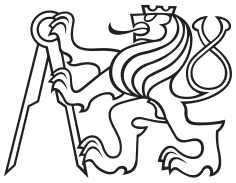
Disertační práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Zobrazení Waterfall spektra na PC

Patrik Petrýdes

Vedoucí: doc. Dr. Ing. Pavel Kovář
Obor: Letectví a kosmonautika
Studijní program: Avionika
Leden 2022

Poděkování

Chtěl bych poděkovat svému vedoucímu práce, doc. Dr. Ing. Pavlu Kovářovi doc. za vstřícnost a ochotu při konzultacích a vypracování diplomové práce..

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 03. ledna 2022

Abstrakt

Cílem této práce je vytvořit program, který přijímá a zobrazuje amplitudové spektrum ve formě waterfall na počítači. V první části práce je teorie, která shrnuje pojmy spojené s waterfall spektrem. Na začátku je velmi krátce popsáno, co znamená signál, systém a digitální zpracování signálu. Poté se postupně popíše, co je to Fourierova řada, Fourierova transformace a diskrétní Fourierova transformace. Z toho dále vychází popis rychlé Fourierovy transformace a s tím sovisející výkonové a amplitudové spektrum. Poté už je popsáno, co je to spektrogram a waterfall graf. Druhá část práce se zaobírá praktickým návrhem zadaného programu. Na začátku jsou popsané požadavky na program a jaký programovací jazyk byl použit. Dále jsou popsány hlavní použité moduly jazyka a jak se dá program využít. Nakonec je vysvětleno, jak program funguje, jeho funkce a celkové zhodnocení.

Klíčová slova: Fourierova řada, Fourierova transformace, FT, diskrétní Fourierova transformace, DFT, rychlá Fourierova transformace, FFT, amplitudové spektrum, výkonové spektrum, spektrogram, waterfall graf, waterfall spektrum

Vedoucí: doc. Dr. Ing. Pavel Kovář
Katedra radioelektroniky,
Technická 2,
Praha 6

Abstract

The aim of this thesis is to create a program that receives and displays amplitude spectrum in the form of a waterfall plot on a computer. The first part of the thesis is theory, that summarizes the concepts associated with the waterfall spectrum. At the beginning, it is very briefly described what the signal, system and digital signal processing is. Then the Fourier series, the Fourier transform and the discrete Fourier transform are described step by step. This is used as the basis for the description of the fast Fourier transform and the associated power and amplitude spectrum. It is then described what a spectrogram and waterfall plot are. The second part of the thesis deals with the practical design of the assigned program. At the beginning, the requirements for the program and what programming language was used are described. The main language modules used and how the program can be used are described afterward. Finally, it explains how the program works, its functions and overall evaluation of it.

Keywords: Fourier series, Fourier transform, FT, discrete Fourier transform, DFT, fast Fourier transform, FFT, amplitude spectrum, power spectrum, spectrogram, waterfall plot, waterfall spectrum

Title translation: Display of Waterfall spectrum on PC

Obsah

Obrázky

Kapitola 1

Úvod

Tato práce si klade za cíl shrnout pojmy související s waterfall spektrem a hlavně vytvořit jasný a funkční program pro jeho prohlížení. Protože je hlavním cílem práce vytvoření programu pro zobrazení amplitudového spektra, nebylo cílem popisovat do teorii spektrální analýzy do detailů. Práce se tedy snaží vysvětli všechny pojmy krátce, jednoduše a bez složité matematiky.

Příští, tedy druhá, kapitola má cíl krátce a pochopitelně vysvětlit některé pojmy, které budou potřeba k pochopení dalších částí práce. Tedy zpracování signálu, Fourierovu řadu a Fourierovu transformaci a diskrétní Fourierovu transformaci.

Protože ne každý zná pojem waterfall graf nebo spektrogram, budou v další kapitole jsou vysvětleny další pojmy, které jsou důležité pro pochopení spektrální analýzy a s tím i waterfall grafu. V této kapitole bude popsána hlavně rychlá Fourierova transformace a poté výkonové a amplitudové spektrum. Z toho už se dá snadno vysvětlit, co to waterfall spektrum je.

Druhá část práce se zabývá samotným návrhem programu. Tedy popsání celého programu, jeho využití a zhodnocení.

Část I

Teretická část

Kapitola 2

Zpracování signálu

V této kapitole se budu snažit jednoduše vysvětlit různé pojmy, které souvisí s Waterfall spektrem. Na začátku jednoduše popíši, co to vlastně je signál a jeho zpracování. Protože nás zajímá hlavně frekvenční analýza signálu, popíši v kapitole Fourierovu řadu a Fourierovy transformace.

2.1 Signál, systém a zpracování signálu

Signál je definován jako jakákoli fyzikální veličina, která se mění v závislosti s časem, prostorem nebo nějakou jinou nezávislou proměnnou nebo proměnnýma. Matematicky popisujeme signál jako funkci jedné nebo více nezávislých proměnných.

Generování signálu je obvykle spojeno se systémem, který reaguje na nějaký podnět. Takový podnět v kombinaci se systémem se nazývá zdroj signálu.

Systém může být definován například jako fyzické zařízení, které provádí operace na signálu. Když prochází signál systémem, jako například při filtrování, říkáme, že jsme tento signál zpracovali. Obecně je systém charakterizován podle typu operace, kterou na signálu provádí, ale všechny tyto operace jsou obvykle označované jako zpracování signálu.

Pro tuto práci je důležité rozšířit definici systému na neobsahují pouze

fyzická zařízení, ale i softwarové realizace. Při digitálním zpracování signálů na počítači se provedené operace na signálu skládají z řady matematických operací specifikovaných v softwarovém programu. Máme tedy digitální systém pro zpracování signálu realizovaný v softwaru. Alternativně může být digitální zpracování signálu prováděno digitálním hardwarem (logickými obvody) nakonfigurovaným tak, aby prováděl požadované operace. V takové realizaci máme fyzické zařízení, které provádí zadané operace. V širším slova smyslu lze digitální systém implementovat jako kombinaci digitálního hardwaru a softwaru, kde každý z nich provádí vlastní sadu zadaných operací. [1]

2.2 Digitální zpracování signálu

Většina signálů, se kterými se setkáváme ve vědě a technice, je analogové povahy. To znamená, že signály jsou funkcemi spojité proměnné, jako je čas nebo prostor, a obvykle nabývají hodnot ve spojitém rozsahu. Takové signály mohou být zpracovány přímo vhodnými analogovými systémy nebo násobiči frekvence za účelem změny jejich charakteristik nebo extrahování nějaké požadované informace. V takovém případě říkáme, že signál byl zpracován přímo ve své analogové podobě, jak je znázorněno na obrázku 2.1. Jak vstupní signál, tak i výstupní signály jsou v analogové formě. [1]

Obrázek 2.1: Analogové zpracování signálu. [1]

Digitální zpracování signálu poskytuje alternativní metodu pro zpracování analogového signálu. Tento proces je znázorněno na obrázku 2.2. K tomu provést zpracování digitálně je třeba mít rozhraní mezi analogovým signálem a digitálním procesorem. Toto rozhraní se nazývá analogově-digitální (A/D) převodník. Výstup z A/D převodníku je digitální signál, který je vhodný jako vstup do digitálního procesoru. [1]

Obrázek 2.2: Blokový diagram systému digitálního zpracování signálu. [1]

Procesorem digitálního signálu může být velký programovatelný počítač nebo malý mikroprocesor naprogramovaný tak, aby prováděl požadované operace na vstupním signálu. Tyto přístroje mohou být pevně nakonfigurované nebo volně programovatelné.

V aplikacích kde má být uveden digitální výstup z digitálního signálového procesoru k uživateli v analogové podobě, musíme poskytnout další rozhraní, které převádí signál z digitální oblasti do oblasti analogové. Takové rozhraní se nazývá digitálně-analogový (D/A) převodník. Tím je zajištěn signál k uživateli v analogové podobě, jak je znázorněno na blokovém schématu na obrázku 2.2.

Existují však i další praktické aplikace zahrnující analýzu signálu, kde požadovaná informace je přenášena v digitální formě a žádný D/A převodník není požadován. [1]

2.3 Fourierova řada

Fourierova řada slouží k rozkladu periodického signálu na součet sinusových funkcí. Fourierova řada je periodická funkce složená z harmonicky příbuzných sinusoid, kombinovaných váženým součtem. S vhodnými váhami lze provést jeden cyklus (nebo periodu) součtu pro aproximaci libovolné funkce v daném intervalu. Suma tedy jako taková je syntézou jiné funkce. Fourierova transformace v diskrétním čase, která bude popsána v další části práce, je příkladem Fourierovy řady. Proces odvozování vah, které popisují danou funkci, je formou Fourierovy analýzy.

Bylo objeveno mnoho různých přístupů k definování a chápání pojmu Fourierovy řady, z nichž všechny jsou ve vzájemném souladu, ale každý z nich zdůrazňuje jiné aspekty tématu. Některé z výkonnějších a elegantnějších přístupů jsou založeny na matematických nástrojích, které nebyly dostupné v době, kdy Fourier dokončil své původní dílo. Fourier původně definoval Fourierovy řady pro funkce s reálnými hodnotami a reálnými argumenty a jako základ pro rozklad používal funkce sinus a kosinus. Od té doby bylo definováno mnoho dalších Fourierových transformací, které rozšiřují původní myšlenku na další aplikace. Tato obecná oblast zkoumání se nyní někdy nazývá harmonická analýza. Fourierova řada však může být použita pouze pro periodické funkce. [11]

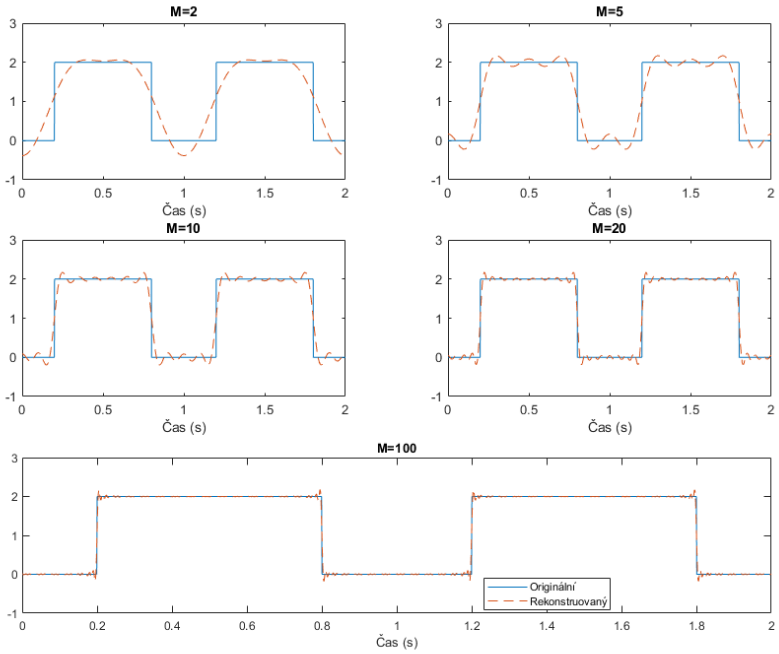
Pro pochopení funkce Fourierovy řady si můžeme uvést následující příklad. Jako první si můžeme ukázat, jak získat váhové koeficienty, které nám pomůžou rekonstruovat danou funkci pomocí součtu sinusových funkcí.

Pro daný libovolný periodický signál $x_p(t)$ s hlavní periodou T lze $x_p(t)$

rozvést takto: [2]

$$
\begin{aligned}
& x_p^M(t) = a_0 + \sum_{i=1}^{M} a_i cos\left(\omega t i\right) + \sum_{i=1}^{M} b_i sin\left(\omega t i\right), \\
& kde \quad a_0 = \frac{1}{T}\int_0^T x_p(t)dt, \\
& a_i = \frac{2}{T}\int_0^T x_p(t) cos\left(\omega t i\right) dt \\
& b_i = \frac{2}{T}\int_0^T x_p(t) sin\left(\omega t i\right) dt \\
& \omega = 2\pi f = \frac{2\pi}{T}
\end{aligned}
\tag{2.1}
$$

V tomto případě máme váhové koeficienty a_i a b_i, jak se M zvyšuje, váha faktorů se zmenšuje a jejich příspěvek k součtu bude menší. Proto, jak se M zvyšuje, se sumace přibližuje a blíže skutečnému periodickému signálu. Obrázek 2.3 ukazuje působení zvyšování M a jak je obdélníkový signál rekonstruován omezeným počtem sumací M. Jak je vidět z obrázku **??**, jak M stoupá rekonstruovaný signál ze součtu Fourierové řady konverguje k původnímu periodickému signálu. [2]



Obrázek 2.3: Součet Fourierovy řady pro $M = 2, 5, 10, 20$ signálu obdélníkového signálu ve dvou periodách. [2]

2.4 Fourierova transformace

Pojem Fourierovy řady lze zobecnit i na neperiodické signály k odvození frekvenční reprezentace pro tento typ signálu. Fourierova transformace je matematická transformace, která umožňuje odvodit frekvenční reprezentaci pro tyto signály. [2]

V matematice je Fourierova transformace (FT), transformace, která rozkládá funkce závisející na prostoru nebo čase na funkce v závisející na prostorové nebo časové frekvenci. Termín Fourierova transformace se vztahuje jak na reprezentaci ve frekvenční oblasti, tak na matematickou operaci, která asociuje reprezentaci ve frekvenční oblasti s funkcí prostoru nebo času. [12]

Signál musí splňovat určitá kritéria známá jako Dirichletovy podmínky před tím než lze aplikovat Fourierovu transformaci. Naštěstí mnoho signálů, se kterými pracujeme, tyto podmínky splňuje.

