazu, kvantování, vizualizace dat



# Keywords

signal processing, Jupyter notebook, digital filtering, spectral analysis, image processing, quantization, data visualization



# Citace

PÁLKA, Petr. *Sada počítačových cvičení pro Signály a systémy.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing. Jan Černocký

# Sada počítačových cvičení pro Signály a systémy

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Dr. Ing. Jana Černockého. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Petr Pálka
31. července 2022


## Poděkování

Děkuji prof. Dr. Ing. Janu Černockému za jeho odbornou pomoc, trpělivost a čas, který mi věnoval na konzultacích. Děkuji studentům, kteří se zapojili do testování, a Ing. Kateřině Žmolíkové za vytvořené prototypy cvičení v Pythonu.

# Obsah

# Kapitola 1

# Úvod

Oborů zabývající se zpracováním a analýzou signálů je celá řada. Jejich moderními odvětvími jsou například počítačové vidění a zpracování přirozeného jazyka. Ty se zabývají komplexními problémy jako samořídící auta, identifikace osoby, syntéza řeči a strojový překlad přirozeného jazyka. Studenti, kteří by se chtěli zabývat řešeními těchto problémů však musí ovládat základní znalosti o signálech, kterým se tato práce věnuje.

Kvalitní výukové materiály jsou pro studenty důležitým zdrojem informací. V této práci se zaměříme na vytvoření interaktivního materiálu pro digitální zpracování signálu. Programovací jazyk v těchto materiálech je Python v prostředí Jupyter. Studenti mají často zkušenosti s Pythonem díky jeho širokému použití. Navíc jsou v Pythonu implementovány knihovny pro strojové učení, které se dnes běžně používají (například pro rozpoznávání obličejů). Prostředí Jupyter podporuje interaktivní widgety, které jsou velmi užitečné pro demonstraci příkladů z probírané látky.

Text práce je rozdělen do šesti kapitol. Kapitola 2 popisuje dostupné materiály ke zpracování signálu a jejich formy. V dalším kapitole jsou vysvětleny implementační prostředky pro vytvoření počítačových cvičení, s kterým bude také pracovat cílový uživatel. Následující kapitola obsahuje ukázky řešených úloh. Každá taková úloha má v tomto textu svou vlastní podkapitolu s detailním popisem. V předposlední kapitole je shrnuto uživatelského testování se zpětnou vazbou formou dotazníku.

# Kapitola 2

# Materiály k digitálnímu zpracování signálů

V této kapitole se nejprve velmi stručně zaměříme na materiály ke zpracování signálu na různých fakultách a následně si uvedeme přehled materiálu na naší fakultě.

## 2.1 Dosavadní formy materialů

### Digitální zpracování signálu na Cardiffské univerzitě

Předmět s kódem CM0268 vyučovaný na fakultě Počítačových věd a Informatiky Cardiffské univerzity se věnuje digitálnímu zpracování signálu, prostředí MATLAB a generování grafiky v tomto prostředí. V tomto kurzu se nejprve zaměří na práci v prostředí MATLAB, a pak na zpracování signálu, kde se nejčastěji jedná o zvuk.

Materiály pro výuku tvoří [1] převážně prezentace ve formátu PDF s teorii a ukázky kódů. Tyto ukázky jsou také poskytnuty jako soubory s příponou `.m` pro prostředí MATLAB s komentáři. Dále jsou pak například poskytnuty ukázky výstupů, například vliv různých hudebních efektů na signál akustické v souborech typu wav.

### Úvod do výpočetního myšlení na MIT

V kurzu pod jménem *Úvod do výpočetního myšlení* (Introduction to Computational Thinking), který je vyučován od roku 2019 na Massachusettském technologickém institutu, se zabývají kombinací témat ze tří oblastí: informatika, matematika a aplikace. Mezi tyto témata uvádí například datové struktury, Monte Carlo metody, filtrace obrazu a mnohé další. V tomto kurzu je jako programovací jazyk zvolen Julia. Jazyk Julia je velmi efektivní, vysokoúrovňový jazyk.

Pro práci s tímto jazykem využívají reaktivní Pluto notebooky [2]. Jedná se o notebooky podporující vykreslování rovnice, formátovaného textu, spustitelných buněk v jazyce Julia, a řady interaktivních prvků (podobně jako u Jupyter notebooků, které jsou blíže popsány v další kapitole). Jako materiály jsou studentům poskytnuty Pluto notebooky, které jsou velmi často hlavní oporou přednášejícího. Přednášející často demonstruje probírané téma přímo v těchto noteboocích na interaktivních příkladech s kódem v jazyce Julia.

[1] Materiály k předmětu CM0268 jsou dostupné z https://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/

[2] Pluto notebooky jsou dostupné z https://github.com/mitmath/18S191/tree/Spring21/notebooks

## 2.2 Cvičení do předmětu ISS

Studijní materiály do ISS (Signály a systémy) jsou dostupné z veřejné stránky předmětu [3]. V této práci se zaměříme pouze na materiály k praktickému procvičování signálů a systémů [4]. Ty jsou studentům poskytnuty formou samostudia, pro praktické ověření znalostí získaných z přednášek a jako příprava pro vypracovaní samostatného projektu. Pro tuto část studijní etapy jsou studentům k dispozici cvičící a přednášející ke konzultacím.

Cvičení ke zpracování signálu na počítači jsou rozdělena na těchto pět částí:

- **Projekce do bází a Fourierova řada**
- **Práce se zvukem, filtrace a spektrální analýza**
- Zpracování obrazu
- **Náhodné signály**
- Vzorkování a kvantování

Každá část má svou složku obsahující jedno PDF s instrukcemi pro vypracování, relevantní teorií a ukázkami MATLAB kódu. Dále se v nich případně nachází soubory se vstupy, jako je zvuk a obraz reprezentující náš signál pro zpracování, nebo nějaká přichystaná data a soubory s MATLAB funkcemi.

K tučně zvýrazněným částem byly už dříve vytvořeny Jupyter notebooky Ing. Kateřinou Žmolíkovou [5]. V seznamu cvičení zde není uvedeno první, které je zaměřeno na seznámení s prostředím. U původních cvičení se jedná o tipy k práci s MATLABem. Pro úvod a tipy k Pythonu studenti mohou využít notebooku od Ing. Žmolíkové.

Základem této práce jsou tyto uvedené materiály. Odkazy na tyto materiály jsou uvedeny v poznámkách. V nové vytvořené formě vytvořených materiálů (kterou si představíme v další kapitole), se tedy vyskytuje upravená teorie původních cvičení, upravený kód poskytnutých Jupyter notebooků, a rozšíření některých kapitol.

[3] Stránka předmětu ISS: https://www.fit.vutbr.cz/study/courses/ISS/public/.cs
[4] Materiály cvičení: https://www.fit.vutbr.cz/study/courses/ISS/public/proj_studijni_etapa/
[5] Dosavadně vytvořené Jupyter notebooky: https://www.fit.vutbr.cz/~izmolikova/ISS/project/

# Kapitola 3

# Implementační prostředky

V této kapitole si představíme implementační prostředky, pomocí kterých byla vytvořena nová forma studijních materiálu. S těmito prostředky bude také pracovat cílový uživatel (student).

## 3.1 Jazyk Python

Python je interpretovaný, interaktivní, objektově orientovaný programovací jazyk. Zahrnuje moduly, výjimky, dynamické psaní, velmi vysoké dynamické datové typy a třídy. Podporuje více programovacích paradigmat nad rámec objektově orientovaného programování, jako je procedurální a funkční programování. Má rozhraní pro mnoho systémových volání a knihoven, stejně jako pro různé okenní systémy, a je rozšiřitelný v C nebo C++ [1]. Jednoduchá syntaxe Pythonu klade důraz na čitelnost, a proto snižuje náklady na údržbu programu.

V Pythonu jsou implementovány knihovny pro strojové učení, které se dnes často používá pro zpracování signálu. Většina studentů má navíc nějaké zkušenosti s tímto jazykem, protože má široké využití, a je často snadno použitelný. Z těchto důvodů je vhodným implementačním jazykem těchto materiálů.

## 3.2 Jupyter notebook

Tato část je vhodně upravený text dokumentace projektu Jupyter dostupného z [2] a čtenáři dává přehled o formě cílových materiálu tvořených v této práci.

Jupyter Notebook je interaktivní výpočetní prostředí, které uživatelům umožňuje vytvářet dokumenty v poznámkovém bloku, které zahrnují: živý kód, interaktivní widgety, plochy, narativní text, rovnice, obrázky a video. Tyto dokumenty poskytují úplný a samostatný záznam výpočtu, který lze převést do různých formátů a sdílet s ostatními pomocí e-mailu, Dropboxu, systémů pro správu verzí (jako git/ GitHub) nebo nbviewer.jupyter.org.

### Webová aplikace notebooku

Tato aplikace uvažuje jako dokumenty notebooku pouze soubory s příponou `.ipynb`. Uživatelům umožňuje:

1. Upravovat kód v prohlížeči s automatickým zvýrazňováním syntaxe, odsazováním
2. Spoustět kód z prohlížeče s výsledky výpočtů připojenými ke kódu, který je vygeneroval

3. Sledovat výsledky výpočtů s multimediálními reprezentacemi, jako jsou HTML, LaTeX, PNG, SVG, PDF atd.

4. Vytvářejte a používejte interaktivní widgety JavaScriptu , které vážou interaktivní ovládací prvky uživatelského rozhraní a vizualizace s reaktivními výpočty na straně jádra.

5. Vytvořte narativní text pomocí značkovacího jazyka Markdown

6. Zahrnout matematické rovnice pomocí syntaxe LaTeXu v Markdown, které jsou vykreslovány v prohlížeči pomocí MathJax (javascriptová knihovna)

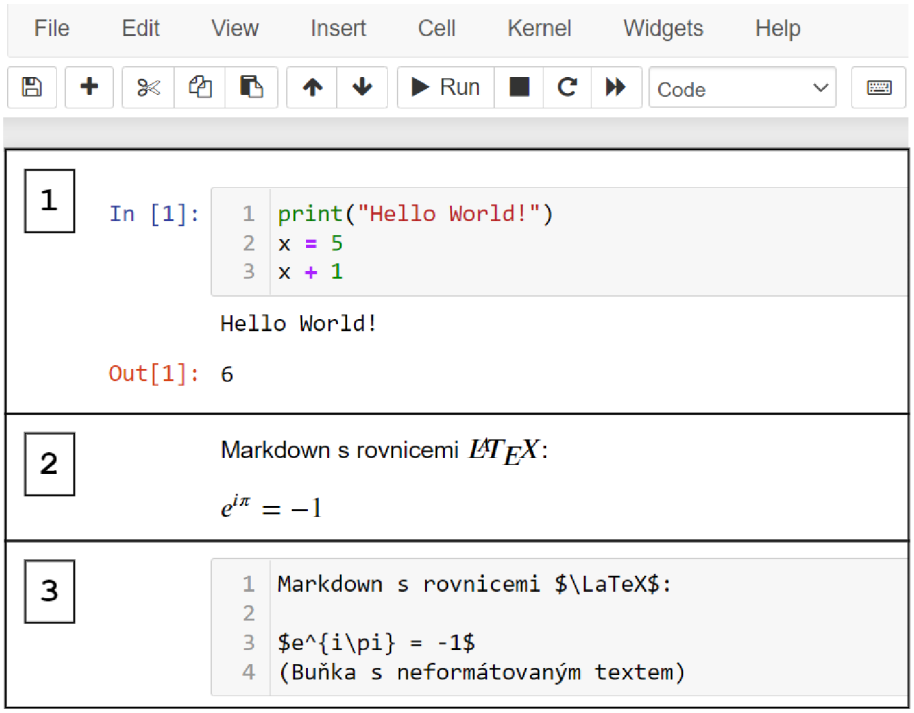
**Jádra**

Prostřednictvím jádra a architektury zasílání zpráv Jupyter umožňuje notebook spouštět kód v řadě různých programovacích jazyků. Pro každý dokument poznámkového bloku, který uživatel otevře, spustí webová aplikace jádro, které spustí kód pro daný poznámkový blok. Každé jádro je schopno spouštět kód v jediném programovacím jazyce a existují jádra dostupná v následujících jazycích: **Python**, Julia, R, Ruby, Haskell, Scala, node.js. Každé z těchto jader komunikuje s webovou aplikací notebooku a webovým prohlížečem pomocí protokolu zpráv JSON.