Fourierova transformace signálu $x(t)$ je definována jako: [13]

$$X(i\Omega) = \int_{-\infty}^{\infty} x(t)e^{-i\Omega t}dt \tag{2.2}$$

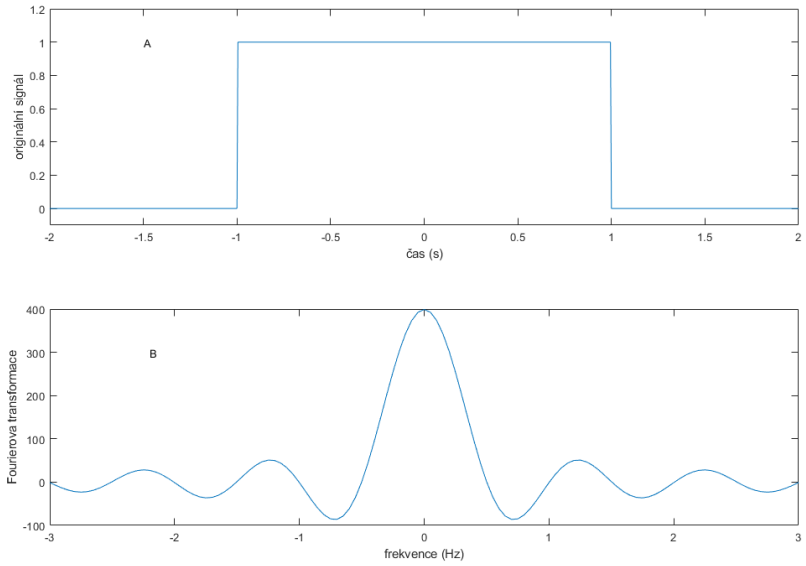
Je zajímavé, že Fourierova transformace je „invertibilní“, což znamená, že pokud máme kompletní informace o Fourierově transformaci signálu, můžeme znovu získat původní signál v časové oblasti, pomocí inverzní Fourierovy transformace. Inverzní Fourierova transformace je definována jako: [2]

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(i\Omega)e^{-i\Omega t}d\Omega \tag{2.3}$$

Představme si jeden pulz s omezenou šířkou, jak je znázorněno na obrázku 2.4a. Signál je prezentován v časové oblasti. Pokud je na tento signál aplikována Fourierova transformace, vznikne transformovaný signál, jaký je prezentován na obrázku 2.4b. Tento signál má speciální tvar, který je známý jako funkce sinc.

Z obrázku 2.4b je vidět, že většina energie signálu leží v intervalu $(-0.5, 0.5)Hz$ a mimo tento interval je energie signálu nízká. [2]

Další příklad může být Obrázek 2.5a, který ukazuje tlumící exponenciální



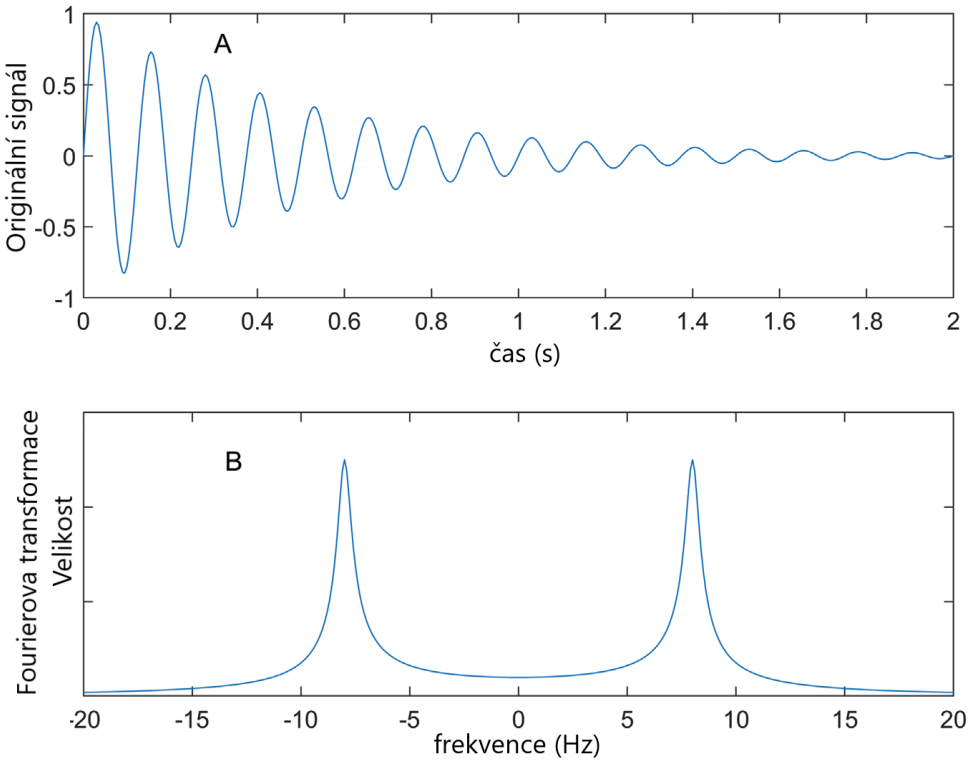
Obrázek 2.4: (a) Časová reprezentace jednoho pulzu v čase a (b) Fourierova transformace tohoto pulzu ve frekvenční oblasti [2]

signál. V tomto případě se jedná o volný indukční rozpad, který lze zaznamenat pomocí nukleární magnetické rezonanční spektroskopie. Obrázek 2.5b ukazuje velikost odpovídající Fourierovy transformace. Na rozdíl od časové reprezentace signálu, která je rozprostřena v určité době, je její frekvenční reprezentace značně zhuštěná do úzkého rozsahu frekvencí. [2]

Je zajímavé, že pokud manipulujeme se signálem v časové oblasti, manipulace se také odráží ve frekvenční oblasti, z hlediska velikost nebo fáze Fourierovy transformace, a naopak. Posun v čase se odráží ve fázi Fourierovy transformace a pokud se dva signály spolu sčítají v časové oblasti, Fourierova transformace součtu bude také součtem jejich Fourierových transformací. Pokud se však vynásobí Fourierovy transformace dvou signálů, odpovídá to konvoluci signálů v časové oblasti.

Jak je vidět z rovnic 2.2 a 2.3, Fourierova transformace a její inverzní funkce, mají podobné matematické formulace. Tato podobnost má za následek velmi zajímavou vlastnost Fouriery transformace nazývanou dualita mezi časovou a frekvenční doménou.

Fourierova transformace signálu $x(t)$ je $X(i\Omega)$ a lze tedy ukázat, že pro signál $X(t)$, který má podobu Fourierovy transformace $x(t)$ v časové oblasti, má jeho Fourierova transformace tvar $2\pi x(-\Omega)$. [2]



Obrázek 2.5: (a) Časová reprezentace signálu volného indukčního rozpadu (FID) a (b) signál Fourierova transformace signálu FID [2]

2.5 Vzorkování spojitého signálu

Počítače nám dnes umožňují využívat velmi výpočetně náročné procesy. Je zde však jeden problém, že nezáleží na tom, jak rychlý je počítač, časové rozlišení překódování a registrace signálu jsou omezené. Nejsme tedy schopni zaregistrovat signály pro nekonečně malé časové kroky. Naštěstí takové časové rozlišení nepotřebujeme a můžeme pracovat se spojitým časovým signálem v daném omezeném časovém rozlišení, pokud je signál spojitého času dostatečně rychle vzorkován. [2]

2.5.1 Diskrétní časový signál

Diskrétní časové signály jsou složeny ze sekvence vzorků signálu. Můžeme si tedy představit, že jsou stejné jako pro spojitý časový signál, jen odebírány ve stejně vzdálených intervalech. Pro diskrétní signál $x(n)$ lze tedy napsat: [2]

$$x(n) = x_c(nT_s) \tag{2.4}$$

Kde $x_c(t)$ je libovolný spojitý časový signál, u kterého ale máme pouze jeho odpovídající amplitudy v časech, které jsou celočíselnými násobky T_s, což je perioda vzorkování. Lze si představit, že v čase 0 získáme první vzorek $x_c(t)$, poté o T_s sekund později získáme druhý vzorek, o dalších T_s sekund později získáme třetí vzorek a tak dále. Matematicky řečeno, tento scénář, po určitém zjednodušení, odpovídá vynásobení $x_c(t)$ sérií ostrých impulsů proložených periodou T_s. Tento sled impulsů lze definovat jako: [2]

$$x_s(t) = s(t)x_c(t) \tag{2.5}$$

V této rovnici se $s(t)$ rovná 0 v libovolném časovém bodě kromě časů, které jsou celočíselnými násobky T_s a v těchto časových bodech se jeho energie rovná 1. Výsledkem násobení $s(t)$ a $x_c(t)$ je sled impulzů, jehož energie při každém z impulsů se rovná amplitudě $x_c(t)$ v odpovídajícím časovém bodě.

2.5.2 Vzorkování ve frekvenční oblasti

Fourierova transformace může být také definována pro signály v diskrétním čase. Její matematická formulace je podobná té, která byla uvedena pro spojité časové signály 2.2, ale pro diskrétní časové signály je třeba integraci nahradit běžným součtem. Fourierova transformace pro diskrétní časový signál $x(n)$ by tedy byla definována jako: [2]

$$X(e^{i\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-i\omega n} \tag{2.6}$$

Již bylo dříve zmíněno, jakákoli manipulace se signály v časové oblasti se bude nějakým způsobem odrážet v oblasti frekvenční. Dá se matematicky ověřit, že pokud je spojitý časový signál výsledkem násobení dvou signálů v časové oblasti, jeho Fourierova transformace by odpovídala konvoluci Fourierovy transformace těchto dvou signálů, a pokud je transformace prezentována v podobě úhlové frekvence, výsledky by byly škálovány podle $\frac{1}{2\pi}$. Tato vlastnost se nazývá konvoluční teorém Fourierovy transformace. Další vlastností je, že

Fourierova transformace sledu impulsů v časové oblasti bude také impulsním sledem ve frekvenční oblasti.

Jinou vlastností konvoluce je to, že konvoluce jakéhokoli signálu impulsem se rovná stejnému signálu, a pokud byl signál konvolvován posunutým impulsem, výsledek by se také posunul ve stejném rozsahu. Tedy Fourierova transformace $x_s(t)$, definována dříve, by byl součet posunutých verzí Fourierovy transformace $x_c(t)$. To je důležitá vlastnost, protože na základě tohoto vztahu můžeme dojít k závěru, že Fourierova transformace pro signály v diskrétním čase je jednoduše verze $X_s(i\Omega)$ v jiném měřítku, ve které se $\Omega T_s = \omega$. To znamená, že pro osu frekvence v diskrétním čase bude $\Omega = \Omega_s$ odpovídat $\omega = 2\pi$. Další vlastností je, že Fourierova transformace signálu v diskrétním čase je vždy periodická s periodou 2π. Tudíž pokud známe Fourierovu transformaci signálu v diskrétním čase v jedné periodě 2π, známe frekvenční reprezentace signálu v diskrétním čase v celé jeho frekvenční oblasti.

Zobrazení frekvenční osy se často volí v rozsahu $(-\pi, \pi)$. Nejnižší frekvenční složka v tomto rozsahu je nulová a nejvyšší frekvence je π. Takový signál lze formulovat jako $x(n) = e^{in\pi}$, jehož Fourierova transformace by byla stejná jako Fourierova transformace konstantního signálu $x(n) = 1$, ale posunutá o π ve frekvenční oblasti. Fourierova transformace $x(n) = 1$ je jednoduchý sled diskrétních impulsů s periodou 2π začínající na 0 radiánech za sekundu.

Vztah mezi frekvenční oblastí pro spojitý a diskrétní časový signál $\Omega T_s = \omega$, v intervalu $(-\pi, \pi)$ by odpovídalo $\left(-\frac{F_s}{2}, \frac{F_s}{2}\right)$, ve frekvenční reprezentaci spojitého časového signálu, kde $F_s = \frac{1}{T_s}$ je vzorkovací frekvence. Jinými slovy, když je spojitý časový signál vzorkován s frekvencí F_s, je maximální frekvence, která může být reprezentována ve frekvenční oblasti signálu v diskrétním čase, $\frac{F_s}{2}$.

Fourierova transformace vzorkovaného libovolného signálu $x_c(t)$ je součtem posunuté verze $X_c(i\Omega)$. $X_c(i\Omega)$ bylo posunuto celými násobky Ω_s. Pokud se Ω_s sníží, $X_c(i\Omega)$ se začne překrývat a výsledné $X_s(i\Omega)$ již nevypadá jako $X_c(i\Omega)$. Pokud je tedy Ω_s příliš nízká, dojde k překrytí mezi posunutými verzemi $X_c(i\Omega)$ a $x_c(t)$ již nelze rekonstruovat z $X_s(i\Omega)$ a ztrácíme důležité informace o $x_c(t)$. Toto překrývání posunutých verzí $X_c(i\Omega)$ je známé jako aliasing and je třeba se mu vyhnout. To znamená, že k tomu, abychom neměli žádný aliasing musí platit $\Omega_s - \Omega_M > \Omega_M$, tedy $\Omega_s > 2\Omega_M$. To znamená, že abychom neztratili žádné informace o Fourierově transformaci $x_c(t)$, musí být vzorkování minimálně dvakrát vyšší než je maximální frekvence ve Fourierově transformaci. Toto je známé jako Nyquistova podmínka pro vzorkování. [2]

2.6 Diskrétní Fourierova transformace

Z rovnice **??** víme, že frekvenční reprezentace diskrétního signálu v časové oblasti je definována napříč spojitým měřítkem. Ve vzorkovaném signálu je jen omezený počet vzorků v omezeném časovém okně. Pokud se ale podíváme na již zmíněný vztah je vidět, že čas (n) je definovaný v rozmezí od mínus nekonečna do nekonečna. Jeden způsob, jak přesto spočítat rovnici mimo její vzorkovaný interval je předpokládat, že signál má mimo tento vzorkovaný interval nulovou amplitudu. Jak již bylo řečeno, když má časová reprezentace signálu omezenou nenulovou oblast napříč časovou osou, její frekvenční reprezentace bude mít neomezenou nenulovou frekvenční oblast. Toto není žádaná vlastnost, protože k provedení vzorkování musí mít signál omezený frekvenční rozsah, aby bylo možné splnit Nyquistovu podmínku a vyhnout se aliasingu.

Víme to, že když vzorkujeme signál v čase bude jeho Fourierova transformace periodická. Tedy dle pravidla duality, pokud vzorkujeme Fourierovu transformaci ve frekvenční oblasti, očekávali bychom, že výsledek bude také periodický signál v čase. To znamená, že vzorkování Fourierovy transformace ve frekvenční oblasti by odpovídala signálu v čas s neomezeným časovým rozsahem a to, že takový signál by pak měl omezený frekvenční rozsah ve frekvenční oblasti. Tedy místo toho, abychom předpokládali, že signál s omezeným počtem vzorků má nulovou amplitudu mimo tento rozsah, můžeme předpokládat, že se mimo tento rozsah signál opakuje. [2]

Pokud si představíme diskrétní signál v čase $x(n)$, který má nenulovou amplitudu pouze uvnitř rozsahu od 0 do $N-1$ a pokud vzorkuje $X(e^{i\omega})$ v intervalech $\omega = \frac{2\pi}{N}k$, kde $k = 0, ..., N-1$, můžeme vzorkovaný $X(e^{i\omega})$ napsat jako: [2]

$$\tilde{X}(k) = X(e^{i\frac{2\pi}{N}k}) = \sum_{n=0}^{N-1} x(n)e^{-i\frac{2\pi}{N}kn} \tag{2.7}$$

Tento vztah je známý jako diskrétní Fourierova transformace nebo také "analytická rovnice", protože umožňuje frekvenční analýzu vzorku signálu. Je už jednoduché algebraicky ověřit, že diskrétní Fourierova transformace je invertibilní a že signál lze rekonstruovat na základě jeho diskrétní Fourierovy transformace.

Protože je diskrétní Fourierova transformace invertibilní, můžeme se podívat na vztah, kterému se často říká "syntézní rovnice": [2]

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{i\frac{2\pi}{N}kn} \tag{2.8}$$

Tato rovnice umožňuje rekonstrukci signálu v čase na základě jeho diskrétní Fourierovy transformace.

2.6.1 Frekvenční rozlišení

Jak již bylo nastíněno, diskrétní Fourierova transformace je vzorkovaná verze Fourierovy transformace pro diskrétní časové signály. Toto vzorkování se provádí v krocích $\frac{2\pi}{N}$ ve frekvenční ose, kde N je počet vzorků signálu v čase. Frekvenční interval mezi dvěma po sobě jdoucími vzorky ve frekvenční oblasti je tedy $\frac{2\pi}{N}$ radiánů za sekundu. Už víme, že pro vztah mezi frekvenční reprezentací spojitého signálu v čase a diskrétního signálu v čase platí $\Omega T_s = \omega$, kde T_s je perioda vzorkování a její reciproční $F_s = \frac{1}{T_s}$ je vzorkovací frekvence. Konkrétně pro 2π radiánů za sekundu ve frekvenční oblasti signálu s diskrétním časem, odpovídá $2\pi F_s$ ve frekvenční oblasti spojitého časového signálu. Odpovídající frekvenční rozlišení pro vzorkovaný spojitý časový signál je $\frac{F_s}{N}$.

Podíváme-li se na spojitý časový signál, který má časový rozsah T, tedy, že signál byl vzorkovaný v časovém okně mezi 0 a T sekundami signálu s vzorkovací frekvencí F_s. To znamená, že N, neboli počet vzorků, bude $F_s T$. Pokud si toto N dosadíme do frekvenčního rozlišení, zjistíme, že $\frac{F_s}{N} = \frac{1}{T}$, tedy že frekvenční rozlišení se rovná převrácené hodnotě časového rozsahu spojitého časového signálu.

Pokud tedy potřebujeme získat diskrétní Fourierovu transformaci, která dokáže brát v potaz i malé změny ve frekvenční oblasti, spojitý časový signál by měl být vzorkován v delším časovém rozsahu, aby se daly zaznamenat i malé změny ve frekvenční oblasti. Z toho vidíme, že pokud chceme zaznamenat změny například o frekvencí 0.1 Hz, spojitý časový signál by měl být vzorkován v rozsahu alespoň 10 sekund, a pokud je požadováno ještě jemnější rozlišení, je možné při výpočtu diskrétní Fourierovy transformace použít ještě delší posloupnost vzorků.

Zdá se, že čím delší je časový rozsah signálu, tím lepší bude jeho frekvenční rozlišení. Ale to platí jen pokud nebereme v potaz "ne-stacionaritu"signálu. Ve skutečnosti si nejsme schopni vybrat velmi dlouhý časový rozsah a navíc

zaznamenávat signál, ve dlouhém časovém rozpětí, přináší různá technická úskalí.

Další problém je, že když je signál analyzován v omezeném časovém okně, frekvenční charakteristiky samotného okna ovlivňují frekvenční rozlišení. To znamená, že i když je interval frekvence mezi dvěma po sobě jdoucími vzorky v diskrétní Fourierově transformaci jen malé Δf, pokud má signál dvě frekvenční složky které jsou o něco dále než Δf, je možné, že kvůli frekvenčním charakteristikám okna, diskrétní Fourierova transformace stále nemusí být schopna plně rozlišit tyto dva komponenty od sebe navzájem. [2]

Kapitola 3

Rychlá Fourierova transformace a analýza spektra

3.1 Rychlá Fourierova transformace

Po nějakou dobu sloužila Fourierova transformace jako most mezi časovou a frekvenční oblastí. Nyní je možné přecházet tam a zpět mezi signálem v časové oblasti a frekvenčním spektrem s dostatečnou rychlostí a jednoduchostí, až vznikla zcela nová řada aplikací pro tento klasický matematický nástroj. Tato kapitola je napsána jako úvod do rychlé Fourierovy transformace a s tím souvisejícího Cooley-Tukeyho algoritmu.

Fourierova transformace se dlouho používala pro charakterizaci lineárních systémů a pro identifikaci frekvenčních složek tvořících spojitý signál. Když je však signál vzorkován nebo má být systém analyzován na digitálním počítači, tak musí být použita diskrétní verze Fourierovy transformace. Ačkoli většina vlastností spojité Fourierovy transformace je zachována, několik rozdílů vyplývá z omezení, že diskrétní Fourierova transformace musí pracovat se vzorkovaným signálem definovaným v konečných intervalech.

Rychlá Fourierova transformace (FFT) je jednoduše jen efektivní metoda pro výpočet diskrétní Fourierovy transformace. Rychlá Fourierova transformace lze použít místo spojité Fourierovy transformace pouze v případě, pokud mohla být použita diskrétní Fourierova transformace, ale s podstatně sníženým časem výpočtu. [3]

Pro pochopení FFT začneme s upraveným, již zmíněným, vztahem pro pár diskrétních Fourierových transformací, pro vzorkovaný signál. Tyto vztahy se dají zapsat jako: [3]

$$\begin{aligned} X(j) &= \frac{1}{N}\sum_{k=0}^{N-1} x(k)e^{-i\frac{2\pi}{N}jk} \\ x(k) &= \sum_{j=0}^{N-1} X(j)e^{i\frac{2\pi}{N}jk} \end{aligned} \tag{3.1}$$

,pro $j = 0, 1, ..., N-1; \quad k = 0, 1, ..., N-1; \quad i = \sqrt{-1}$. Jak $X(j)$, tak $x(k)$ jsou obecně komplexní řady. Velké $X(j)$ představuje funkci ve frekvenční oblasti a malé $x(k)$ je funkce v časové oblasti.

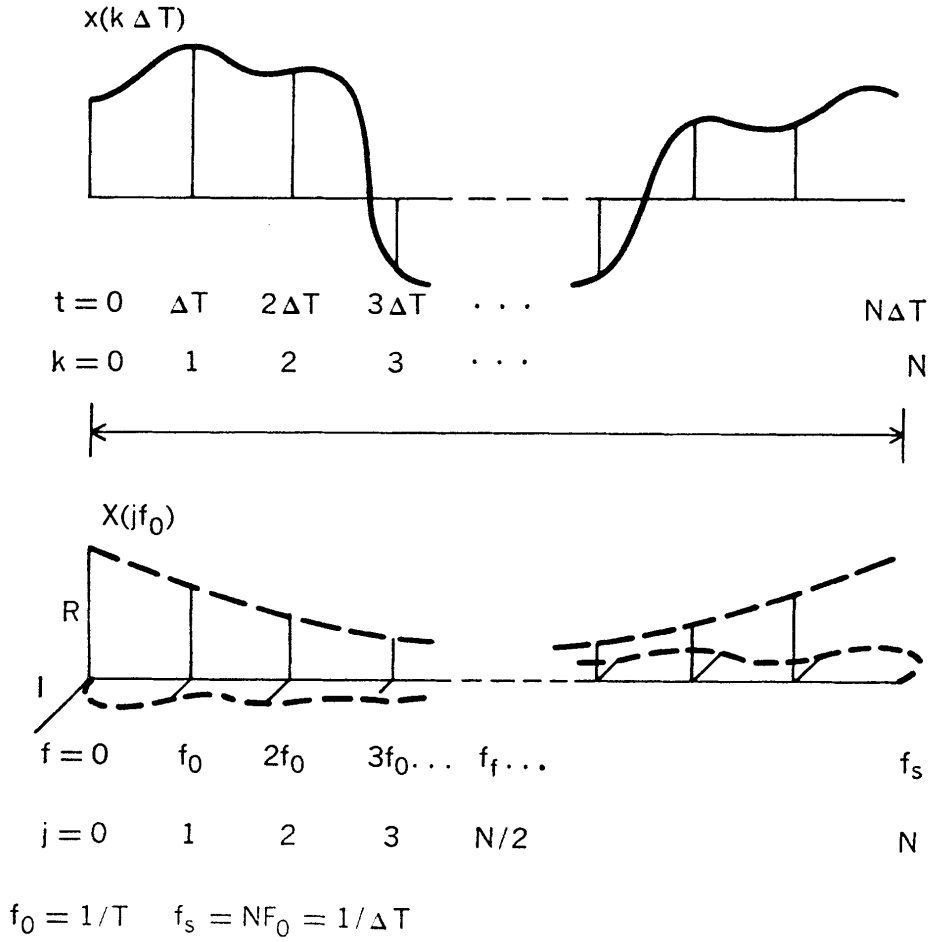
Pokud je výraz $e^{\frac{2\pi i}{N}}$ nahrazen pojmem W_N, získá Transformační pár diskrétní Fourierovy transformace tvar: [3]

$$\begin{aligned} X(j) &= \frac{1}{N}\sum_{k=0}^{N-1} x(k)W_N^{-jk} \\ x(k) &= \sum_{j=0}^{N-1} X(j)W_N^{jk} \end{aligned} \tag{3.2}$$

Příklad časové řady s reálnou hodnotou a její související diskrétní Fourierova transformace je znázorněna na obrázku číslo 3.1. Předpokládá se, že časová řada $x(k\Delta T)$ je periodická v časové oblasti periody T sekund a množina Fourierových koeficientů $X(jf_0)$ je periodická přes vzorkovací frekvenci f_i. U každé funkce je zobrazena pouze jedna úplná perioda.

Základní frekvence f_0 a vzorkovací perioda ΔT se v rovnici 3.2 explicitně neobjevují, ale každé j by mělo být interpretováno jako harmonické číslo a každé k odkazuje na číslo periody vzorku. To znamená, že skutečná frekvence je součinem j a f_0 a skutečný čas je součinem k a ΔT.

Když je $x(k)$ reálná řada, reálná část $X(j)$ je symetrická podle skládací frekvence f_f (kde $f_f = f_s/2$) a imaginární část je antisymetrická. Protože $X(j)$ bylo interpretováno jako periodické. Můžeme tedy říci, že skutečná část $X(j)$ je sudá funkce a že imaginární část $X(j)$ je lichá funkce. To také znamená, že Fourierovy koeficienty mezi $N/2$ a $N-1$ lze považovat za harmonické "negativní frekvence"mezi $-N/2$ a -1. Také lze druhou polovinu časové řady interpretovat jako záporný čas, tedy, že nastala před $t = 0$.