### Dokumenty k notebooku

Dokumenty obsahují vstupy a výstupy interaktivní relace a také narativní text, který doprovází kód, ale není určen k provádění. Výstup generovaný spuštěným kódem, včetně HTML, obrázků, videa a grafů, je zabudován do notebooku, což z něj činí úplný a samostatný záznam výpočtu. Webová aplikace notebooku vnímá pouze soubory s příponou **.ipynb** jako dokumenty.
Notebooky se skládají z lineární sekvence buněk. Existují tři základní typy buněk:

1. **Spustitelné buňky kódu** - vstup a výstup živého kódu, který je spuštěn v jádře.
2. **Buňky Markdown** - narativní text s vloženými rovnicemi LaTeX.
3. **Nezpracované buňky** - neformátovaný text, který je zahrnut bez úprav při převodu poznámkových bloků do různých formátů pomocí nbconvert.




Obrázek 3.1: Typy buněk v noteboocích Jupyter

Interně jsou dokumenty notebooku data JSON s binárními hodnotami zakódovanými base64. To jim umožňuje číst a programově manipulovat jakýmkoli programovacím jazykem. Vzhledem k tomu, že JSON je textový formát, dokumenty poznámkového bloku jsou vhodné pro správu verzí.

Notebooky lze exportovat do různých statických formátů včetně HTML, reStructeredText, LaTeX, PDF a prezentací pomocí nástroje Jupyter nbconvert. Jakýkoli dokument poznámkového bloku dostupný z veřejné adresy URL nebo na GitHubu sdílet prostřednictvím **nbvieweru**. Tato služba načte dokument poznámkového bloku z adresy URL a vykreslí jej jako statickou webovou stránku. Výslednou webovou stránku tak lze sdílet s ostatními, aniž by si museli instalovat Jupyter Notebook. Vyzkoušet si vzdálený Jupyter notebook bez jakékoli instalace je možné na https://docs.jupyter.org/en/latest/start/index.html.

### 3.2.1 Jupyter widgets

Jupyter Widgets je primárně framework pro poskytování interaktivních ovládacích prvků. Balíček ipywidgets také poskytuje základní lehkou sadu základních ovládacích prvků formuláře, které používají tento rámec. Mezi tyto ovládací prvky patří textová oblast, textové pole, ovládací prvky pro výběr a vícenásobný výběr, zaškrtávací políčko, posuvníky, panely karet, rozložení mřížky atd [3].

Ukázka widgetu je v této práci uvedena například na obrázku 4.1. Tento widget pro ovládaní obsahuje: prvek pro výběr (zde typ filtru), posuvník pro parametr omega $\omega$, a zaškrtávací políčko pro parametr `symmetry`. Uživatel si těmito prvky zobrazuje různé příklady výstupů pro zvolené parametry, pro hlubší pochopení probírané látky.

Interaktivní widgety jsou v těchto cvičení vytvořeny pomocí základních prvků. Hlavně pomocí funkce `interactive`. Jejími vstupy je funkce, která zaopatřuje například vykreslení grafů, fixní parametry nebo parametry nastavitelné ovládacími prvky widgetu. Jednoduchý tutoriál pro tvorbu těchto widgetů je dostupný v dokumentaci v ipywidgets [1].

[1] Dokumentace k ipywidgets https://ipywidgets.readthedocs.io/en/7.6.2/user_guide.html

# Kapitola 4

# Ukázky řešení a výstupů

Tato kapitola se věnuje popisu vybraných částí řešení a prezentuje výstupy, které jsou vytvořeny po spuštění kódu nacházejícího se v buňkách notebooků. Téma z notebooku **Projekce do bází a Fourierova řada** v této textové části práce není. Z každého dalšího notebooku o signálech je zde uvedena jedna podkapitola. Celkem jde tedy o čtyři témata:

1. **Číslicová filtrace** - Práce se zvukem, filtrace a spektrální analýza
2. **Filtrace 2D signálů** - Základy zpracování obrazu
3. **Souborové odhady** - Náhodné signály
4. **Kvantování audio signálu a obrazu** - Vzorkování, aliasing a kvantování

Krátký popis jednotlivých části se skládá ze základní teorie, kódu a generovaného výstupu.

## 4.1 Číslicová filtrace

V této části si ukážeme jak filtrovat audio signál. Toto téma se nachází v notebooku *Zpracování zvuku*. Nejdříve je v notebooku uvedeno jak postupovat krok za krokem při analýze filtru, včetně výstupů tohoto postupu. Na konci této podkapitoly cvičení se v notebooku nachází interaktivní widgety, které prezentují výstupy všech těchto kroků pro zvolený filtr. Student si zde může definovat vlastní filtr uvedením koeficientů $b$, $a$. Jako tři příklady filtrů jsou uvedeny:

1. jednoduchá dolní propusť (zvýrazní nízké frekvence a oslabí vysoké)
2. ostrá pásmovou propusť (zvýrazňující velmi úzký interval frekvencí)
3. nestabilní filtr (velmi podobný filtru 2, ale navržený tak, aby byl nestabilní)

Všechny filtry se v notebooku zadávají pomocí jejich koeficientů $b$, $a$, které určují diferenční rovnici filtru:

$$y[n] = \sum_{k=0}^{Q} b_k x[n-k] - \sum_{k=1}^{P} a_k y[n-k],$$

kde $y$ je výstupní signál a $x$ vstupní signál.
U každého filtru si zobrazíme jeho impulsní odezvu, frekvenční charakteristiku, nuly a póly. Podle polohy pólů navíc zhodnotíme stabilitu filtru a nakonec provedeme samotnou filtraci našeho signálu.

**Impulsní odezva** filtru $h[n]$ je reakce filtru na jednotkový impuls, dává nám informaci o chování filtru v čase. V Pythonu ji získáme přímo vytvořením jednotkového impulsu a jeho vyfiltrováním použitím funkce `scipy.signal.lfilter`.

**Frekvenční charakteristika** filtru $H(e^{j\omega})$ ukazuje chování filtru pro jednotlivé frekvenční složky obsažené v signále. Můžeme z ní tedy vyčíst, které frekvence filtr posiluje nebo které naopak potlačuje. V Pythonu k jejímu získání použijeme funkci `scipy.signal.freqz`. Protože frekvenční charakteristika je komplexní funkce, budeme ji zobrazovat zvlášť jako její modul (magnitudu) a argument (fázi). Z magnitudy můžeme vyčíst, jak filtr různé frekvenční složky v signále posílí nebo potlačí, z argumentu, jak se jednotlivé frekvenční složky zpozdí nebo předběhnou.

**Nuly a póly** filtru jsou body v komplexní rovině $z$, která tvoří definiční obor přenosové funkce filtru $H(z)$. Přenosová funkce je zobecnění frekvenční charakteristiky – zatímco frekvenční charakteristika je definovaná pouze pro hodnoty $e^{j\omega}$, kde $\omega$ je kruhová frekvence, přenosová funkce je definovaná pro jakékoli komplexní číslo $z$. Obecně má přenosová funkce tvar

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{Q} b_k z^{-k}}{1 + \sum_{k=1}^{P} a_k z^{-k}}.$$

Dosazením $e^{j\omega}$ za $z$ bychom z přenosové funkce mohli vyčíst frekvenční charakteristiku filtru. Pro nás jsou v tomto cvičení důležité nuly a póly přenosové funkce. Nuly jsou takové hodnoty $z$, pro které je přenosová funkce nulová (tzn. její čitatel je roven nule), póly jsou takové hodnoty $z$, pro které je přenosová funkce nekonečná (tzn. její jmenovatel je roven nule). V Pythonu můžeme nuly a póly filtru získat pomocí funkce `scipy.signal.tf2zpk`.

**Stabilitu** filtru můžeme zhodnotit pomocí polohy pólů. Pokud je filtr stabilní, generuje pro omezené vstupy omezené výstupy. Omezenými výstupy myslíme, že tyto hodnoty neutíkají do $\infty$ ani $-\infty$. Platí, že stabilní filtr má všechny póly uvnitř jednotkové kružnice, neboli že