Obrázek 3.1: Reálný signál a jeho komplexní diskrétní Fourierova transformace zobrazená ve formátu algoritmu rychlé Fourierovy transformace. [3]

Dále se podíváme na odvození Cooley-Tukeyho algoritmu rychlé Fourierovy transformace pro vyhodnocení druhé rovnice 3.2, pro dané $N = 8$. Cooley-Tukeyho algoritmus je zdaleka nejpoužívanější algoritmus pro rychlou Fourierovu transformaci. Toto odvození je vhodné i pro dopřednou transformaci, protože první rovnice 3.2 lze přepsat do formy: [3]

$$X(j) = \frac{1}{N}[\sum_{k=0}^{N-1} x(k) * W_N^{jk}]^* \tag{3.3}$$

,kde hvězdička odkazuje na operaci komplexní konjugace. Alternativně, lze algoritmus rychlé Fourierovy transformace, použitý pro výpočet druhé rovnice 3.2, změnit předefinováním W_N na $e^{\frac{-2\pi i}{N}}$ a vydělením každého výsledku daným N.

Použitím Cooleyho zápisu, zahrnuje algoritmus výpočtu ryvhlé Fourierovy transformace vyhodnocení výrazu: [3]

$$\hat{X}(j) = \sum_{k=0}^{N-1} A(k) * W^{jk} \tag{3.4}$$

,kde $j = 0, 1, ..., N-1$ a $W = e^{\frac{2\pi i}{N}}$. $\hat{X}$ a A mohou být interpretovány jako X^* a x^*/N, pokud je počítána dopředná transformace, nebo x a X, pokud je počítána transformace inverzní.

Když se N rovná 8, je vhodné reprezentovat jak j, tak k jako binární čísla. Tedy pro $j = 0, 1, ..., 7$ a $k = 0, 1, ..., 7$, můžeme psát: $j = j_2 4 + j_1 2 + j_0$ a $k = k_2 4 + k_1 2 + k_0$, kde $j_0, j_1, j_2, k_0, k_1, k_2$ mohou nabývat hodnot 0 a 1. Pomocí této reprezentace j a k, můžeme psát rovnici 3.4 jako: [3]

$$\begin{aligned} &\hat{X}(j_2, j_1, j_0) = \\ &\sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} A(k_2, k_1, k_0) W^{(j_2 4 + j_1 2 + j_0)(k_2 4 + k_1 2 + k_0)} \end{aligned} \tag{3.5}$$

Když vezmeme v potaz, že $W^{m+n} = W^m W^n$, získáme: [3]

$$\begin{aligned} &W^{(j_2 4 + j_1 2 + j_0)(k_2 4 + k_1 2 + k_0} = \\ &W^{(j_2 4 + j_1 2 + j_0) k_2 4} W^{(j_2 4 + j_1 2 + j_0) k_1 2} W^{(j_2 4 + j_1 2 + j_0) k_0} \end{aligned} \tag{3.6}$$

Podíváme-li se na tyto pojmy jednotlivě, je vidět, že je lze zapsat ve formě: [3]

$$\begin{aligned} W^{(j_2 4 + j_1 2 + j_0) k_2 4} &= [W^{8(j_2 2 + j_1) k_2}] W^{j_0 k_2 4} \\ W^{(j_2 4 + j_1 2 + j_0) k_1 2} &= [W^{8 j_2 k_1}] W^{(j_1 2 + j_0) k_1 2} \\ W^{(j_2 4 + j_1 2 + j_0) k_0} &= W^{(j_2 4 + j_1 2 + j_0) k_0} \end{aligned} \tag{3.7}$$

Všimněme si také, že $W^8 = [e^{\frac{2\pi i}{8}}]^8 = e^{2\pi i} = 1$. Tudíž části rovnic 3.7 v

závorkách lze nahradit jedničkou. To znamená, že rovnici 3.5 lze zapsat ve formě: [3]

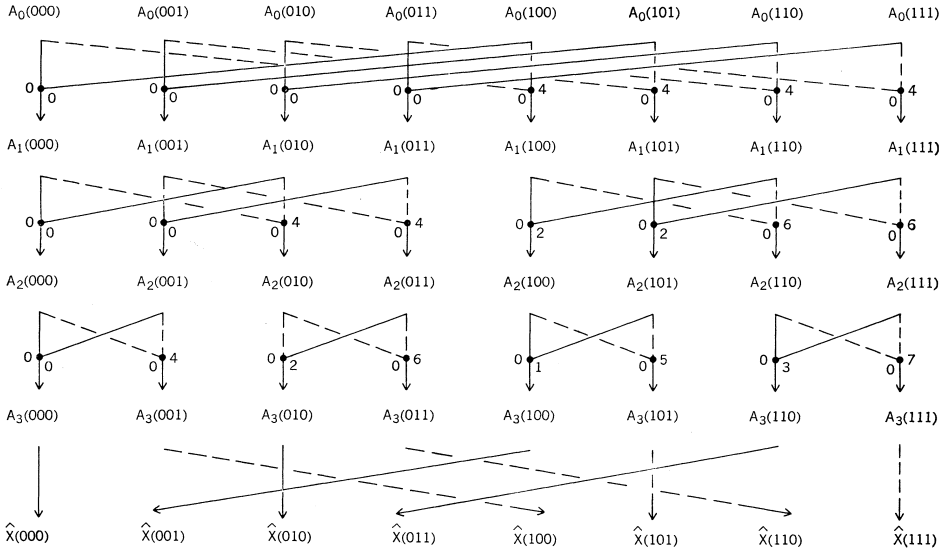
$$\hat{X}(j_2, j_1, j_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} A(k_2, k_1, k_0) W^{j_0 k_2 4} W^{(j_1 2 j_0) k_1 2} W^{(j_2 4 j_1 2 + j_0) k_0} \tag{3.8}$$

V této formě je vhodné provádět každý ze součtů samostatně a pracovat s mezivýsledky. Všimněme si, že každá sada se skládá pouze z osmi částí a že je třeba uložit pouze nejnovější sadu. Rovnice tak mohou být přepsány do tvaru: [3]

$$\begin{aligned} A_1(j_0, k_1, k_0) &= \sum_{k_2=0}^{1} A(k_2, k_1, k_0) W^{j_0 k_2 4} \\ A_2(j_0, j_1, k_0) &= \sum_{k_1=0}^{1} A_1(j_0, k_1, k_0) W^{(j_1 2 j_0) k_1 2} \\ A_3(j_0, j_1, j_2) &= \sum_{k_0=0}^{1} A_2(j_0, j_1, k_0) W^{(j_2 4 j_1 2 + j_0) k_0} \\ \hat{X}(j_2, j_1, j_0) &= A_3(j_0, j_1, j_2) \end{aligned} \tag{3.9}$$

Části přispívající ke každému součtu jsou znázorněny na obrázku číslo 3.2. Každé malé číslo označuje mocninu W aplikovanou podél sousední dráhy. Poslední operací zobrazenou na obrázku číslo 3.2 je přeuspořádání. To je způsobeno bitovou inverzí v argumentech posledního vztahu v sadě rovnic 3.9.

Tato sada rekurzivních rovnic představuje původní Cooleyovu-Tukeyovu formulaci algoritmu rychlé Fourierovy transformace pro $N = 8$. Ačkoli přímé vyhodnocení rovnice 3.4 pro $N = 8$ by vyžadovalo téměř 64 složitých operací násobení a sčítání, rovnice rychlé Fourierovy transformace zdánlivě potřebuje jen 48 operací. A když si všimneme, že první násobení v každém součtu je ve skutečnosti násobením jedničkou, stane se toto číslo pouze 24. Po všimnutí, že $W_0 = -W_4$, $W_1 = -W_5$ etc., lze počet násobení snížit až na 12. Tyto redukce pokračují až k obecnějšímu případu $N = 2^m$, což snižuje výpočet z téměř N^2 operací na $(N/2)log_2N$ komplexních násobení, $(N/2)log_2N$ komplexních sčítání a $(N/2)log_2N$ odečítání. Pro $N = 1024$ to představuje snížení výpočtů o více než 200 na 1. Tento rozdíl je graficky znázorněn na obrázku číslo **??**



Obrázek 3.2: Vývojový diagram Cooley-Tukeyho algoritmu rychlé Fourierovy transformace pro provedení osmibodové transformace. [3]

3.1.1 Možnosti využití FFT

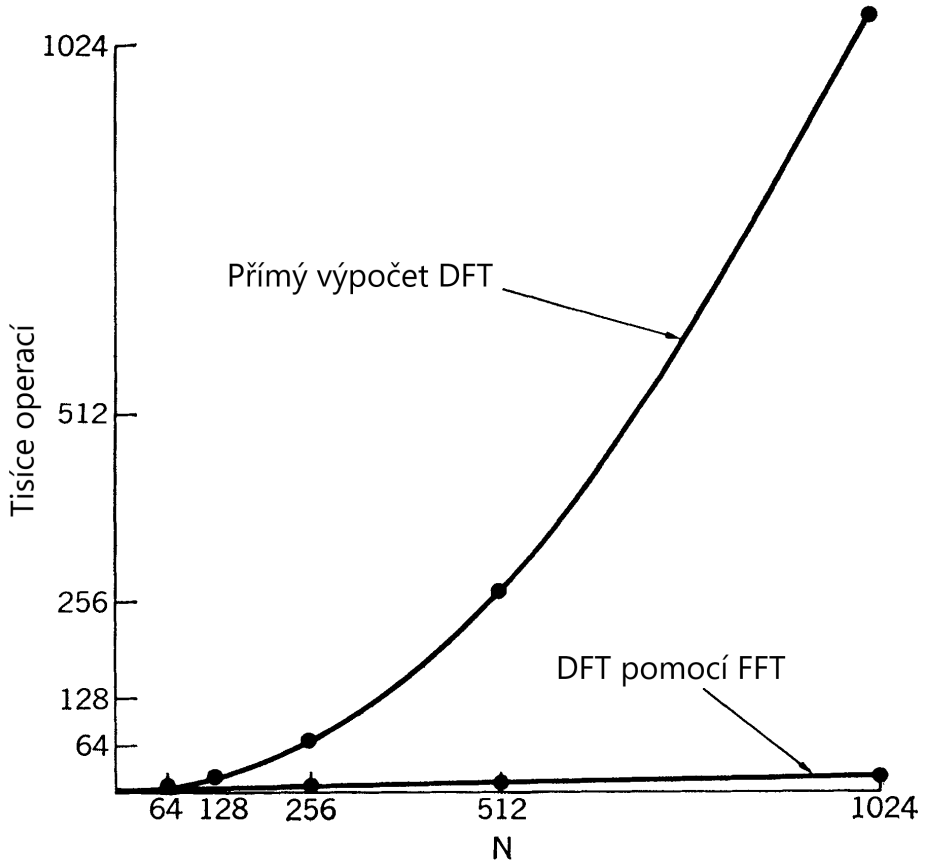
Operace obvykle spojené s FFT jsou:

- Výpočet spektrogramu (zobrazení krátkodobého výkonového spektra jako funkce času)
- Konvoluce dvou časových řad pro provádění digitální filtrace
- Korelace dvou časových řad

Přestože všechny tyto operace lze provádět bez rychlé Fourierovy transformace, její výpočetní úspory výrazně zvýšily zájem o provádění těchto operací digitálně.

3.2 Korelace a konvoluce

Další pojmy, které je důležité vysvětlit jsou korelace a konvoluce. Tyto pojmy už byly a budou zmíněny, proto by bylo vhodné je popsat.



Obrázek 3.3: Počet operací potřebných pro výpočet diskrétní Fourierovy transformace pomocí algoritmu rychlé Fourierovy transformace ve srovnání s počtem operací potřebných pro přímý výpočet diskrétní Fourierovy transformace. [3]

3.2.1 Korelace

Korelace se ve statistice používá ke zjištění, jak jsou dvě proměnné spolu spojeny. například, jestli pokud má jedna proměnná rostoucí trend, můžeme zjistit, jestli má druhá proměnná stejně rostoucí trend nebo má trend opačný nebo si nejsme jistí. Předpokládejme, že pracujeme se signály, jejichž průměr se rovná nule. Nenulový průměr jen způsobí posun ve výpočtech

Pro stanovení míry asociace se používá korelační koeficient (CC) mezi dvěma signály. Ten může mít hodnotu mezi -1 a 1. Pokud je korelační koeficient kladný, znamená to že oba signály spolu pozitivně korelují. Rostoucí/klesající trend jednoho signálu je pravděpodobně spojen s podobným trendem druhého signálu. Pokud se korelační koeficient rovná 1, tato asociace je perfektní a každý rostoucí/klesající trend jednoho signálu je spojen se stejným trendem

druhého signálu. Ve statistice se definují různé verze korelačního koeficientu, ale zde je popsán korelační koeficient, který je znám jako Pearsonova korelace. [2]

Pokud se vypočítá korelace mezi signálem a jeho verzí posunutou v čase, získáme funkci času, která je známá jako korelační funkce. Matematicky lze napsat jako: [2]

$$z(t) = \int_{-\infty}^{\infty} x(\tau)y(t+\tau)d\tau \tag{3.10}$$

Budeme předpokládat, že rozptyl $x(t)$ a $y(t)$ je normalizovaný. Můžeme tedy psát funkci i jako: [2]

$$z(t) = x(t) * y(-t) \tag{3.11}$$

Pokud potřebujeme, aby byl výsledek v mezích -1 a 1, měl by se výsledný integrál vydělit druhou odmocninou součinu rozptylů signálů. [2]

3.2.2 Konvoluce

V práci nebudu vysvětlovat, jak je matematicky konvoluce odvozená, jen jednoduše ukážu, jak tato operace vypadá matematicky a jak je implementována. Předpokládejme, že signál $x(t)$ je vstup do systému LTI (Lineární časově invariantní systém) s impulsní odezvou $h(t)$. LTI systémy nejsou v rozsahu této práce. Pro nás stačí jen vědět, že LTI je speciální druh systému, který je lineární a zároveň nemění své chování v čase.$h(t)$ se také dá představit jako další signál nezávislý na jeho přidruženém systému. Poté je $y(t)$ definováno jako konvoluce $x(t)$ a $h(t)$ a lze psát jako: [2]

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau) * h(t-\tau)d\tau \tag{3.12}$$

Pro některé specifické signály může být tento integrál analyticky vyřešen a $y(t)$ může být zapsaný jako funkce času, ale je možné tento integrál vypočítat i numericky.