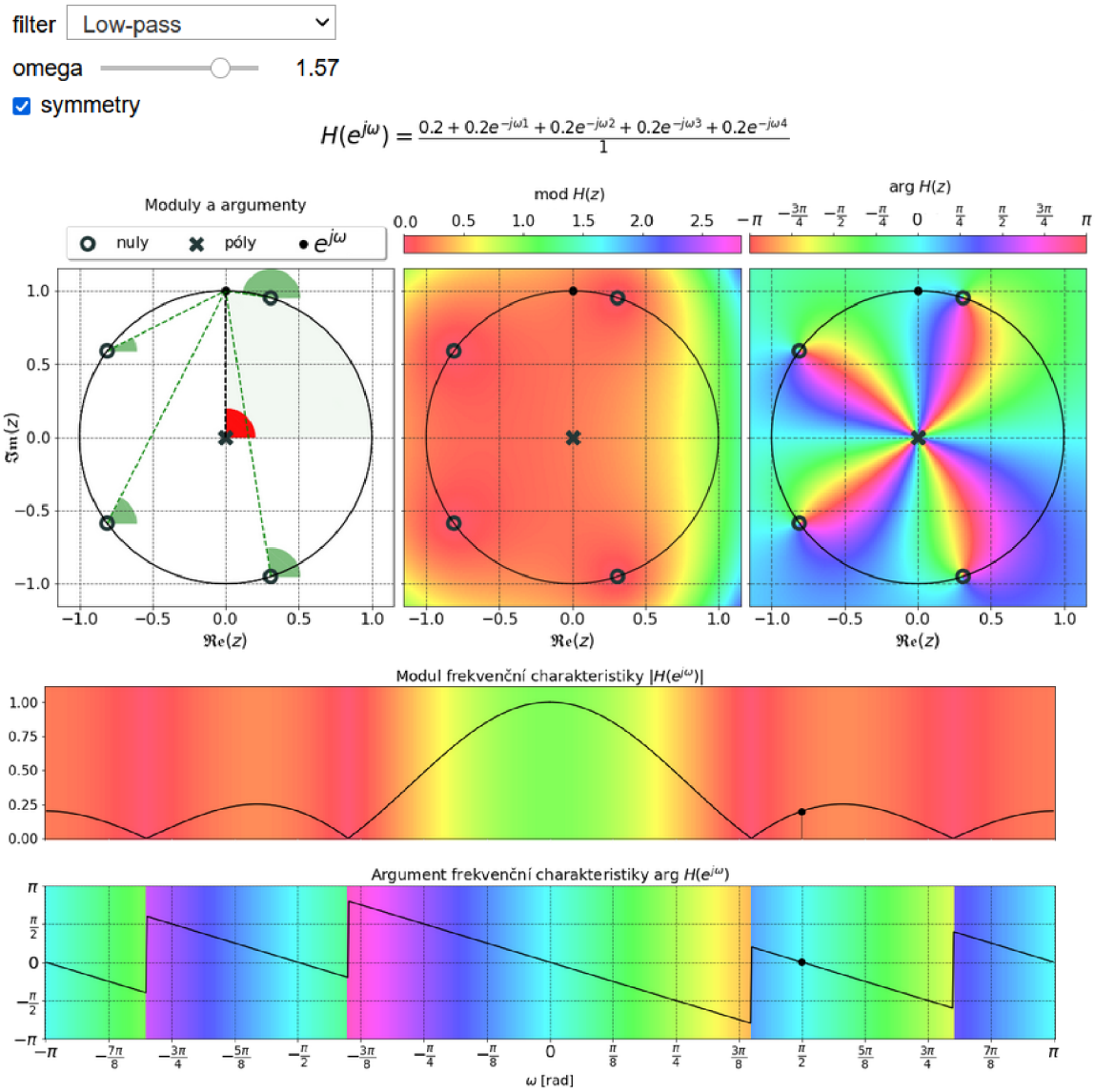
$$|p_k| < 1 \quad \forall k,$$

v opačném případě je filtr nestabilní.

Nakonec provedeme samotnou filtraci našeho signálu pomocí filtru zadaného koeficienty $b$, $a$ funkcí `scipy.signal.lfilter`, a výsledek porovnáme s originálem poslechem.

### 4.1.1 Interaktivní ukázky filtrace

Následují snímky obrazovky widgetů, které shrnují informace o zvolených filtrech.

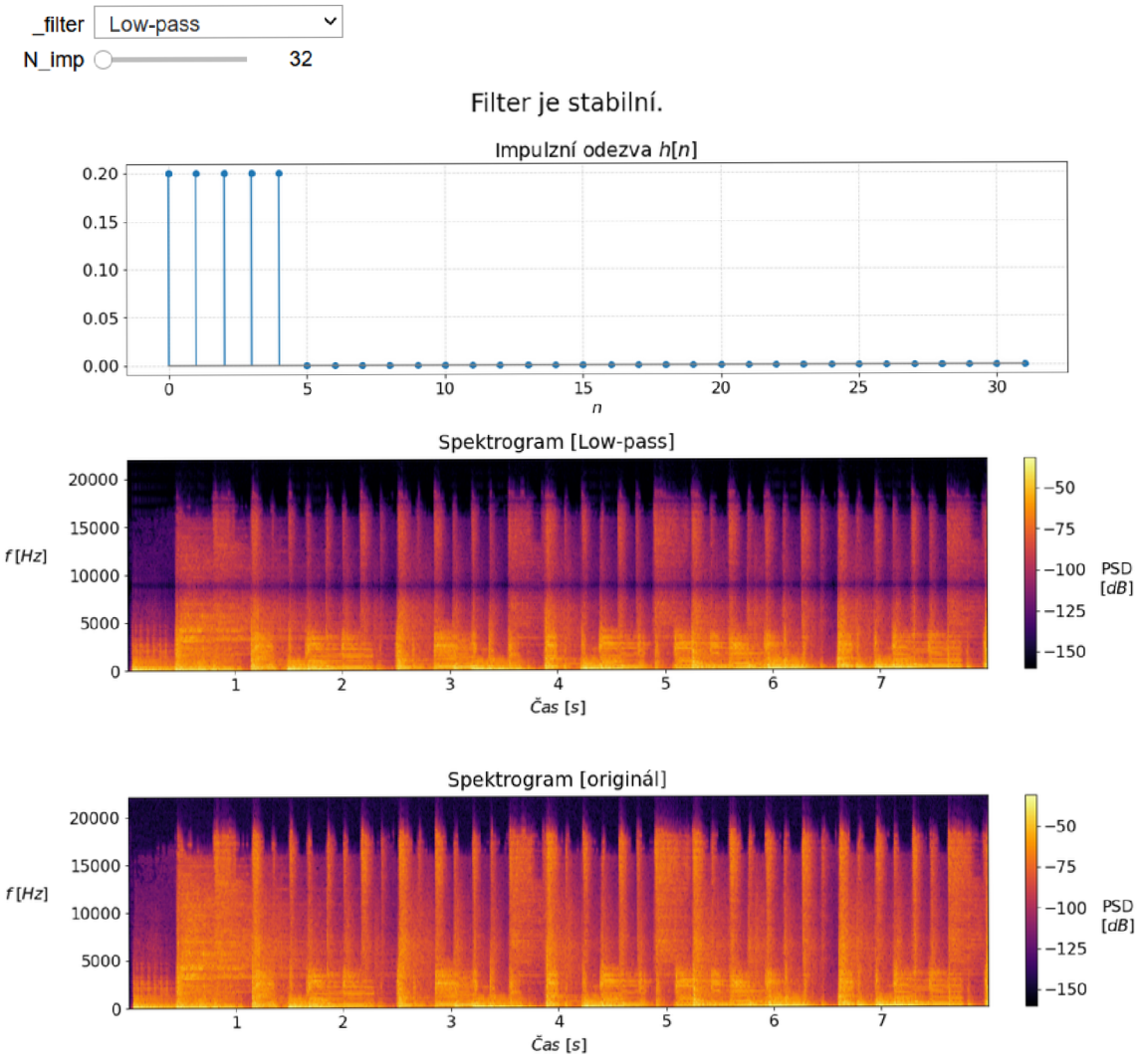



Obrázek 4.1: Interaktivní widget – zvolený filtr *Dolní propust*

Na obrázku 4.1 vidíme několik různých pohledů na stejnou informaci. Ve všech čtvercích máme zobrazené pozice nul a pólů. Póly jsou v tomto případě umístěné 4 na stejném místě $[0, 0i]$. V prvním čtverci máme vyobrazené vzdálenosti od nul a pólů k bodu $e^{j\omega}$ na jednotkové kružnici. Pokud vzdálenosti od bodu $e^{j\omega}$ od nul mezi sebou vynásobíme a podělíme je součinem vzdáleností od pólů (ten je v tomto případě roven 1), dostaneme modul frekvenční charakteristiky v bodě pro $\omega = 1.57$. Jelikož jsou všechny vzdálenosti od nul vetší než 1 a dělíme jedničkou, vyjde nám nenulový modul. Pokud bychom sečetli úhly naznačené v prvním čtverci naznačené zelené se znaménkem plus a červené se znaménkem mínus (znovu poznamenejme, že póly jsou 4 na stejném místě, takže se v obrázku tento úhel nachází čtyřikrát), získáme argument frekvenční charakteristiky pro zvolenou $\omega$. Zeleně vyznačené úhly dají dohromady 360° a čtyřnásobný červený má velikost -90°, což dá dohromady -360°. Výsledný argument je tedy roven 0° nebo také 0 radiánům. Svírané úhly (argumenty) jsou sestrojeny podle pomyslných vodorovných os, které procházejí nuly/póly, a úsečkou od nul/pólů k bodu $e^{j\omega}$).

Uvedené výsledky jsou lépe zobrazeny v dalších grafech, kde je zobrazen modul (druhý čtverec) a argument (třetí čtverec) komplexní funkce $H(z)$ pro intervaly $<-1.15, 1.15>$ reálné a $<-1.15, 1.15>$ imaginární složky. Všimněte si, že jsou zde charakteristiky uvedeny v radiánech. Pro nějaký náš signál s vzorkovací frekvencí $f_s$, bychom měřítko upravili vynásobením hodnotou $f_s/2\pi$ a dále pracovali s Hz.

Další interaktivní widget ukazuje impulzní odezvu filtru a spektrogram signálu po filtraci. Poskytne taky informaci v textové formě o tom, zda je vybraný filtr stabilní.




Obrázek 4.2: Interaktivní widget 2 – zvolený filtr *Dolní propust*, originální signál je uložen v souboru *music.wav* se vzorkovací frekvencí 44100 Hz.

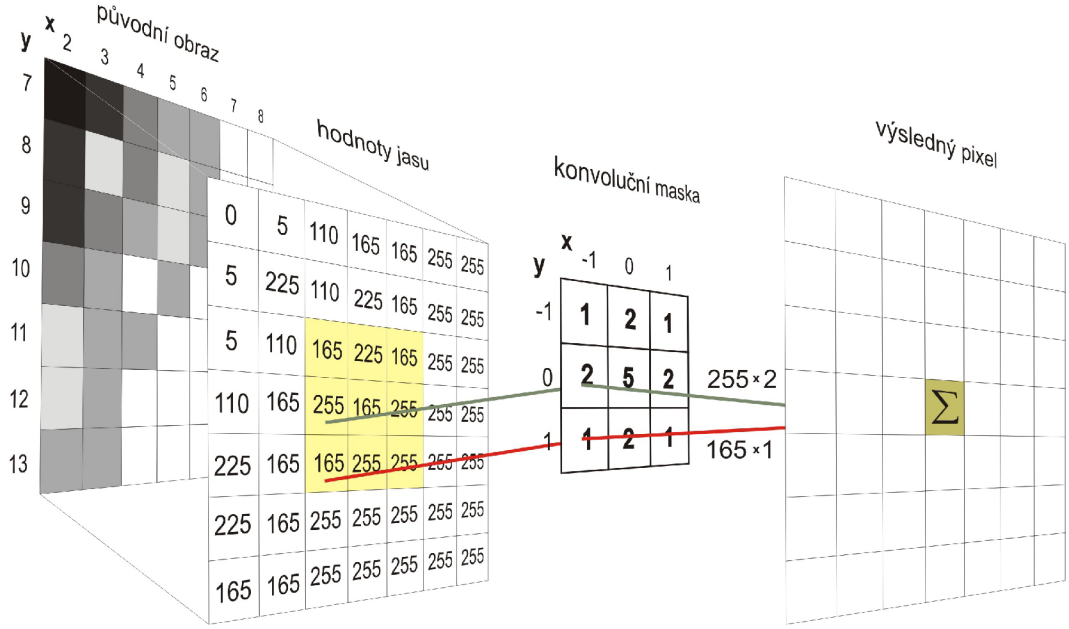
## 4.2 Filtrace 2D signálů

### 4.2.1 Filtrace pomocí konvoluční masky

Pro 2D signály se lineární filtry většinou realizují pomocí filtrů s konečnou impulsní odezvou FIR, které mají definovanou impulsní odezvu jako:

$$h[i,j] \text{ pro } \quad -\frac{I}{2} \leq i \leq \frac{I}{2}, \quad -\frac{J}{2} \leq j \leq \frac{J}{2}.$$

Filtry bývají nejčastěji definovány maticí nazývanou maska s velikostí $(I+1) \times (J+1)$. Samotná filtrace je prováděna následovně: matici resp. masku $h$ přiložíme na každý pixel obrazu, odpovídající hodnoty spolu vynásobíme a výsledky sečteme. Tím získáme výstupní hodnotu pro daný pixel [6]. Obrázek 4.3 tento postup demonstruje.