Tento integrál se dá jednoduše popsat tak, že bychom měli najít zrcadlový obraz $h(\tau)$ s respektem k ose Y a následně posunout tento zrcadlový obraz v čase. Pokud se bude zrcadlový obraz posouvat na levou stranu, dokud se $h(\tau)$ a $x(\tau)$ nebudou překrývat, bude se integrovaná funkce $x(\tau)h(t-\tau)$ rovnat nule a integrál bude tedy také nulový. Pokud ale posunete zrcadlový obraz doprava, $h(\tau)$ a $x(\tau)$ se budou znovu překrývat a integrál se dá znovu vypočítat. [2]

Podobnou operaci lze provést pro signály v diskrétním čase, ale integrace se musí převést na součet. Konvoluce mezi $x(n)$ a $h(n)$ pro diskrétní časové signály lze tedy definovat jako: [2]

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k) * h(n-k) \tag{3.13}$$

3.3 Výkonové a amplitudové spektrum

I přesto, že nás, kvůli zadanému programu, zajímá amplitudová spektrální hustota (ASD), neboli amplitudové spektrum, v kapitole bude hlavně popsáno výkonové spektrum. A to z jednoduchému důvodu, protože amplitudové spektrum se rovná druhé odmocnině výkonového spektra.

3.3.1 Autokorelační funkce

První se musíme podívat na pojem "stacionarita", který bude důležitá dále v kapitole. Pokud pro náhodnou proměnnou platí, že její statistické vlastnosti jsou zachovány v průběhu času, říkáme, že je tato náhodná proměnná "silně stacionární". Silná stacionarita je velmi omezující předpoklad, například u většiny biologických signálů zachována není. Proto existuje i uvolněnější alternativa zvaná "slabá stacionarita". Náhodná proměnná se slabou stacionaritou nemusí zachovat všechny své statistické vlastnosti v čase, ale musí zachovat alespoň některé vybrané. Typicky se vybírají vlastnosti jako je střední hodnota, rozptyl nebo autokorelační funkce.

Funkce automatické korelace má určité podobnosti s konceptem konvoluce. Výpočtem autokorelace lze kvantifikovat, do jaké míry signál koreluje s jeho

posunutými verzemi v čase. Když je časový posun 0, signál je plně korelován sám se sebou a korelační koeficient mezi nimi by byl 1. Když začneme posouvat signálem v čase a vypočítávat korelační koeficient mezi posunutou verzí a původním signálem, posunutá verze signálu již nebude plně korelovaná s původním signálem a korelačním koeficientem se nebude rovnat 1. Pokud bychom ale ukládali vypočítané korelační koeficienty napříč posuny v čase, můžeme z těchto koeficientů v čase vytvořit funkci autokorelace. Těmto posuvům v čase se často říká časové zpoždění.

Jak již bylo řečeno slabá stacionarita je pro náhodné proměnné uvolněnější požadavek než silná stacionarita. Velké množství náhodných procesů tuto podmínku splňují alespoň v krátkých časových oknech. Pokud je tedy požadována frekvenční analýza náhodného procesu, provedení analýzy pomocí autokorelační funkce by umožnilo dosáhnout konzistentního výpočtu frekvenční reprezentace tohoto náhodného procesu.

Z tohoto všeho můžeme odvodit jaký je vztah mezi frekvenční reprezentací autokorelační funkce a samotné frekvenční reprezentace náhodné proměnné. [2]

Pro signál s reálnou hodnotou, průměrem 0 a směrodatnou odchylkou 1 $x(t)$, můžeme psát autokorelační funkci jako: [2]

$$r_x(t) = \int_{-\infty}^{\infty} x(\tau)x(t+\tau)d\tau \tag{3.14}$$

Pokud se tato formulace porovná s tím, co bylo uvedeno pro konvoluci, můžeme dojít k závěru, že: [2]

$$r_x(t) = x(t) * x(-t) \tag{3.15}$$

Z této rovnice můžeme lépe pochopit frekvenční reprezentaci autokorelační funkci. Když se podíváme na konvoluční teorém, je vidět, že můžeme psát: [2]

$$R_x(j\Omega) = X(j\Omega) * X(-j\Omega) \tag{3.16}$$

, kde $X(j\Omega)$ je Fourierova transformace signálu $x(t)$. $x(t)$ je signál s reálnou

hodnotou, což znamená, že amplituda tohoto signálu je reálné číslo. To znamená, že můžu psát: [2]

$$X(-j\Omega) = X * (j\Omega) \tag{3.17}$$

, kde $X * (j\Omega)$ je konjugát $X(j\Omega)$. V matematice je konjugát komplexního čísla, číslo se stejnou reálnou částí a imaginární částí, se stejnou velikostí, ale s opačným znaménkem. Nyní se můžeme podívat na zajímavý vztah mezi Fourierovou transformací původního signálu a jeho autokorelační funkce: [2]

$$R_x(j\Omega) = |X(j\Omega)|^2 \tag{3.18}$$

, kde $|X(j\Omega)|$ je velikost Fourierovy transformace signálu $x(t)$. Podobný koncept může být odvozený i pro diskrétní časové signály. [2]

3.3.2 Výpočet spektra

Nyní, máme definici autokorelační funkce signálu a víme, že je tato funkce sama o sobě signálem v čase a její frekvenční reprezentace má jasný vztah s frekvenční odezvou signálu. Fourierova transformace autokorelační funkce se nazývá výkonová spektrální hustota (PSD), neboli výkonové spektrum, protože představuje to, jak je výkon signálu distribuován podél frekvenční osy. [2]

K získání hodnoty výkonové spektrální hustoty lze použít různé algoritmy. O těch obecně platí, že se dělí do tří hlavních kategorií: [2]

- neparametrické metody
- parametrické metody
- tzv. subspace metody

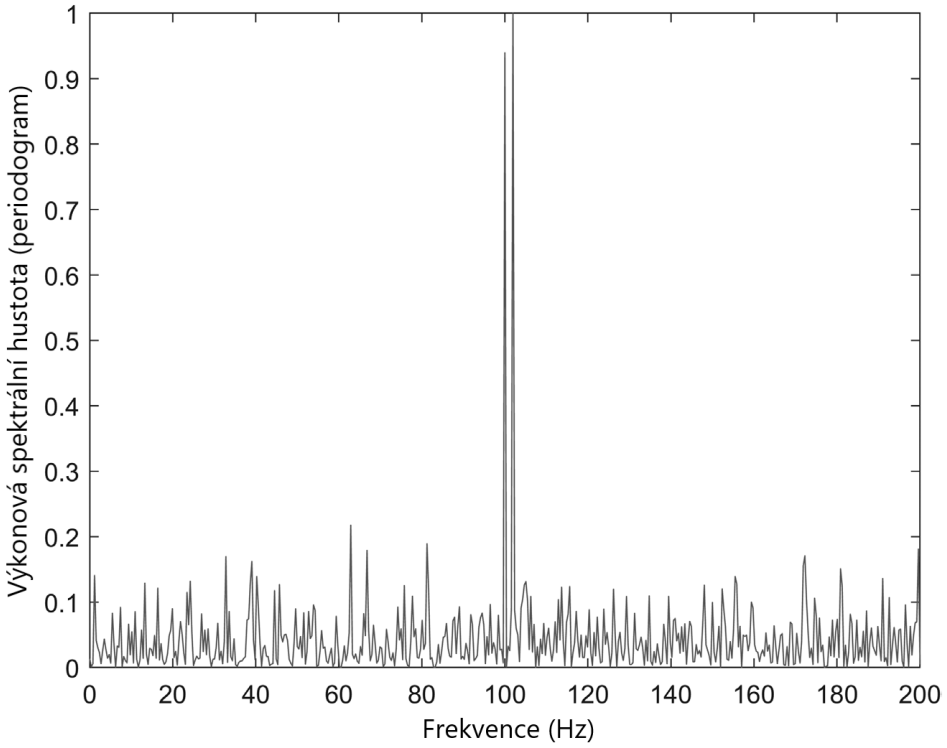
Subspace metody nejsou v rozsahu této práce. Zajímají nás hlavně neparametrické metody, proto ty, a v menší míře parametrické metody, budou popsány dále v kapitole.

3.3.3 Neparametrické metody

Podstata této metody je jen spočítat autokorelační funkci a poté signál transformovat pomocí rychlé Fourierovy transformace, čímž získáme výkonovou spektrální hustotu. I když se podstata tohoto konceptu může zdát jednoduchá, lze narazit na technické problémy, jako jsou vliv šumu a oken nebo bias a odchylka výsledku.

K provedení výpočtů můžeme buď transformovat signál pomocí rychlé Fourierovy transformace a poté vypočítat druhou mocninu jeho velikosti (periodogram) nebo spočítat autokorelační funkci a poté jí transformovat pomocí rychlé Fourierovy transformace (korelogram).

Obrázek číslo 3.4 ukazuje periodogram synteticky generovaného signálu. Signál se skládá ze dvou specifických frekvenčních složek na 100 a 102 Hz, a náhodného šumu. [2]

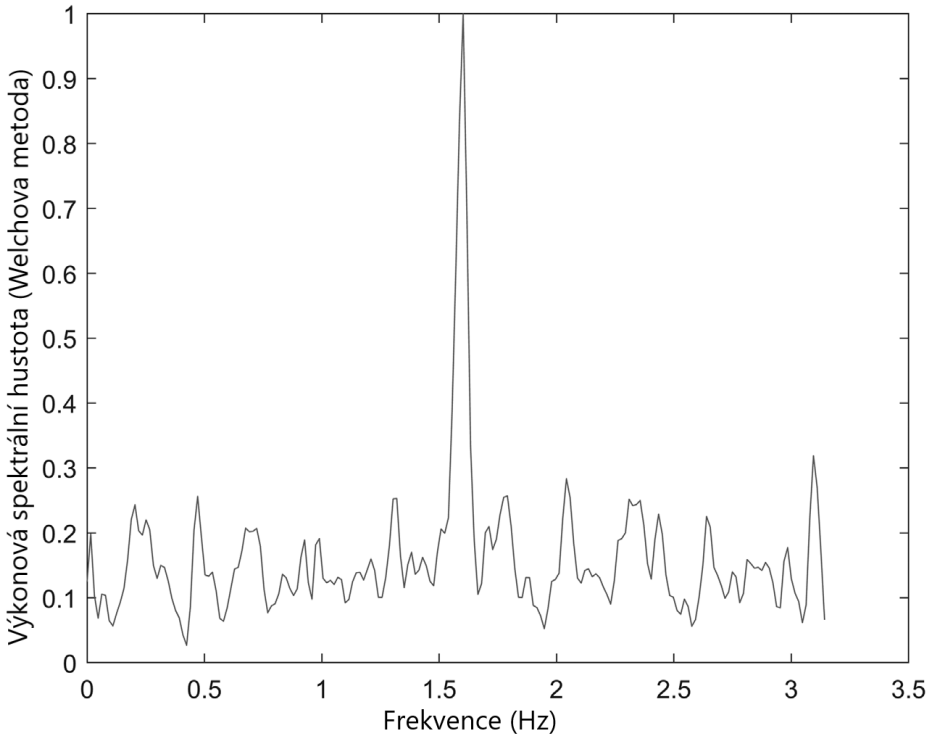


Obrázek 3.4: Periodogram syntetického signálu. Dvě frekvenční složky na 100 a 102 Hz jsou jasně zřetelné ve srovnání se šumem v pozadí. [2]

Periodogram vypadá velmi zubatě, protože rozptyl výsledku je vcelku vysoký. Dále si můžeme všimnout, že frekvenční rozsah ve zobrazení tohoto grafu zahrnuje pouze kladné hodnoty. Pokud by byl výpočet výkonové spektrální

hustoty oboustranný, byl by výsledek zobrazený po celou periodu frekvence mezi nulou a vzorkovací frekvencí.

Existuje několik způsobů, jak omezit rozptyl výsledku. Jednoduchý způsob je použít průměrovací filtr na výsledku výkonové spektrální hustoty (Daniel periodigram). Velmi často používaný algoritmus se nazývá Welchova metoda. Welchova metoda rozdělí signál na menší překrývající se části a poté vypočte výkonovou spektrální hustotu pro každou z těchto částí. Celkový výsledek výkonové spektrální hustoty se získá zprůměrováním vypočtené výkonové spektrální hustoty z každé části. Welchova metoda předpokládá stacionaritu signálu v daném časovém okně a tedy, že vypočtená výkonová spektrální hustota v každé z částí je odvozena pro realizaci stejné náhodné proměnné. Na obrázku číslo 3.5 můžeme vidět periodogram pomocí Welchovy metody stejného signálu, jako který byl použit pro obrázek číslo 3.4.



Obrázek 3.5: Periodogram syntetického signálu pomocí Welchovy metody. Syntetický signál je stejný jako signál použitý při vytváření periodogramu na obrázku číslo 3.4 [2]

U Welchovy metody je vidět, že rozptyl výsledku byl značně snížen. Nicméně dvě frekvenční složky na 100 a 102 Hz, které byly jasně rozlišitelné na obrázku číslo 3.4, jsou nyní nerozlišitelné. Je zřejmé, že když je původní signál rozdělen do menších částí, frekvenční rozlišení se zhorší.

Při výpočtu výkonové spektrální hustoty je často nutné vybrat správné okno

pro ovlivnění frekvenčního úniku, což bude popsáno v další části o okénkování, a rozlišení. Dále, v závislosti na počtu vzorků použitých ve výpočtu rychlé Fourierovy transformace, může být signál na konci doplněn nulami.

Okénkování

Další pojem, který je důležitý vysvětlit je okénkování. Už bylo vysvětleno, že když se časové zpoždění zvyšuje, bude překrytí mezi původním signálem a jeho posunutou verzí menší. Proto počet vzorků zahrnutých do výpočtu korelačního koeficientu, mezi původním signálem a posunutou verzí signálu, bude nižší s tím, jak se časové zpoždění blíží k začátku a konci původního signálu.

To povede k vyššímu rozptylu výsledku na okrajích. Za účelem snížení tohoto nechtěného jevu se využívá okénkování. Běžně používaná okna se na okrajích zužují, a tím oslabují účinek vzorků na okrajích při výpočtu korelačního koeficientu. Pokud se žádné okenkování nepoužívá, je to stejné jakoby bylo použito obdélníkové okno o stejné velikosti jako originální signál. Dokud se tedy pracuje se signálem s omezeným časovým rozpětím, okna jsou nedílnou součástí výpočtu výkonového spektra. [2]

Předpokládejme, že existuje libovolný diskrétní časový signál $x(n)$ a funkce okna v čase $w(n)$. Matematicky řečeno, okénkování odpovídá násobení funkce okna a signálu v čase. Můžeme tedy psát výsledný signál $x_r(n)$ jako: [2]

$$x_r(n) = x(n)w(n) \tag{3.19}$$

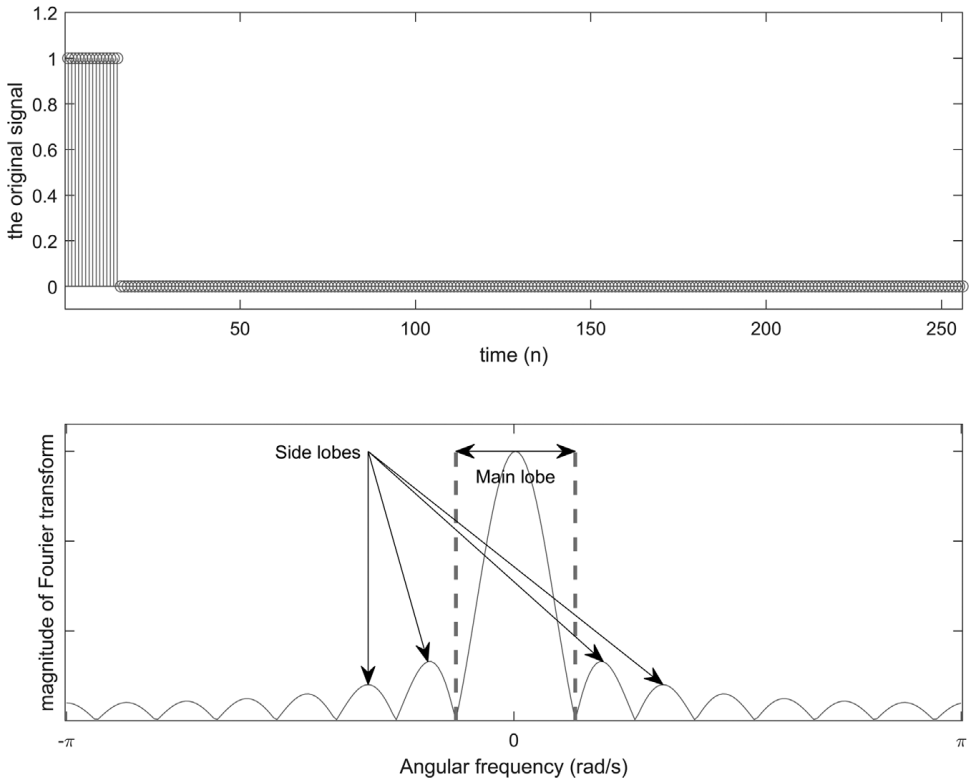
Pro diskrétní časové signály, násobení v čase odpovídá operaci ve frekvenční oblasti, která se nazývá „periodická konvoluce“. Tento druh konvoluce se používá, protože Fourierova transformace signálu s diskrétním časem je periodická funkce, takže je třeba počítat konvoluční integrál dvou periodických signálů s hlavní periodou 2π. Fourierova transformace $x_r(n)$ je sama o sobě periodická, takže by měla být i periodická konvoluce počítána v periodě 2π. Získáme tedy rovnici: [2]

$$X_r(e^{i\omega}) = X(e^{i\omega}) * W(e^{i\omega}) \tag{3.20}$$

, kde $*$ v tuto chvíli značí periodickou konvoluci. $X_r(e^{i\omega})$ a $W(e^{i\omega})$ jsou Fourierovy transformace $x(n)$ a $w(n)$. Pokud by $w(n)$ bylo konstantní přes celou časovou osu bez jakýchkoliv limitů, $W(e^{i\omega})$ by byl sled impulzů s periodou $2/pi$ a $X_r(e^{i\omega})$ by se rovnalo $X(e^{i\omega})$.

Dále se podívejme, jaký by byl účinek okénkování ve frekvenční oblasti. Na začátek si ukážeme případ pro obdélníkové okno. Obrázek číslo 3.6 ukazuje velikost Fourierovy transformace obdélníkového okna. Jak je vidět, energie signálu je kondenzována v určitých frekvenčních rozsazích nazývaných laloky. Největší lalok, který je centrován na 0 Hz, se nazývá hlavní lalok a vedle něj jsou vidět postranní laloky.

S rostoucí délkou okna se šířka laloků zmenšuje, ale plocha pod křivkou zůstává konstantní. Je nutné mít na paměti, že oscilace ve frekvenční reprezentaci okna $W(e^{i\omega})$ budou mít vliv na frekvenční reprezentaci okénkovaného signálu $X_r(e^{i\omega})$. [2]

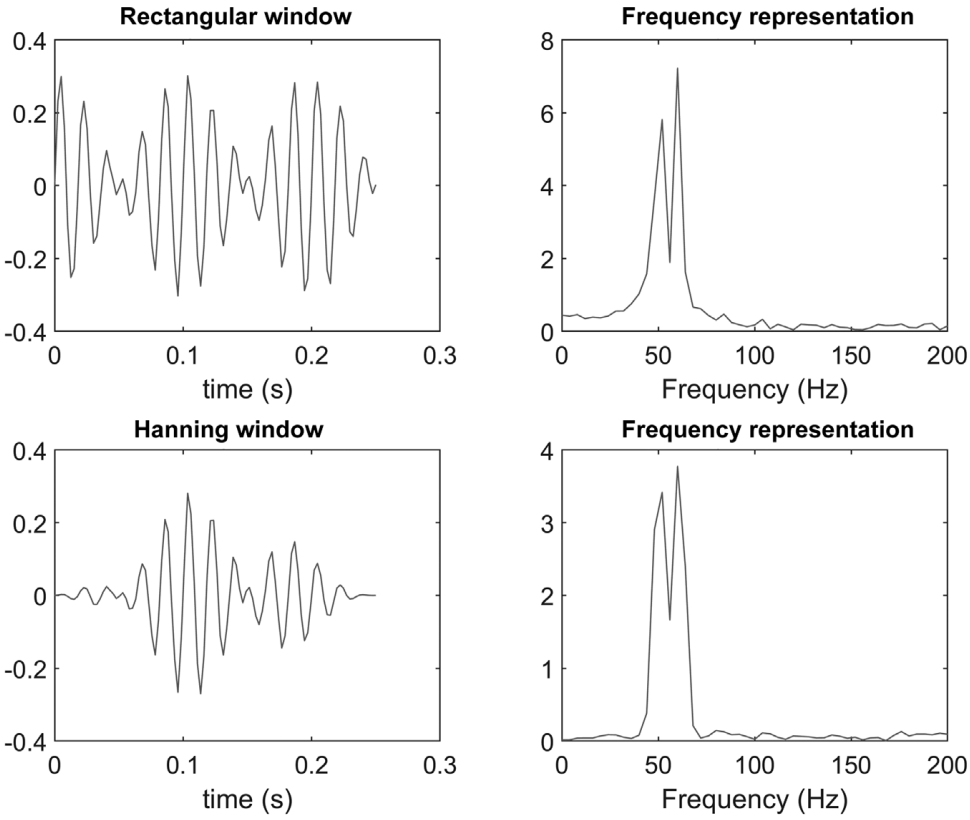


Obrázek 3.6: Velikost Fourierovy transformace pravoúhlého okna a jeho hlavního a postranní laloky. [2]

Jedním z velmi důležitých pojmů pro výsledek okénkování je něco, čemu se říká rozmazání spektra. Obrázek číslo 3.7 znázorňuje signál, který má dvě frekvenční složky na 50 a 60 Hz. Tyto dvě složky mají stejnou energetickou hladinu. Pokud je však na signál aplikováno okno, detekuje frekvenční re-

prezentace signálu dvě složky s různou energetickou hladinou. Důvodem je to, že část energie signálu ve stávajících frekvenčních komponentách unikla do ostatních komponent. Tento jev je primárně ovlivněn relativní velikostí hlavního laloku vůči postranním lalokům. Tedy, čím větší je amplituda postranních laloků vzhledem k hlavnímu laloku, tím vlivnější je efekt rozmazání spektra. Obdélníkové okno má poměrně velké postranní laloky, což vede ke značnému efektu rozmazání. Obdélníkové okno tedy není příliš populární v odhadu výkonového spektra.

Existuje mnoho různých druhů oken, například v odhadu výkonového spektra se běžně používají okna „Hamming“ a „Hanning“. Tato okna se na okrajích zužují, a proto mají, ve srovnání s obdélníkovým oknem, menší bočními laloky. Na obrázku číslo 3.7 je vidět, že když je signál okénkovaný Hanningovým oknem, energie dvou frekvenčních složek je reprezentována rovnoměrněji, ve srovnání s obdélníkovým okénkem, což sníží efekt rozmazání spektra. Bohužel je to ale za cenu snížení frekvenčního rozlišení. Šířka hlavního laloku v Hanningově okně je také širší než u obdélníkového okna a jak je vidět na obrázku číslo 3.7 obdélníkové okno je tedy, ve srovnání s Hanningovým oknem, efektivnější pro oddělování dvou frekvenčních složek v signálu. [2]



Obrázek 3.7: Velikost Fourierovy transformace pravoúhlého okna a jeho hlavního a postranní laloky. [2]

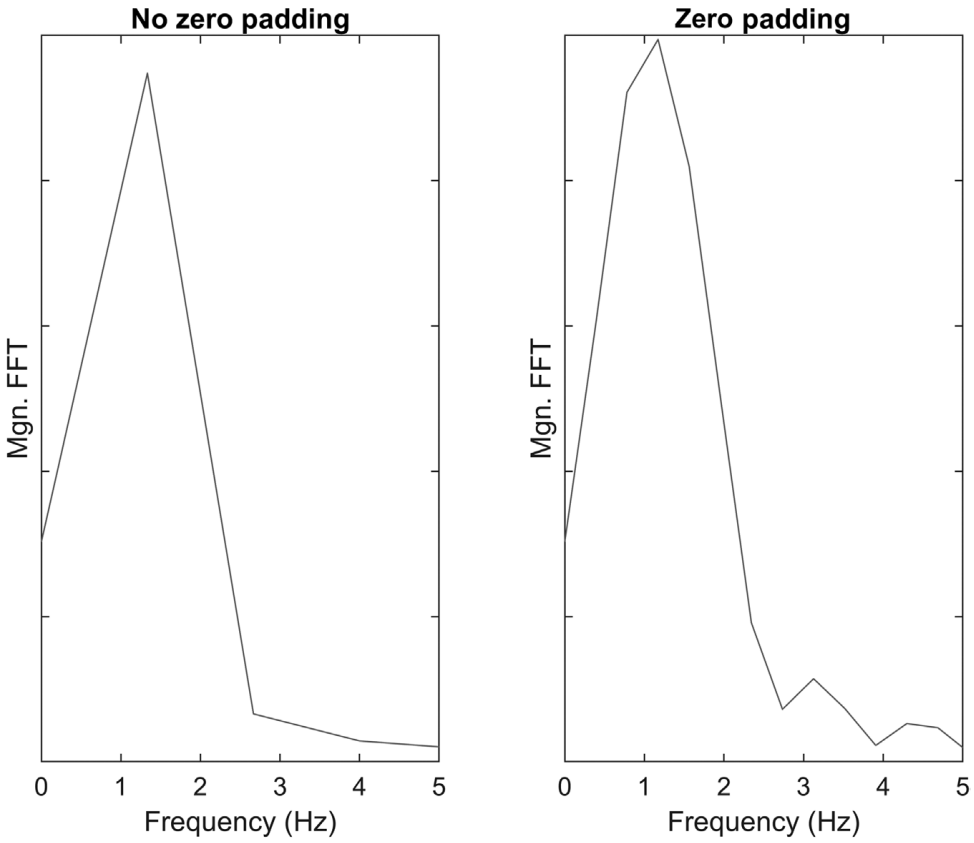
Doplnění nulami

Algoritmy rychlé Fourierovy transformace poběží rychleji a jsou výpočetně účinnější, pokud je délka signálu mocninou dvou. Pokud tedy signál nemá délku mocniny dvou, lze přidat nuly na konec vzorku sekvence tak, aby její délka mocninou dvou byla. Toto doplnění nám, ale pochopitelně ovlivňuje výsledek frekvenční analýzy signálu. Doplnění nulami je pro nás v podstatě ekvivalentní situace, jako mít signál s větší délkou než má původní signál, který je pak vynásoben obdélníkovým oknem o stejné délce jako má původní signál. Předpokládejme, že rychlá Fourierova transformace je jednostranná a ukazuje jen kladné frekvence. Její frekvenční interval je tedy mezi nulou a $\frac{F_s}{2}$ a jak se zvyšuje počet vzorků v algoritmu rychlé Fourierově transformace, frekvenční interval mezi po sobě jdoucími vzorky v rychlé Fourierově transformaci bude kratší. Poté nás zajím, jak doplnění nulami ovlivní frekvenční rozlišení. Obrázek číslo 3.8 ukazuje signál se dvěma sousedními frekvenčními složkami na 1 a 1.5 Hz. Původní signál byl vzorkován frekvencí 400 Hz po dobu 750 ms. Jak je vidět, rychlá Fourierova transformace bez doplnění nulami nemůže ukázat existenci těchto dvou složek. Nyní si představme, že vyplníme vzorkovaný signál nulami a přepočítáme rychlou Fourierovu transformaci. V tomto případě byl signál doplněn nulami a měl 2048 vzorků. Pokud je vzorkovací frekvence 400 Hz, 2048 vzorků by mělo odpovídat 5.1 sekundy signálu a frekvenční interval mezi po sobě jdoucími vzorky rychlé Fourierovy transformace by měl být 0.2 Hz. Kdybychom měli tak přesnou frekvenci rozlišení, měli by být frekvenční složky 1 a 1.5 v rychlé Fourierově transformaci identifikovatelné. Nicméně, jak je vidět na obrázku číslo 3.8, výsledek rychlé Fourierovy transformace sice vypadá o moc hladší, ale stále není schopen rozlišovat mezi těmito dvěma sousedními frekvenčními komponenty. [2]