Obrázek 4.3: Princip diskrétní dvourozměrné konvoluce. [1]

Matematický zápis tohoto výpočtu můžeme zapsat jako:

$$y[k,l] = x[k,l] \star h[k,l] = \sum_{i=-\frac{I}{2}}^{\frac{I}{2}} \sum_{j=-\frac{J}{2}}^{\frac{J}{2}} h[i,j]x[k-i,l-j],$$

kde $y$ s indexy $k$ a $l$ je výsledná hodnota na pixelu s těmito indexy, $\star$ značí operaci konvoluce, $h$ je zvolená maska a $x$ původní 2D signál. V notebooku je tento algoritmus navíc vyjádřen Python kódem připomínajícího syntax jazyka C, aby byl student schopný implementovat tento algoritmus i v jiných programovacích jazycích. K samotné filtraci se dále používá funkce `scipy.signal.convole2d`. Nejprve je demonstrována na obrázku s masky pro rozmazání a zaostření obrazu, a následně je použita pro detekci hran (viz následující podkapitola).

[1] obr. 4.3 byl převzat z https://commons.wikimedia.org/wiki/File:Konvoluce_2rozm_diskretni.jpg

### 4.2.2 Základní detekce hran

V předchozím příkladě byla představena operace konvoluce a její aplikace na dvě matice: obraz a jádro. Ve vytvořeném cvičení jsou dále uvedeny příklady jader, a jejich vliv při konvoluci s obrázkem. V této části si ukážeme příklad, který v tomto cvičení následuje. Pomocí konvoluce se pokusíme detekovat hrany v obrázku, a ukážeme si jaké informace získáme pokud na výsledek navíc aplikujeme některé elementární matematické funkce.

#### Zjednodušený Cannyho hranový detektor

Bude se jednat o první 2 kroky algoritmu Cannyho hranového detektoru. Tyto a další kroky, které byly pro jednoduchost cvičení vynechány, jsou popsány v článku [4]. Odkazovaný článek napsal samotný tvůrce tohoto detektoru John F. Canny.

**1) Eliminace šumu v obrazu**

Pro přesnější detekci hran je vhodné vstupní obraz zbavit šumu. Zbavíme se tím detekce falešných hran. V Cannyho algoritmu se většinou používá jádro pro gaussovské rozostření. To lze vypočítat pomocí Gaussovské funkce, která je pro 2D definována jako:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Výsledná matice se musí vydělit sumou její hodnot, aby se její součet rovnal jedné. Pokud by byla větší, pak by výsledný obraz byl světlejší a v opačném případě tmavší.

**2) Spočtení velikosti a směru gradientu**

Pro zjištění hodnot gradientu aplikujeme nejčastěji používaný Sobelův operátor, a to o velikosti 3x3. Hodnoty těchto operátorů jsou vypsány v následujících maticích:

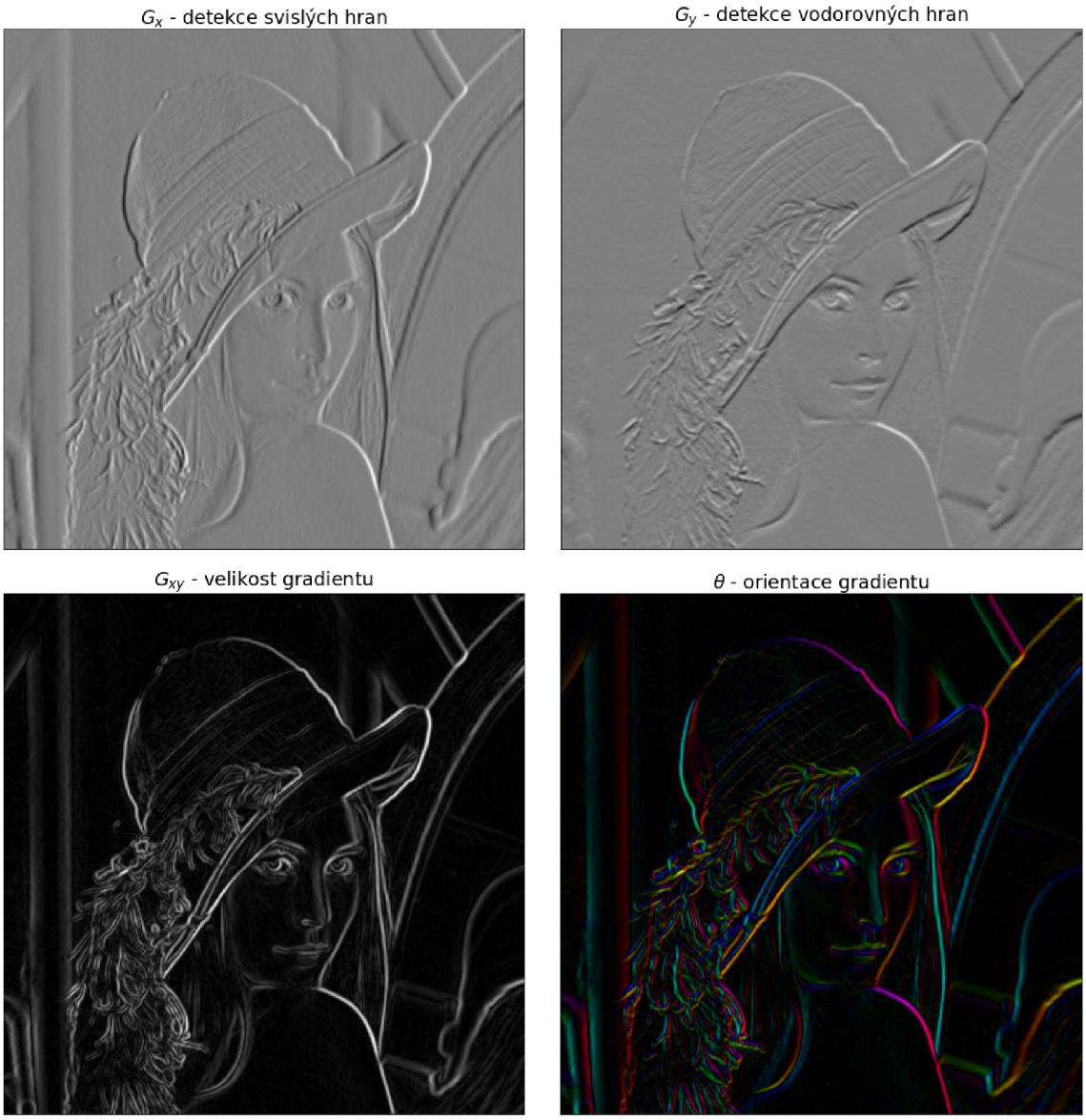
$$E_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad E_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

Tyto matice aplikujeme na originální obraz každou zvlášť. Matice $G_x$ slouží k detekci horizontálních hran a $G_y$ pro nalezení vodorovných. Případně se dají aplikovat současně (tj. po sobě na stejný obraz), kde by výsledkem byla absolutní hodnota gradientu pro daný pixel.

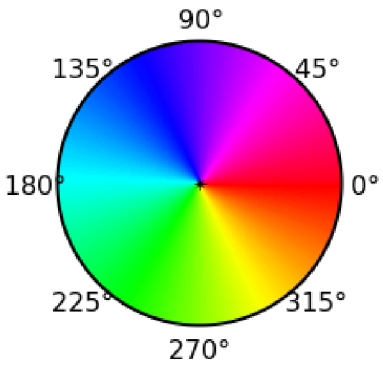
$$G_{xy} = \sqrt{G_x^2 + G_y^2} \quad \theta = \arctan\left(\frac{G_y}{G_x}\right),$$

kde $G_x$, $G_y$ jsou výsledky po konvoluci s uvedenými maticemi $E_x$, $E_y$, a $G_{xy}$ je velikost intenzity gradientu a $\theta$ je směr nebo také úhel gradientu. Tento úhel se pro další zpracování většinou zaokrouhluje na násobky 45°.

Na další straně jsou zobrazeny všechny výsledky druhého kroku (tj. aplikace masek $E_x$ a $E_y$ zvlášť, velikost a úhel gradientu). Originální obraz můžeme v textu najít zde.




Obrázek 4.4: Výstupy jednoduché detekce hran na základě gradientu, originální obrázek *lena.gif* je v textu uveden zde 4.9.




Orientace gradientu reprezentována barvou

### 4.2.3 Filtrace ve frekvenční oblasti

V předchozích dvou podkapitolách byla předvedena filtrace 2D signálu pomocí masky a ukázka její aplikace. Zde si ukážeme filtraci pomocí rychlé Fourierovy transformace, která se používá ve standardních knihovnách v případech, kdy je naše maska větších rozměrů. Operaci konvoluce obrazu s jádrem je v těchto případech výhodnější počítat ve frekvenčním oblasti. Napřed si uvedeme stručnou teorii k 2D-DFT převzatou ze školních materiálů [6].

2D signál můžeme reprezentovat maticí:

$$x[k,l] = \begin{bmatrix} x[0,0] & x[0,1] & \cdots & x[0,L-1] \\ x[1,0] & x[1,1] & \cdots & x[1,L-1] \\ \vdots & & & \vdots \\ x[K-1,0] & x[K-1,1] & \cdots & x[K-1,L-1] \end{bmatrix}$$

Pro 2D signál, definujeme 2D Fourierovu transformaci jako:

$$X(f,g) = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} x[k,l] e^{-j2\pi(fk+gl)}$$

kde $f$ a $g$ jsou frekvence obrazu. Obrázky s kterými pracujeme jsou vzorkované a proto $f$ a $g$ jsou **normalizované frekvence obrazu**. Uvedená rovnice odpovídá DTFT (Discrete Time Fourier Transform), kde $f$ a $g$ můžou nabýt libovolných hodnot. Při praktickém použití pracujeme s 2D diskrétní Fourierovou transformací (2D-DFT):

$$X[m,n] = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} x[k,l] e^{-j2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)}$$

Vypočteme pouze pro diskrétní frekvence:

$$f = m\Delta f \quad g = n\Delta g,$$

kde

$$\Delta f = \frac{1}{M} \quad \Delta g = \frac{1}{N}.$$

$M, N$ jsou celými čísly, nejčastěji používáme $M = K, N = L$.
2D-DFT může být spočtena pro každou dimenzi zvlášť

$$X[m,n] = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} x[k,l] e^{-j2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)} = \sum_{k=0}^{K-1} e^{-j2\pi\frac{mk}{M}} \sum_{l=0}^{L-1} x[k,l] e^{-j2\pi\frac{nl}{L}},$$