Zdá se, že doplnění nul nepřidává žádné nové informace o signálu, tedy, že nezlepšuje frekvenční rozlišení. Všimněte si však, že frekvenční interval mezi dvěma po sobě jdoucími vzorky rychlé Fourierovy transformace signálu s doplněním nul je kratší. A rychlá Fourierova transformace by se tedy vypočítávala pro hustší koncentraci bodů podél frekvenční osy. [2]

3.3.4 Parametrické metody

Pokud se výkonové spektrum odhaduje pomocí neparametrických metod, předpokládáme, že nemáme mnoho informací o jevu, který generoval zpracovávaný signál a máme jen běžné předpoklady, které jsme měli pro výpočet rychlé Fourierovy transformace. Nicméně, v některých případech můžeme být schopni signál modelovat a poté popsat signál jako omezený soubor pa-



Obrázek 3.8: Rozdíl mezi dvěma rychlými Fourierovy transformacemi stejného signálu, kde jedna je doplněna nulami. [2]

rametrů, kterým říkáme parametry modelu. Pomocí těchto metod, můžeme dosáhnout mnohem lepšího frekvenční rozlišení, ale jsou přesné pouze pokud jsme schopni sestrojit model, který dokáže dostatečně popsat signál a můžeme vyvinout výpočetní metody pro odhad parametrů tohoto modelu na základě vzorků signálů. Tyto metody pro tuto práci nejsou zajímavé a proto se jimi nebudeme více zabývat. [2]

3.4 Spektrogram a waterfall spektrum

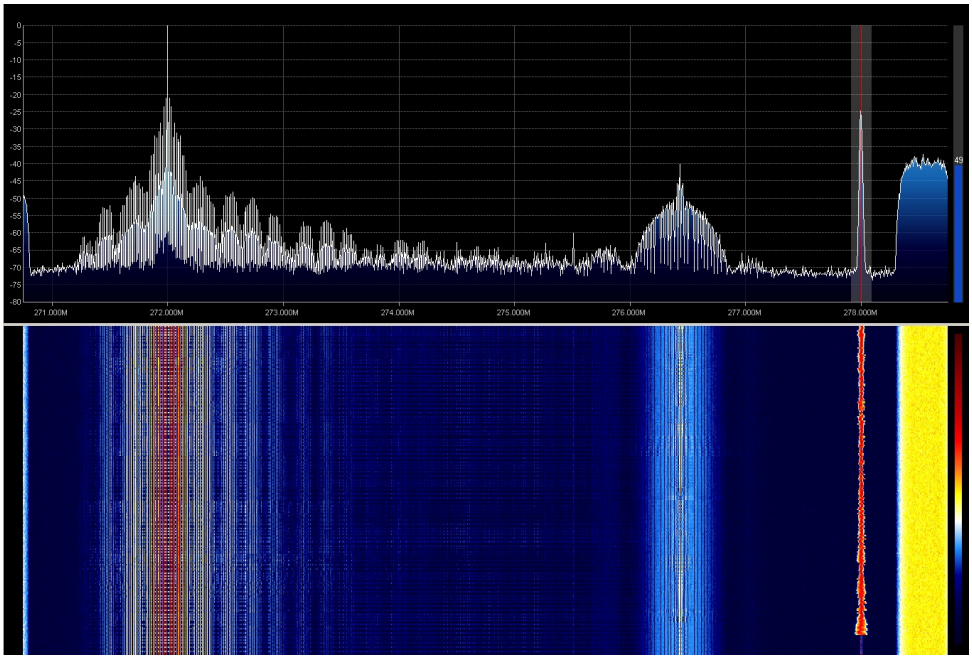
Teď, když známe jak rychlou Fourierovu transformaci, tak i výpočet výkonového spektra, můžeme si krátce vysvětlit, co je to vlastně waterfall spektrum a s tím související spektrogram. Pokud provedeme rychlou Fourierovu transformaci signálu a poté získáme výkonové spektrum a s tím i amplitudové spektrum, nastává otázka, jak tento výsledek zobrazit. K tomu nám slouží spektrogram, který je v některých případech zvaný také waterfall graf.

Spektrogram je vizuální reprezentace toho, jak se spektrum frekvencí signálu mění v čase. Spektrogramy jsou široce používány v oblasti hudby, lingvistiky, sonaru, radaru, zpracování řeči, seismologie a dalších. Běžným formátem spektrogramu je graf se dvěma osami, jedna osa představuje čas a druhá osa představuje frekvenci. Třetí rozměr udávající amplitudu nebo velikost konkrétní frekvence v konkrétním čase je reprezentována intenzitou nebo barvou každého bodu na grafu.

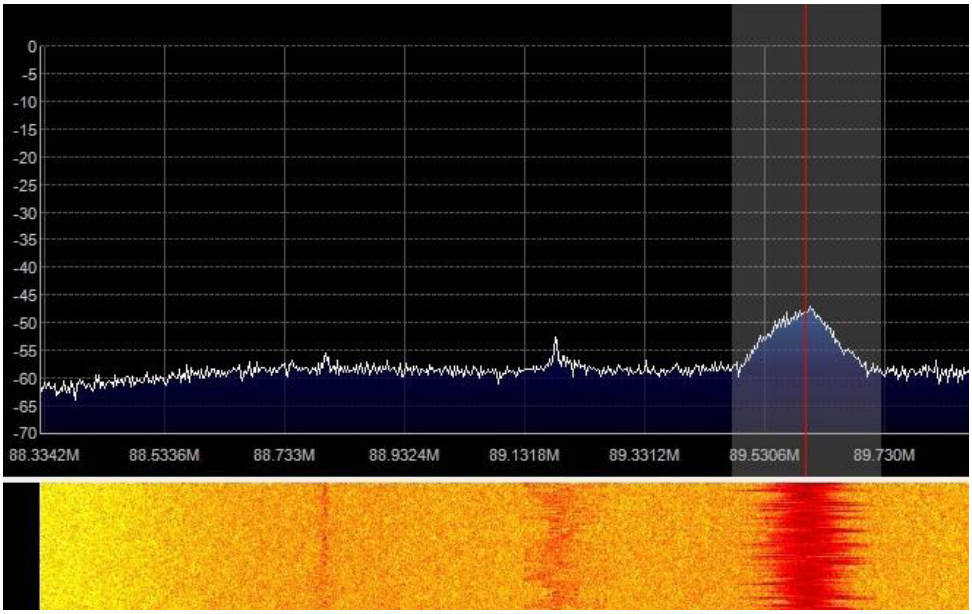
Spektrogram neobsahuje žádné informace o fázi signálu. Z tohoto důvodu není možné proces obrátit a ze spektrogramu vytvořit kopii původního signálu.

Existuje mnoho variant grafu. Často se prohazují vertikální a horizontální osy, někdy je amplituda nebo velikost reprezentována třetí osou ve 3D grafu, namísto barvy nebo intenzity. Osy frekvence a amplitudy mohou být lineární nebo logaritmické v závislosti na tom, k čemu se graf používá. Zvuk je obvykle reprezentován logaritmickou osou amplitudy, v decibelech, a frekvence je lineární pro zdůraznění harmonických vztahů nebo logaritmická pro zdůraznění hudebních, tónových vztahů. [4]

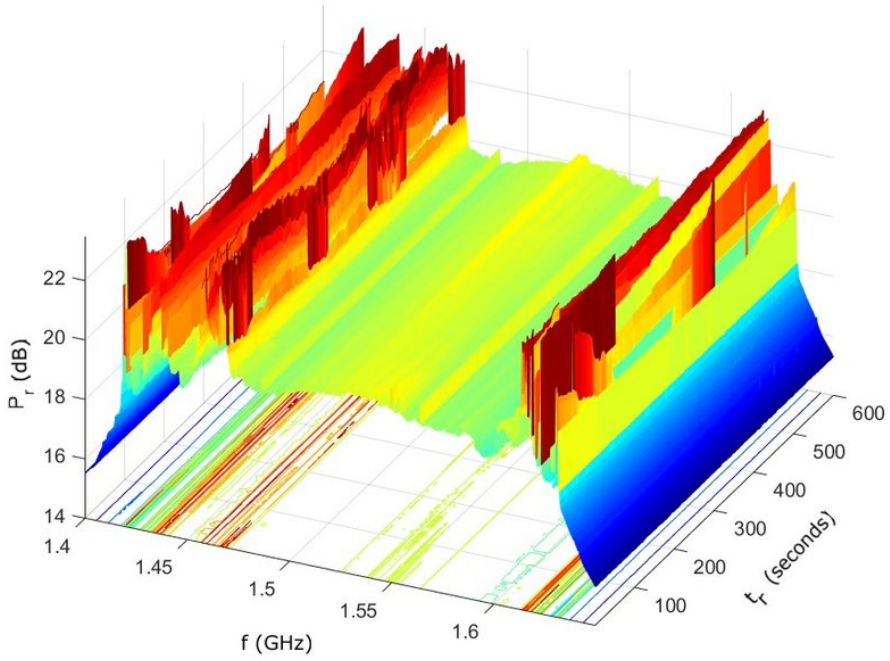
Dále uvádím některé příklady waterfall grafů:



Obrázek 3.9: Spektrum (nahoře) a waterfall graf (dole) televizního signálu PAL-I o šířce 8 MHz. [4]



Obrázek 3.10: FFT spektrum signálu FM vysílání a jeho waterfall graf. Na horizontální ose je frakvence a na vertikální amplituda. [5]



Obrázek 3.11: Waterfall graf slunečních radiových vzplanutí. [6]

Část II

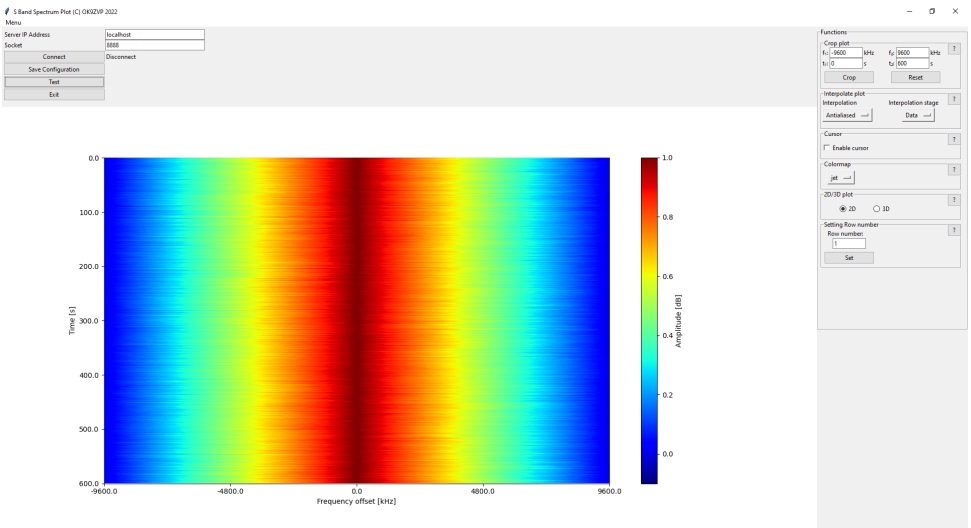
Praktická část

Kapitola 4

Program pro zobrazení Waterfall spektra

Cílem této práce bylo vytvořit program, který dokáže Zobrazovat waterfall spektrum na počítači. V této kapitole popíšu, jaké byly požadavky a jestli jsem je splnil. Také popíšu celý program a ukážu některé způsoby využití.

Na následujícím obrázku, číslo 4.1, můžeme vidět, jak ve výsledku celý program vypadá.



Obrázek 4.1: Celý program s vykreslenými fiktivními hodnotami.