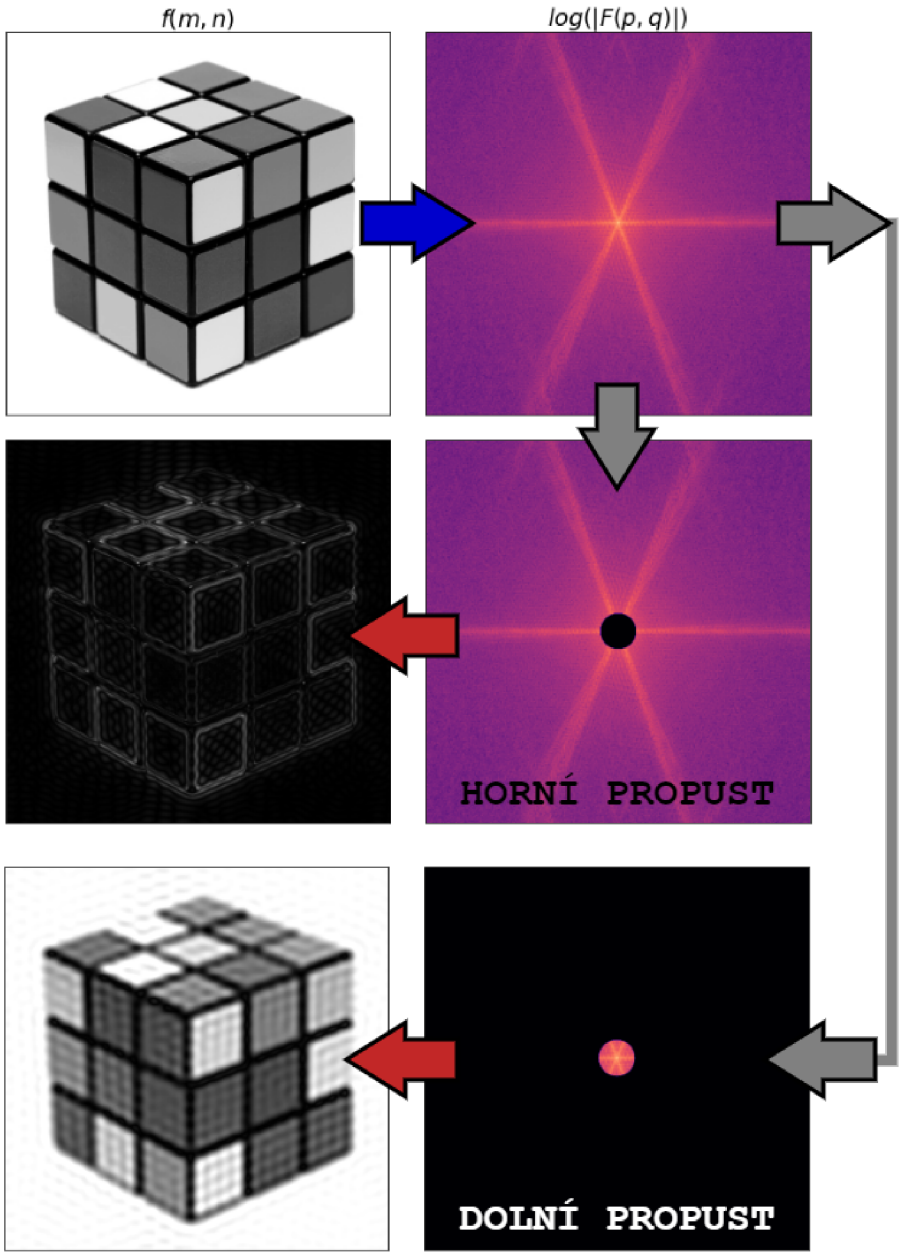
takže můžeme spočítat 2D-DFT jako aplikování dvou 1D-DFT - napřed řádky a potom sloupce (nebo obráceně).

Inverzní Fourierova transformace v dvourozměrném prostoru (2D-IDFT):

$$x[k,l] = \frac{1}{MN} \sum_{m=0}^{K-1} \sum_{n=0}^{L-1} X[m,n] e^{+j2\pi\left(\frac{mk}{M}+\frac{nl}{L}\right)}.$$

Vzhledem k vlastnostem podobných u DFT je matice $X[m,n]$ symetrická, proto se stačí zaměřit na první kvadrant s indexy $m = 0 \ldots \frac{M}{2} - 1, n = 0 \ldots \frac{N}{2} - 1$.

V Pythonu můžeme pro výpočet využít funkce `numpy.fft.fft2`. Výsledek této operace budeme nazývat spektrum (frekvencí) obrazu. Naším vstupem této funkce bude obrázek ve stupních šedi s hodnoty typu `float`. U barevných obrázků by se 2D Fourierova transformace aplikovala na každou barevnou složku zvlášť (jako bychom pracovali se třemi různými obrázky). Často se před vizualizací nebo při operací toto symetrické spektrum posouvá tak, aby se nultý koeficient nacházel uprostřed spektra (funkce `numpy.fft.fftshift`). Operaci tohoto posuvu v tomto cvičení používáme jak při vizualizaci spektra, tak před aplikací filtru, který se aplikuje kolem nultého koeficientu spektra. Pro vizualizaci spektra se většinou použije pouze jeho modul, ke kterému přičteme hodnotu 1, a spočteme logaritmus této hodnoty. Tento výsledek je vhodný pouze pro vizuální vnímaní člověka.

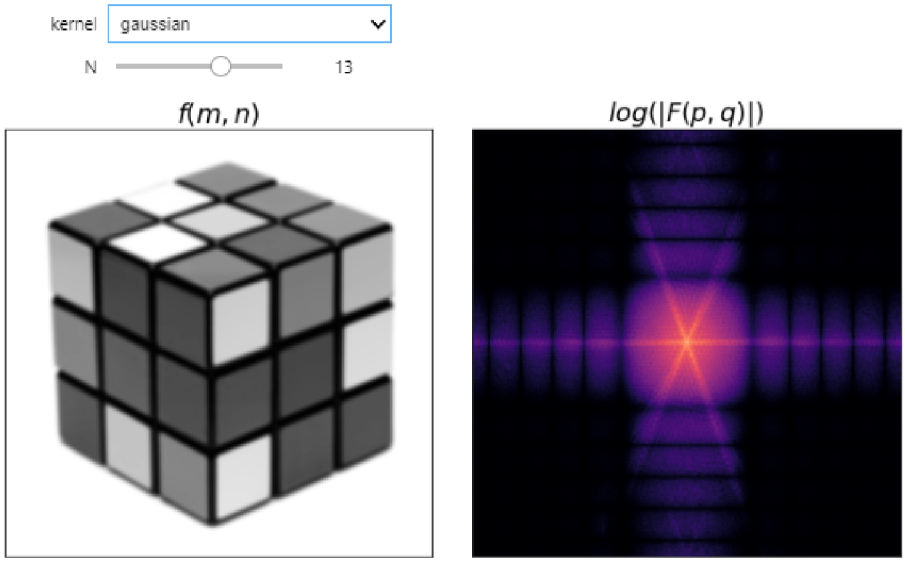



Obrázek 4.5: Filtrace ve frekvenční doméně pomocí FFT

Na obrázku 4.5 modrá šipka značí přímou 2D FFT, červená značí zpětnou (inverzní) 2D FFT, šedé šipky značí filtraci původního spektra obrázku a to buď nízkých nebo vysokých frekvencí (podle vepsaných popisků). Filtr nastavuje vybrané frekvence na hodnotu nula, která je zde reprezentována černou barvou, což odpovídá barevnému schématu **magma**.

Tento obrázek je vytvořen ze snímků obrazovky s interaktivním widgetem Jupyter notebooku. Na tomto widgetu si student může vyzkoušet měnit rozsah frekvenčního filtru pomocí parametru radius (poloměr) a sledovat vliv na výstupní obrázek získaný pomocí 2D IFFT. Dále může měnit jeho typ (dolní/horní - v obrázku 4.5 jsou zde uvedeny oba najednou), případně si načíst jiný vstupní obrázek. Zde byla jako vstup záměrně vybrána Rubikova kostka, která má ve frekvenčním spektru jasně viditelné vzory. Těmi jsou vysoké frekvence zobrazené jako paprsky, které se ve spektru nacházejí pravě kvůli pravidelným hranách kostky.

Velmi podobně vypadá poslední podkapitola tohoto cvičení, kde se tak jako v 4.2.1 pracuje s jádry (masky). Místo přímé aplikace jednoduchého filtru, který nastaví vybrané frekvence, se provede operace konvoluce s vybraným jádrem. Ta je ve frekvenční oblasti definována jako Hadamardův součin (element-wise product). Jádro je tedy zarovnáno na velikost vstupního obrazu a převedeno do frekvenčního spektra. Na obrázek i jádro je pro konzistenci v notebooku nejprve aplikována operace posuvu `fftshift`, a to než se předají implementované funkci `conv_fft2` (funkce posuvu se může aplikovat i před samotnou fft). Zde budeme sledovat vliv na spektrum vstupu a vliv na výstup op2t získaný pomocí 2D inverzní rychlé Fourierovy transformace, která byla aplikována na vypočtené spektrum.




Obrázek 4.6: Filtrace ve frekvenční doméně pomocí jádra (gaussovské rozostření, $\sigma = 3$)

```python
def conv_fft2(x, h):
    """ 2D konvoluce pomocí FFT. """
    X = np.fft.fft2(x)  # obraz
    H = np.fft.fft2(h)  # jádro

    product = X * H  # Element-wise product ve frekvenční doméně
    return np.fft.ifft2(product)  # vyfiltrovaný obraz
```

## 4.3 Souborové odhady

Tato část cvičení pokrývá třetinu základní práce s náhodnými signály (neboli náhodnými procesy), a to souborové odhady. Podkapitola této práce je čtenáři prezentována jako kdyby se jednalo o samotný Jupyter notebook. Prolíná se zde narativní text, rovnice, kód buněk a výstup provedeného kódu. Chybí zde pouze příkazy pro vykreslování grafů.

Náhodné signály jsou signály, které nemají deterministickou hodnotu, můžeme pracovat jen s jejich pravděpodobnostním popisem. Teoreticky je náhodný signál reprezentován jako množina (soubor) realizací. Každá realizace představuje možný průběh signálu. Pro náhodné signály je užitečné umět spočítat různé statistiky, které je popisují.

### 4.3.1 Generování náhodného procesu

Náhodný proces bude gaussovský šum se střední hodnotou 0 a směrodatnou odchylkou 5, který necháme projít filtrem s přenosovou funkcí:

$$H(z) = \frac{1}{1 - 1.1314z^{-1} + 0.6400z^{-2}}$$

Vygenerujeme $\Omega = 10000$ realizací tohoto náhodného procesu po $N = 200$ vzorcích. Uložíme je do matice `ksi`.

```
omega = 10000
N = 200

# Vygenerování náhodného procesu:
mean = 0
std = 5
orig = np.random.normal(mean, std, (omega, N))

# Filtrace:
b = [1, 0, 0]
a = [1, -1.1314, 0.6400]
ksi = lfilter(b, a, orig)

ksi.shape
```

```
(10000, 200)
```

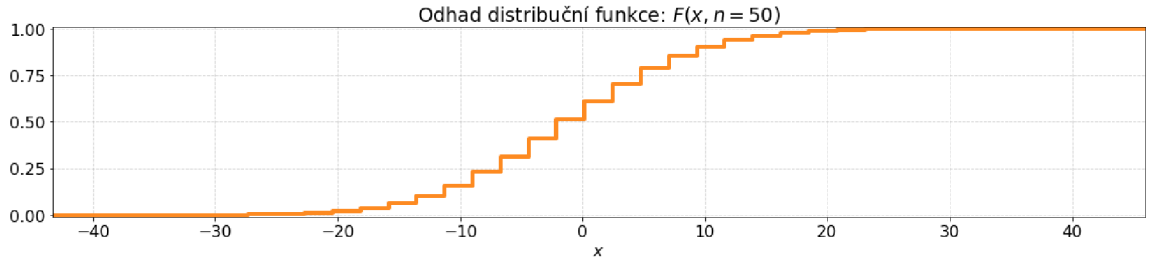
### 4.3.2 Odhad distribuční funkce

Souborové odhady budeme počítat pro určitý čas $n$, v našem případě $n = 50$. Distribuční funkce $F(x, n)$ v zásadě odpovídá na otázku *jak je pravděpodobné, že hodnota mého signálu pro vzorek $n$ bude menší než nějaká hodnota $x$.* Její hodnotu pro určité $x$ tedy odhadneme tak, že spočítáme v kolika realizacích byla hodnota signálu v čase $n$ menší než $x$ a tento počet podělíme celkovým počtem realizací. Odhad funkce spočítáme pro $x$, které se pohybují mezi minimální a maximální hodnotou našeho signálu v daném čase (pro menší $x$ je distribuční funkce nulová, pro větší $x$ má hodnotu 1). Spojitý průběh funkce aproximujeme pomocí 40 hodnot, pravidelně rozmístěných mezi minimem a maximem.

```
xmin = np.min(ksi)
xmax = np.max(ksi)

n_aprx = 40
x = np.linspace(xmin, xmax, n_aprx)

n = 50  # čas, pro který odhadujeme CDF
Fx = np.zeros(x.shape)
for i in range(n_aprx):
    Fx[i] = np.sum(ksi[:, n] < x[i]) / omega
```

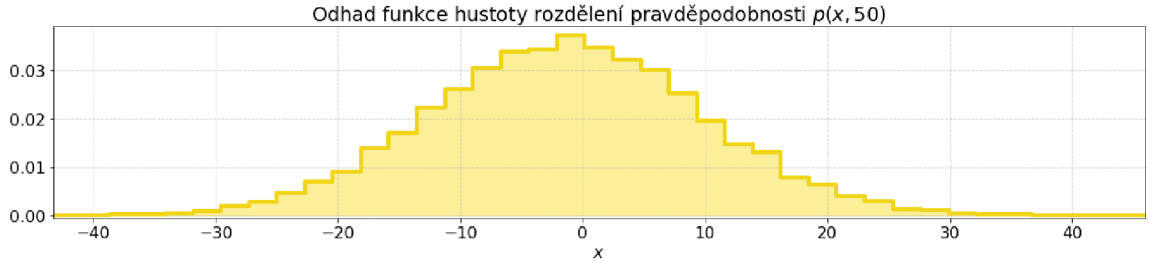



### 4.3.3 Odhad funkce hustoty rozdělení pravděpodobnosti

Funkce hustoty rozdělení pravděpodobnosti $p(x, n)$ je derivací distribuční funkce $F(x, n)$. Její hodnoty jsou nezáporné a platí, že integrál této funkce přes interval $<a, b>$ odpovídá pravděpodobnosti, že hodnota signálu v čase $n$ je v tomto intervalu. Integrál přes celou funkci hustoty rozdělení pravděpodobnosti (od $-\infty$ po $\infty$) by tedy měl být roven 1.

Souborový odhad pro čas $n = 50$ můžeme udělat pomocí histogramu. Budeme opět uvažovat $x$ z intervalu od minima po maximum našeho signálu a opět provedeme aproximaci funkce, rozdělením intervalu do diskrétních úseků. Pro každý úsek spočítáme, v kolika realizacích hodnota signálu v čase $n$ spadá do daného úseku. Tímto získáme histogram hodnot - v Pythonu za nás tento výpočet udělá funkce `numpy.histogram`. Abychom z histogramu dostali funkci $p(x, n)$, musíme jej podělit počtem realizací a velikostí jednoho úseku.

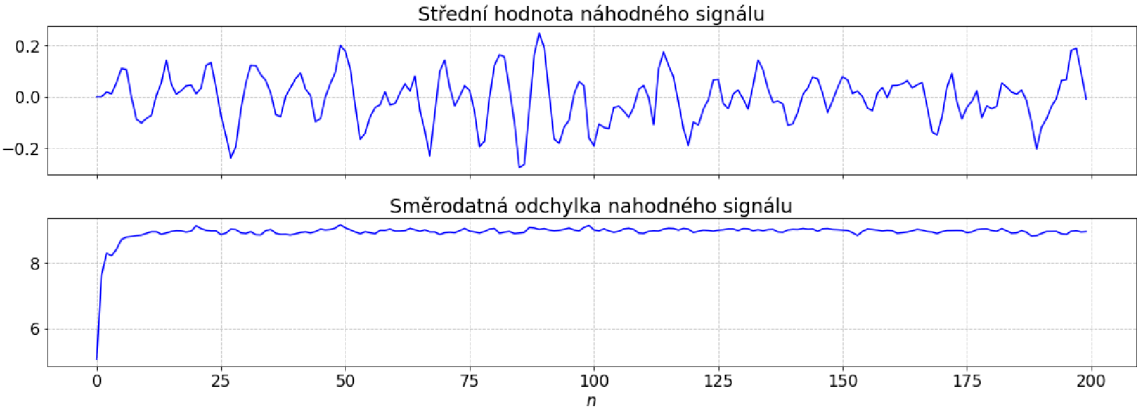
```
binsize = np.abs(x[1] - x[0])
hist, _ = np.histogram(ksi[:, n], n_aprx)
px = hist / omega / binsize
```



### 4.3.4 Odhad střední hodnoty a směrodatné odchylky v čase n

Souborový odhad střední hodnoty a směrodatné odchylky spočítáme pro každý čas $n$ a vykreslíme si je v závislosti na čase.

```
mean_n = np.mean(ksi, axis=0)
std_n = np.std(ksi, axis=0)
```




### 4.3.5 PDF mezi dvěma časy, autokorelační koeficienty

Funkce hustoty rozdělení pravděpodobnosti (PDF) mezi dvěma časy $p(x_1, x_2, n_1, n_2)$ popisuje vztah hodnot signálu v časech $n_1$ a $n_2$. Integrál této sdružené funkce přes intervaly $x_1 \in < a_1, b_1 >$, $x_2 \in < a_2, b_2 >$ odpovídá pravděpodobnosti, že hodnota signálu v čase $n_1$ spadá do intervalu $< a_1, b_1 >$ a zároveň hodnota $x_2$ spadá do intervalu $< a_2, b_2 >$. Odhad této funkce můžeme opět udělat pomocí histogramu - tentokrát 2D histogramu.

V Pythonu jej spočítá funkce `numpy.histogram2d`. Výsledný histogram můžeme opět vydělit počtem realizací a velikostí jednoho úseku (tentokrát 2D úseku), abychom získali odhad funkce $p(x_1, x_2, n_1, n_2)$. Druhá možnost je použít parametr `normed=True`, který histogram vynormalizuje za nás.

Autokorelační koeficient $R(n_1, n_2)$ udává míru toho, jak je *signál sám sobě podobný mezi časy* $n_1$ *a* $n_2$. Vypočteme jej z funkce hustoty rozdělení pravděpodobnosti jako:

$$R(n_1, n_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1 x_2 p(x_1, x_2, n_1, n_2) dx_1 dx_2.$$

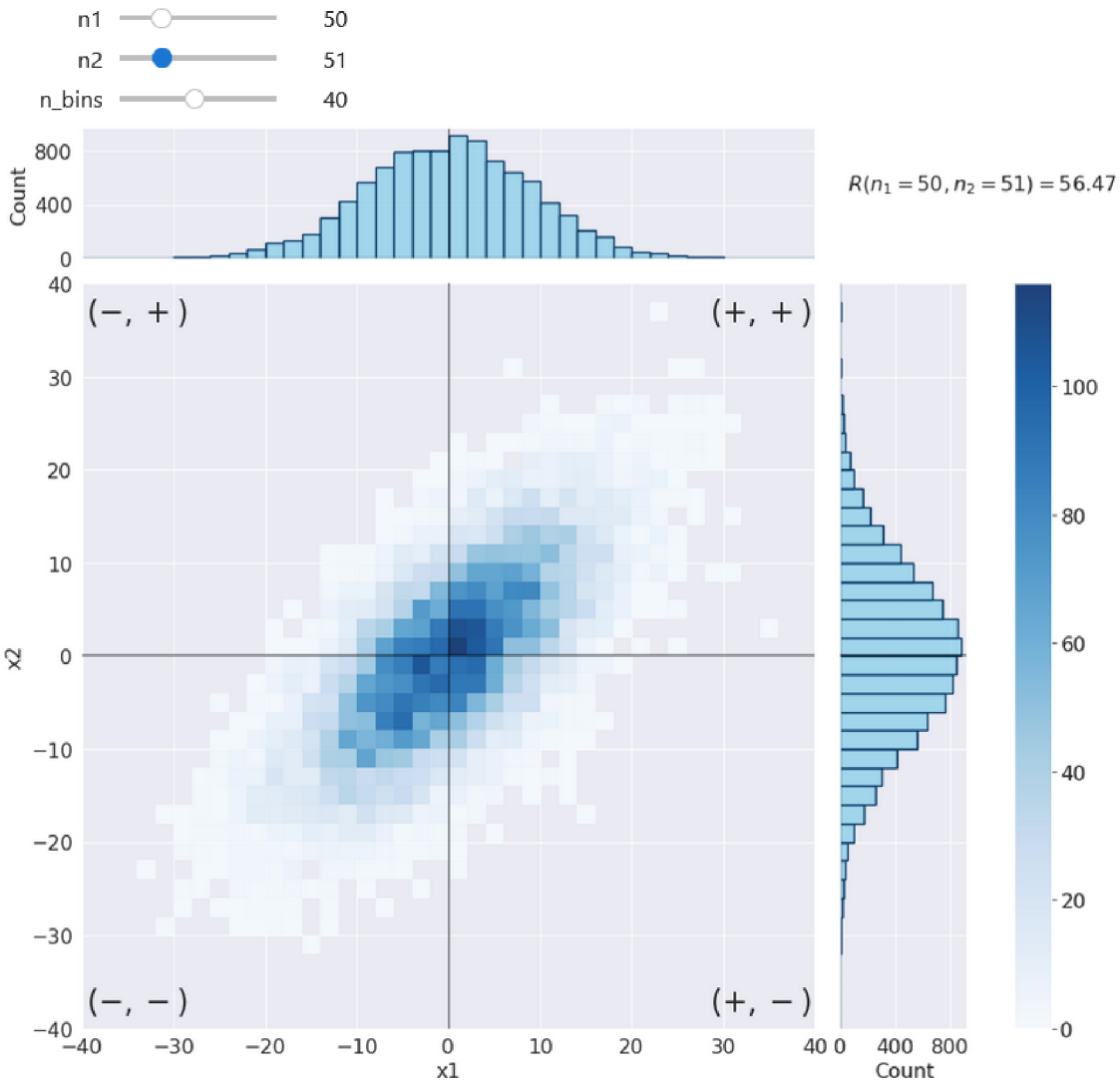
Prakticky tento integrál v Pythonu můžeme spočítat následovně:

- $p(x_1, x_2, n_1, n_2)$ spočítáme pomocí 2D histogramu - výsledkem bude matice hodnot
- vytvoříme matici součinů středu jednotlivých úseků $x_1$, $x_2$, nad vnějšího produktu vektorů těchto středů
- tyto dvě matice prvek po prvku vynásobíme
- samotný integrál pak získáme jako součet hodnot v matici vynásobený obsahem jednotlivých 2D úseků

Na obrázku 4.7 je zachycen interaktivní graf, kde si student může zobrazit různé typy korelací na reálných náhodných signálech volbou časů $n1$ a $n2$. V grafu nejsou uvedeny PDF, ale histogramy s četností.

```python
def acorr_coef(ksi, n_bins=40, n1=50, n2=51):
    """ Vypočte hodnotu autokorelačního koeficientu pro časy n1 a n2. """
    px1x2, x1_edges, x2_edges = np.histogram2d(ksi[:, n1], ksi[:, n2],
                                               n_bins, normed=True)
    # x1 * x2
    bin_centers_x1 = x1_edges[:-1] + (x1_edges[1:] - x1_edges[:-1]) / 2
    bin_centers_x2 = x2_edges[:-1] + (x2_edges[1:] - x2_edges[:-1]) / 2
    x1x2 = np.outer(bin_centers_x1, bin_centers_x2)

    binsize = abs(x1_edges[0] - x1_edges[1]) * abs(x2_edges[0] - x2_edges[1])
    # auto-correlation coefficient
    R = np.sum(x1x2 * px1x2 * binsize)
    return R
```




Obrázek 4.7: Interaktivní 2D Histogram

## 4.4 Uniformní kvantování

Tato část se nachází v posledním cvičení (Vzorkování a kvantování), které se nejprve zabývá vzorkováním, rekonstrukcí signálu a aliasingem. Věnuje se minimálním základům kvantování signálu a demonstracím na příkladech.

Nejběžnější ze všech skalárních kvantifikátorů je uniformní kvantifikátor, někdy také známý jako "lineární" kvantizátor, protože jeho schodovitá vstupní-výstupní odezva leží na přímce (se strmostí rovné jedničce). Všechna kvantování jsou ve správném slova smyslu ze své podstaty **nelineární** operace. Známé operace jako `floor` (ořez – celočíselné vyjádření, tj. bez hodnoty za desetinnou čárkou) nebo zaokrouhlení v aproximování reálných čísel jsou příklady uniformního kvantování. Digitální teploměr je dalším takovým příkladem. Většina univerzálních analogově digitálních převodníků (A/D) také využívá uniformního kvantování [5].

Předpokládejme, že hodnoty našeho signálu se mohou měnit v intervalu $[a,\ b]$, potom kvantifikační funkce bude mít tvar

$$x_Q = \left\lfloor \frac{x-a}{b-a} \cdot \left(2^N - 1\right) + \frac{1}{2} \right\rfloor \cdot \frac{b-a}{2^N - 1} + a,$$

kde N je počet bitů, na kterých bude uložena vzorkovaná hodnota a hranaté závorky reprezentují funkci `floor` (vysvětlena výše). Signál je rozdělen do N intervalů, které jsou rovnoměrně rozloženy. Tato kvantizační funkce pracuje následovně:

- Signál je převeden do intervalu $[0,\ 1]$
- Naškálován do rozsahu $\left[0,\ 2^N - 1\right]$
- Ořezem desetinné čárky (funkce `floor`) je následně vyjádřen celočíselně
- Kvantovaný signál je převeden zpět do intervalu $[a,\ b]$

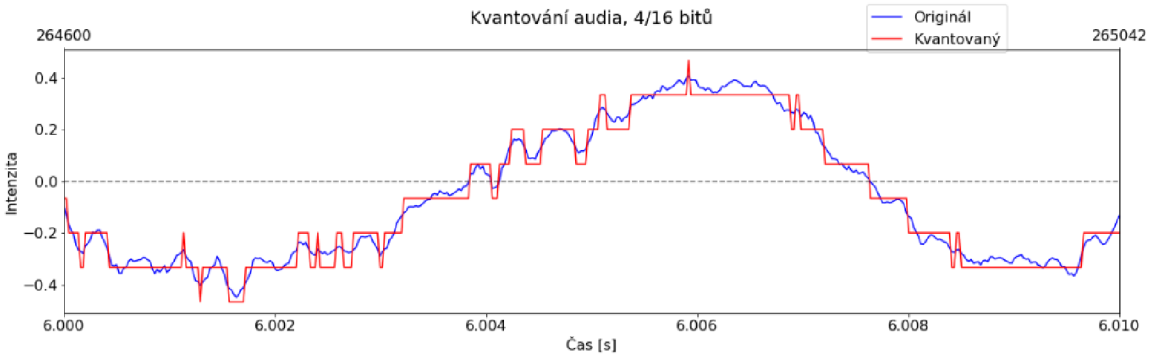
V Pythonu tuto funkci můžeme implementovat například takto:

```
def uf_quant(x: np.ndarray, a: float, b: float, N: int) -> np.ndarray:
    delta = (b - a)/(2**N - 1)
    return np.floor((x - a)/delta + 1/2) * delta  + a
```

### 4.4.1 Audio signál

Ve cvičení je uniformní kvantování nejprve demonstrováno na audio signálu s jedním zvukovým kanálem, kde jeden vzorek může nabývat jedné z $2^{16}$ hodnot z intervalu -1 až 1. Pokud si zvolíme parametr $N = 4$, kde $N$ bude udávat počet bitů naší kvantizační funkce, dostaneme $2^4 = 16$ různých hodnot kvantizačních hladin, které od sebe budou stejně vzdálené (jak je uvedeno výše), v tomto případě $2/(2^4 - 1) = 0.1\overline{3}$. Hodnota 2 je délka intervalu hodnot, kterých může vzorek zvukového signálu nabývat. Mezi hodnotami -0.2 a 0.2 si můžeme na obrázku 4.8 všimnout čtyř kvantizačních hladin, a to odpovídá $0.4 = (4-1) \times 0.1\overline{3}$.

```
uf_quant(x=audio, a=-1, b=1, N=4)
```




Obrázek 4.8: Kvantování audio signálu

Obrázek 4.8 je opět snímek obrazovky interaktivního widgetu, s nastavitelným počtem bitů kvantifikační funkce, posuvným začátkem a rozsahem viditelného intervalu signálu. V tomto případě je pro uživatele hlavním porovnáním vstupu s výstupem **poslech**. Při nízkém $N$ je zřetelně slyšet zkreslení signálu (kvantizační šum) způsobené zaokrouhlováním na kvantizační hladiny. Pokud by se uživateli nepodařilo přehrát výstup přímo v notebooku – což bohužel bývá častou chybou, pak může využít alternativu a spustit připravenou buňku, která výstupy ukládá do vhodně pojmenovaných souborů. Tyto wav soubory si pak může přehrát v nějakém externím programu mimo notebook.

### 4.4.2 Obraz

Vybrané obrázky pro demonstraci kvantování 2D signálů obsahují hodnoty složek, pixelů od 0 až po 255. Budeme tedy volat funkci `uf_quant` s parametry:

```
uf_quant(x=image, a=0, b=255, N=pocet_bitu)
```

Funkce bude kvantovat pixel po pixelu, a pokud má více barevných složek, tak každou zvlášť. Dá se tedy říct, že se k našemu 2D (případně pokud je obraz barevný, tak 3D) signálu reprezentujícímu obraz chová jako k 1D poli, tak jako v našem předchozím příkladě s mono audiem.

Demonstraci kvantování na obrazu opět obstarává ipywidget. Uživatel si může vybrat z několika načtených obrázku (včetně samotného spektra stupně šedi) a volit, na kolik bitů chce výstup kvantovat, případně si také zobrazit rozdíl vstupu s výstupem. Dále se mu zobrazují relevantní metriky jako je paměťová náročnost výstupu oproti vstupu a průměrná chyba na jeden pixel. Na obrázku 4.9 je ukázka jednoho z možných zobrazení minimálního výstupu.

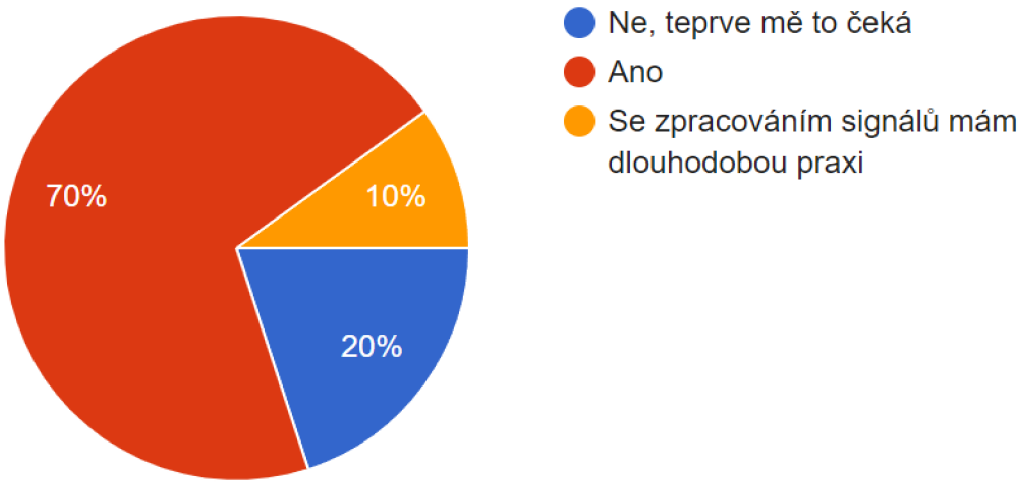

Obrázek 4.9: Kvantování obrazu (statická část výstupu generovaného interaktivním widgetem)

# Kapitola 5

# Uživatelské testování

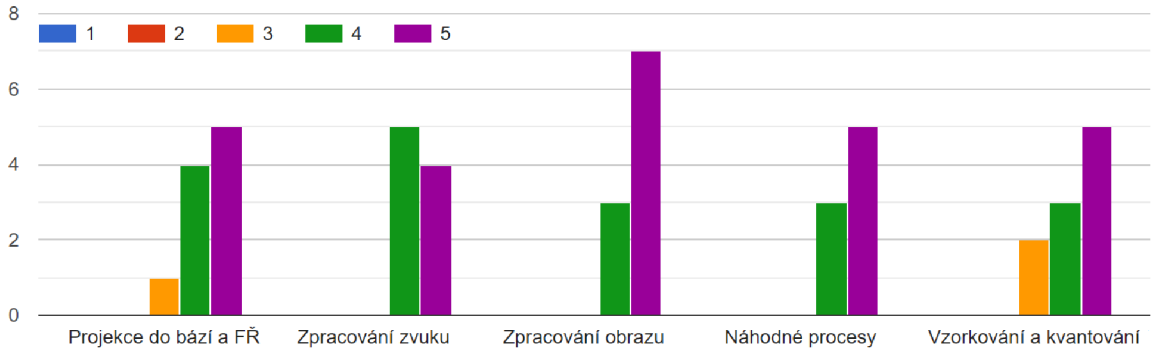
Uživatelské testování proběhlo na malé skupince 10 studentů. Na obrázku 5.1 je zobrazeno procentuální rozdělení studentů podle jejich předešlých zkušeností se zpracováním signálu, kteří si vytvořené materiály prošli a vyzkoušeli. Zpětnou vazbu jsem od uživatelů získal prostřednictvím dotazníku, který je uvedený v příloze B. Instrukce k instalaci prostředí studenti dostali v souboru `README.md`, kde doporučeným postupem bylo stáhnutí Pythonu distribuce Anaconda, aktualizace knihovny `matplotlib` na verzi 3.5.1 a vyšší (další instrukce jsou uvedeny souboru `README.md` na přiloženém paměťovém disku).

S výsledky uživatelského testování jsem spokojen, protože ohlasy na vytvořené notebooky byly pozitivní. Většina uživatelů uvedla v připomínkách některé části cvičení, které se jim opravdu líbili a byly pro ně přínosné tím, že získali nové poznatky. Dále byly uvedeny připomínky pro přidání detailnějšího popisu některých demonstračních příkladů (hlavně pro cvičení *Vzorkování a kvantování*) a k nim vztahující teorie. Dvěma uživatelům se nepovedlo spustit některé buňky s kódem interaktivního widgetu, konkrétně v notebooku *Zpracování zvuku*. Polovina studentů pracovala v IDE VSCode.

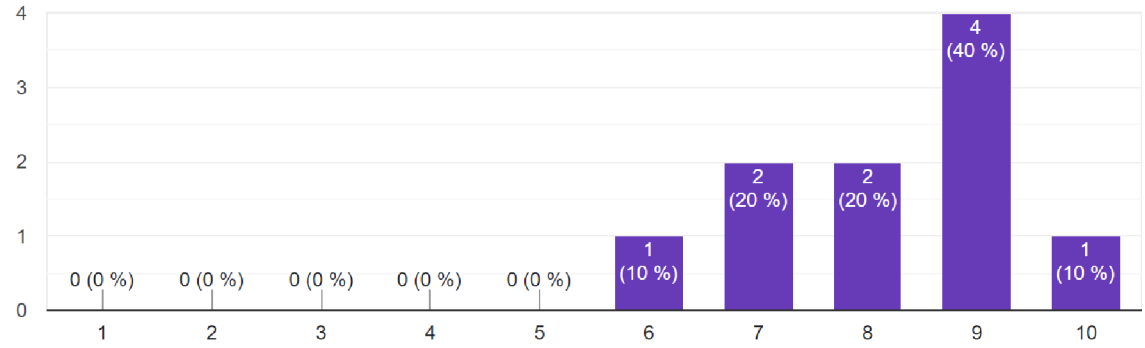



Obrázek 5.1: Graf, který znázorňuje procentuální zastoupení studentů podle toho, zda už absolvovali předmět Signály a Systémy nebo mají širší znalosti.

Následují ještě dva grafy vytvořené ze získaných informací prostřednictvím dotazníku.




Obrázek 5.2: Graf hodnocení notebooků, poskytnutého studenty, kteří si je vyzkoušeli (1 – vůbec jsem nepochopil dané téma, 5 – téma v notebooku je dobře zpracováno)




Obrázek 5.3: Graf hodnocení přehlednosti a orientace ve výstupech, interaktivních widgetech, kódu atd. (1 neintuitivní, 10 intuitivní)

# Kapitola 6

# Závěr

Tato práce se věnuje základům z oblasti číslicového zpracování signálů. Jejím cílem bylo vytvořit podpůrný materiál pro samostudium podle dosavadně vytvořených. Zaměřil jsem se na cvičení, která v Pythonu nebyla zatím implementována a ty, které byla doplnil a rozšířil například o některé interaktivní prvky nebo obsah navíc.

První cvičení bylo zaměřen na zopakování teorie o vektorových bází a Fourierovu řadu. Jako jediné je rozděleno jen do krátkých číslovaných příkladů. Ostatní notebooky do krátkých kapitol a podkapitol. Druhé cvičení se zabývá zpracování audio signálu, jeho filtrací a spektrální analýzou. Další cvičení představuje základní techniky zpracování obrazu a použití diskrétních 2D transformací. Čtvrtý notebook demonstruje jak pracovat s náhodnými procesy, a to nejprve na souborových odhadech a následně na časových odhadech. V poslední cvičení je zaměřeno na vzorkování a kvantování signálů. Struktura všech notebooků je uvedena v příloze A níže.

Implementaci v jazyce Python je členěna do spustitelných buněk těchto notebooků, kde ji doprovází původní a upravená teorie. Dále byla pomocí knihovny ipywidgets vytvořena interaktivní rozšíření, aby si studenti mohli zobrazit mnoho různých příkladů pro hlubší pochopení probírané látky. Vytvořené notebooky si vyzkoušela testovací skupinka studentů, která následně vyplnila dotazník se zpětnou vazbou, který je uveden v příloze B. Toto testování dopadlo pozitivně a sepsané připomínky pro vylepšení mě přivedly k minimálním dodatečným úpravám výsledných notebooků.

V průběhu práce jsem si zopakoval poznatky z bakalářského předmětu ISS a zároveň obohatil svoje vědomosti z této oblasti. Osvojil jsem si práci s programovacím jazykem Python, některé jeho knihovny a interaktivní prostředí Jupyter notebook. V textu této práce je popsán jen vybraný zlomek obsahu všech cvičení. Navíc pokrývají jen základy z oblasti signálů a systémů. Případným rozšířením práce by proto mohlo být zpracování pokročilejších témat. Dále přidat vizuální interakce pomocí pokročilejších knihoven nebo vytvořit vhodné animace, které by se mohly přehrávat přímo v notebooku.

# Literatura


[1] *What Is Python? Executive Summary* [online]. Guido van Rossum, 2001 [cit. 2022-7-20]. Dostupné z: https://www.python.org/doc/essays/blurb/.

[2] *What is the Jupyter Notebook* [online]. Jupyter Team, 2015 [cit. 2022-7-20]. Dostupné z: https://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/What%20is%20the%20Jupyter%20Notebook.html.

[3] *Jupyter Widgets* [online]. Project Jupyter Revision, 2017 [cit. 2022-7-20]. Dostupné z: https://ipywidgets.readthedocs.io/en/stable/.

[4] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1. vyd. 1986, PAMI-8, č. 6, s. 679–698.

[5] Gersho, A. a Gray, R. M. *Vector quantization and signal compression*. 1. vyd. Kluwer Academic Publishers, 1992. 151 s. ISBN 978-1-4613-6612-6.

[6] Hubeika, V. a Černocký, J. *Image Processing* [online]. FIT VUT v Brně, září 2009 [cit. 2022-7-20]. Dostupné z: http://www.fit.vutbr.cz/~ihubeika/ISS/lect/2d_en.pdf.

# Příloha A

# Přehled podkapitol v noteboocích

1. Prostředí Jupyter (Python)
2. Báze a promítání do nich (opakování)
    1. Báze a promítnání
        1.1. Příklady (1 až 6)
    2. Fourierova řada
        2.1. Příklady (7 až 13)
3. Zpracování audia
    1. Načtení a základní analýza
        1.1. Jedno spektrum
        1.2. Spektrogram
    2. Filtrace (teorie)
        2.1. Analýza filtru
    3. Několik příkladů filtru
        3.1. Interaktivní ukázky
    4. Třetino-oktávový ekvalizér
        4.1. Rozklad a složení signálu
        4.2. Experimentování s ekvalizérem
    5. Zvukové efekty (nepovinné)
        5.1. Klasifikace
        5.2. Nelineární zpracování
        5.3. Delay
        5.4. Modulace
        5.5. Časově proměnné filtry
        5.6. Prostorové efekty

4. Základy zpracování obrazu
    1. Načtení a vykreslování obrázku
    2. Základní techniky zpracování obrazu
        2.1. Inverze barev
        2.2. Jednoduchá detekce hran
        2.3. Thresholding
    3. Histogram (ekvalizace)
    4. Lineární filtry
    5. Komprese à la JPEG pomocí DCT
    6. Chyba v obraze
    7. Filtrování ve frekvenční oblasti (nepovinné)
5. Náhodné procesy - základy
    1. Souborové odhady
        1.1. Generování náhodného procesu
        1.2. Odhad distribuční funkce
        1.3. Odhad funkce hustoty rozdělení pravděpodobnosti
        1.4. Odhad střední hodnoty a směrodatné odchylky v čase n
        1.5. Odhad autokorelačních koeficientů
    2. Časové odhady
        2.1. Generování náhodného procesu
        2.2. Odhad distribuční funkce a PDF
        2.3. Odhad střední hodnoty a směrodatné odchylky
        2.4. Odhad autokorelačních koeficientů
    3. Odhady spektra
        3.1. Spektrální hustota výkonu
        3.2. Průchod náhodného signálu lineárním systémem
6. Vzorkování, aliasing a kvantování
    1. Vytvoření analogového signálu
    2. Analýza našeho spojitého signálu
    3. Vzorkování
        3.1. Rekonstrukce navzorkovaného signálu
    4. Aliasing & Anti-aliasing
        4.1. Experimentování s aliasingem
        4.2. Aliasing - 2D signál
    5. Kvantování
        5.1. Kvantování audia
        5.2. Kvantování obrazu

# Příloha B

# Dotazník pro uživatelské testování

# Uživatelské testování vytvořených jupyter notebooků

Tento dotazník slouží jako zpětná vazba studentů, kteří si vyzkoušeli práci s interaktivními jupyter notebooky. Obsah těchto notebooků se věnuje tématům z oblasti zpracování signálu.

*Povinné pole

1. Absolvoval/a jste předmět ISS (Signály a Systémy) nebo jiný předmět, zaměřený na signály? *

   *Označte jen jednu elipsu.*

   - ◯ Ne, teprve mě to čeká
   - ◯ Ano
   - ◯ Se zpracováním signálů mám dlouhodobou praxi

2. Jaké máte znalosti jazyka Python? *

   *Označte jen jednu elipsu.*

   - ◯ Minimální
   - ◯ Ovládám základy jazyka Python
   - ◯ Pokročilé

3. Jaké prostředí jste si pro práci s materiály zvolil? *

   *Označte jen jednu elipsu.*

   - ◯ Jupyter Notebook
   - ◯ JupyterLab
   - ◯ VSCode
   - ◯ Jiné: _____

4. Zde ohodnoťte notebooky, které jste prošel/a.

(1 - vůbec jsem nepochopil dané téma, 5 - téma v notebooku je dobře zpracováno)

*Označte jen jednu elipsu na každém řádku.*

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Projekce do bází a FŘ** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Zpracování zvuku** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Zpracování obrazu** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Náhodné procesy** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Vzorkování a kvantování** | ◯ | ◯ | ◯ | ◯ | ◯ |

5. Zde ohodnoťte přehlednost v noteboocích (orientace v zobrazených výstupech, ovládání interakce apod.). *

*Označte jen jednu elipsu.*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Neintuitivní | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | Intuitivní |

6. Jak dlouho Vám trvala práce s notebooky, které jste si prošel? *

_____

*Příklad: 8:30*

7. Připomínky. *

Zde prosím uveďte jakých nedostatků jste si v noteboocích všimli, uveďte nápady pro vylepšení a podobně.

_____

_____

_____

_____

_____