4.1 Požadavky

Požadavek byl vytvořit program pro zobrazovaní přijímaných dat z SDR přijímače a zobrazovat je v grafu. Přes TCP/IP mi přicházejí data z SDR přijímače. Tyto data jsou již částečně zpracované a mým úkolem bylo je v reálném čase zobrazit a přidat různé funkce pro snadnější prohlížení spektra a manipulaci s ním. Všechny jednotlivé funkce budou popsány dále v kapitole.

4.2 Zvolený programovací jazyk

K vytvoření programu jsem použil programovací jazyk Python. Požadavek byl jen aby byl program schopný pracovat na Windows a dokázal dostatečně rychle zpracovávat data, což Python zvládá bez problémů.

Python je univerzální programovací jazyk. Filozofie jeho designu klade důraz na čitelnost kódu s použitím výrazného odsazování. Jeho jazykové konstrukce stejně jako jeho objektově orientovaný přístup mají za cíl pomoci programátorům napsat jasný, logický kód pro malé a velké projekty. Python se trvale umisťuje jako jeden z nejpopulárnějších programovacích jazyků.

První verze Pythonu byla vydaná už v roce 1991 Guidem van Rossem jako nástupce programovacího jazyka ABC. Python 2.0 byl vydán v roce 2000 a Python 3.0 byl vydán v roce 2008 a byl první revizí jazyka, která již není zcela zpětně kompatibilní. Python 2 byl ukončen s verzí 2.7.18 v roce 2020. [14]

Namísto toho, aby byly všechny funkce zabudovány do jeho jádra, byl Python navržen tak, aby byl vysoce rozšiřitelný (pomocí modulů). Díky této kompaktní modularitě je obzvláště populární jako prostředek pro přidávání programovatelných rozhraní ke stávajícím aplikacím. Van Rossumova vize byla vytvořit malý základní jazyka s velkou standardní knihovnou a snadno rozšiřitelným interpreterem.

Python se snaží o jednodušší, přehledný syntax a gramatiku a zároveň dává vývojářům možnost volby v metodologii kódování. [15]

Hlavní výhody Pythonu oproti jiným programovacím jazykům:

- Python funguje na různých platformách (Windows, Mac, Linux, Raspberry Pi apod.)
- Python má jednoduchou syntaxi podobnou angličtině.
- Python má syntaxi, která umožňuje vývojářům psát programy s méně řádky než některé jiné programovací jazyky.
- Python běží na interpretačním systému, což znamená, že kód lze spustit, jakmile je napsán. To znamená, že prototypování může být velmi rychlé.
- Pythonový kód může být zpracován procedurálním způsobem, objektově orientovaným způsobem nebo funkčním způsobem. [15]

Python je vyvíjen pod open-source licencí schválenou OSI, díky čemuž je volně použitelný a šiřitelný, a to i pro komerční použití. Licence Pythonu je spravována nadací Python Software Foundation. [14]

4.2.1 Použité moduly

Jak už jsem popsal, Python je snadno rozšiřitelný různými moduly. Pro vytvoření programu jsem jich nepoužil mnoho. Ale budu se snažit je v práci všechny jednoduše popsat a napsat proč jsem si je zvolil. Některé menší moduly popíšu až při popisování jednotlivých částí programu.

Tkinter

Nejzákladnějsším použitým modulem byl Tkinter, což je jednoduchý modul pro vytváření uživatelského rozhraní.

Tkinter je open source, přenosná knihovna grafického uživatelského rozhraní (GUI) navržená pro použití v Pythonovým kódu. Tkinter spoléhá na knihovnu Tk, knihovnu GUI používanou Tcl / Tk a Perl, která je zase implementována v C. Proto lze říci, že Tkinter je implementován pomocí více vrstev. [16]

Výhody Tkinter oproti jiným modulům pro grafické rozhraní: [16]

- Vrstvený přístup použitý při navrhování Tkinter poskytuje Tkinteru všechny výhody již vymyšlené a testované knihovny TK.

- Naučení se používání Tkinter je velmi intuitivní, a proto rychlé a bezbolestné. Implementace Tkinter skrývá detailní a komplikovaná volání v jednoduchých, intuitivních metodách.
- Python skripty, které používají Tkinter, nevyžadují úpravy, aby byly přeneseny z jedné platformy na druhou. Tkinter je dostupný pro jakoukoli platformu, pro kterou je implementován Python.
- Tkinter je nyní součástí každé distribuce Pythonu. Ke spouštění skriptů pomocí Tkinter proto nejsou potřeba žádné doplňkové moduly.

Tkinter jsem vybral, protože je již součástí distribuce Pythonu a práce s ním mi přijde velmi a jednoduchá a intuitivní.

Matplotlib

Matplotlib je komplexní vykreslovací knihovna pro vytváření statických, animovaných a interaktivních vizualizací v Pythonu. Byla vytvořena pro programovací jazyk Python a jeho numerickou matematickou nadstavbu NumPy. Poskytuje objektově orientované API (Application Programming Interface) pro vkládání grafů do aplikací pomocí uživatelských rozhraní, jako je Tkinter. Matplotlib byl původně napsán jako open source alternativa pro MATLAB. Ale v dnešní době je jeho objektovně orientované API přizpůsobitelnější a výkonnější než jeho starší modul pyplot, který poskytuje rozhraní podobné MATLABu. Matplotlib byl navržen tak, aby byl stejně účelný jako MATLAB, s možností použití pomocí Pythonu a výhodou, že je zdarma a open source. [17]

K dispozici je několik toolkitů, které rozšiřují funkčnost Matplotlib. Já jsem v progrmu použil jen toolkit Mplot3d, pro vytváření 3D grafu.

Já jsem použil matplotlib, protože je to jeden z největších a nejrozšířenějších modulů pro vytváření grafů v Pythonu.

4.2.2 Numpy

NumPy je knihovna pro Python, která přidává podporu pro velká, vícerozměrná pole a matice a přidává velkou sbírku matematických funkcí pro

výpočty v těchto maticích. Numpy je povinná závislost pro matplotlib, který používá funkce numpy pro numerická data a vícerozměrná matice. [18]

4.3 Zdroj signálu

Přesto, že je v zadání práce zadáno, že má být signál přijímán ze sériové linky, po předchozí dohodě s vedoucím práce bylo dohodnuto, že vstup signálu do programu bude realizován pomocí TCP-IP.

4.3.1 TCP/IP

TCP/IP je sada protokolů, sloužící ke komunikaci v počítačové síti a propojení dvou počítačů. Základ tohoto modelu tvoří protokoly TCP (Transmision Control Protocol) a IP (Internet Protocol). Model TCP/IP poskytuje propojení mezi dvěma koncovými uzly a specifikuje, jak budou data rozděleny na pakety, označeny adresami, přenášeny, směrovány a přijímány v cílovém bodě.

4.4 Využití programu

Program dokáže zobrazovat již zpracované amplitudové spektrum v reálném čase. Program je určený k zobrazení spektra různých radiových, komunikačních signálů nebo i slunečních radiových vzplanutí, ale protože data přicházejí přes TCP/IP je možné přijímat i jiná data ve stejném formátu. V sekci dále uvádím, proč je užitečné některé tyto signály sledovat.

4.4.1 Sluneční erupce

Sluneční erupce a vzplanutí představují nejsilnější procesy uvolňování energie v celé Sluneční Soustavě. Poruchy spojené s těmito jevy, jako jsou magnetická mračna, paprsky urychlených částice, zesílené záření (v rentgenových, UV a rádiových pásmech) šířící se ze Slunce přes meziplanetární prostor, jsou nyní známé jako efekty vesmírného počasí. V případě dosažení Země a jejího

okolí, představují vážné riziko pro technologie. Různé typy vesmírného počasí mohou ovlivnit různé technologie na Zemi. Sluneční vzplanutí mohou produkovat silné rentgenové záření, které degraduje nebo blokuje vysokofrekvenční rádiové vlny používané pro rádiovou komunikaci. Solární energetické částice (energetické protony) mohou proniknout elektronikou satelitu a způsobit elektrické selhání. Tyto energetické částice také blokují rádiovou komunikaci ve vysokých zeměpisných šířkách během slunečních bouří. Aby bylo možné podat deterministické předpovědi vesmírného počasí, musíme ho nejprve pochopit. Výzkum fyziky slunečních erupcí představuje klíč k budoucím předpovědím vesmírného počasí. [6] [19]

Nás zajímá hlavně efekt slunečních vzplanutí na radiových systémech. Sluneční radiové emise (zejména vysokoenergetické solární rádiové záblesky) mají velký vliv na navigační, komunikační, mobilní a IoT (internet věcí) systémy. Unikátní rádiové vzplanutí z prosince 2006 s špičkovou úrovní jednoho milionu SFU (jednotka slunečního toku) způsobila významnou redukci v poměru signál/šum (SNR) většiny přijímačů GPS, některé přijímače úplně ztratily informace o satelitech. Nicméně degradace výkonu může nastat také u nižších úrovní rádiových vzplanutí, například solární rádiová vzplanutí o intenzitě 100 000 SFU způsobují v běžném GPS přijímači s typickou teplotou hluku 150 K a ziskem antény 3 dB, pokles poměru signál/šum o 15 dB, což může přijímači zabránit přijímat nové signály. [6]

I přesto, že pozorování degradace SNR u satelitních systémů způsobená slunečními vzplanutími již byly zdokumentovány, podrobné informace o vlivu těchto pozorovaných jevů na výkon systému jsou ještě nedostatečné a mohou tedy být objektem budoucího výzkumu.

4.5 Popis programu

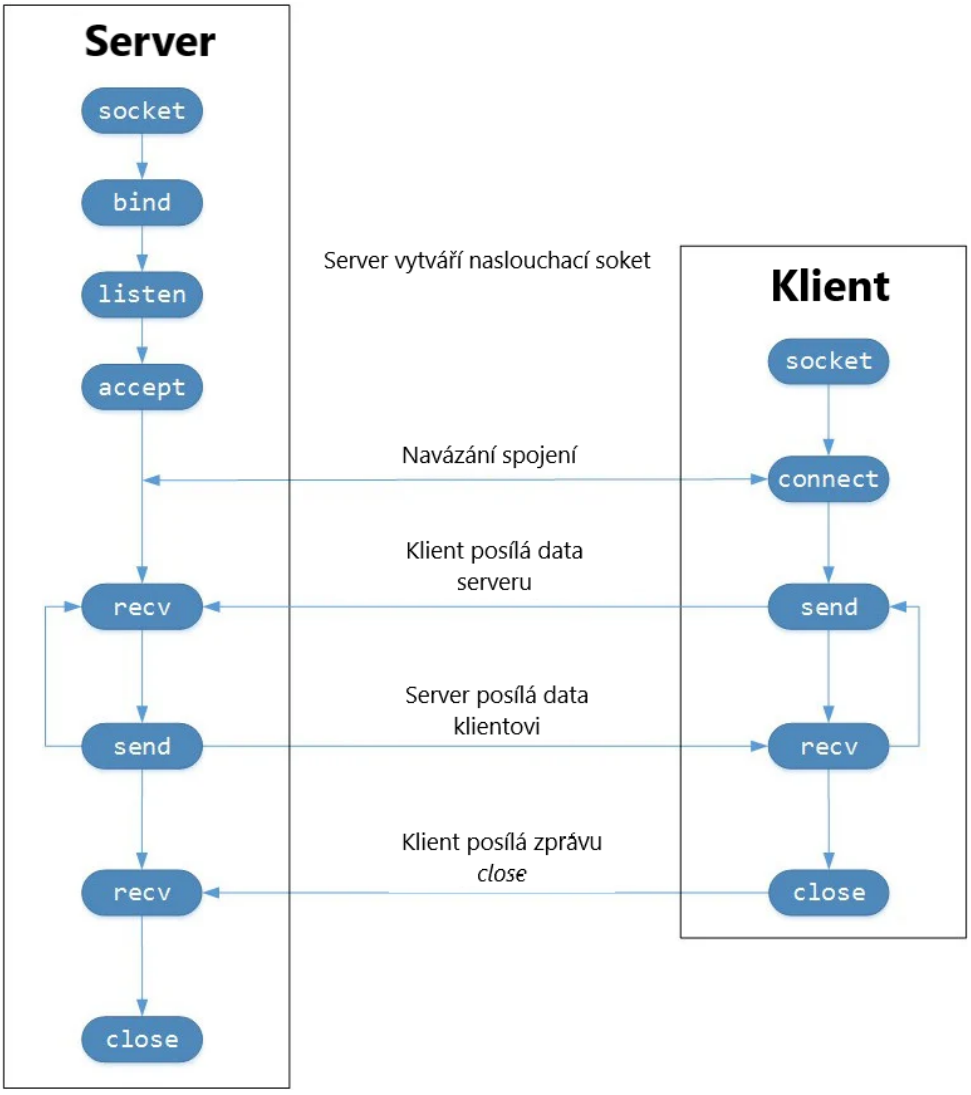
V této sekci popíšu jednotlivé části programu.

4.5.1 Přijímání signálu

V první řadě musíme mít způsob, jak signál přijmout. Základ této části programu mi pan vedoucí dodal dle jeho specifikací. Jak již bylo řečeno signál přichází přes TCP/IP. Po vyplnění IP adresy a soketu serveru, odkud přicházejí data, stačí stisknout tlačítko *Connect*. Tím se spustí funkce, která

se pomocí modulu *Socket* s tímto serverem spojí. Pomocí *socket.socket()* byl vytvořen objekt soketu a určen typ soketu jako *socket.SOCK_STREAM*. Díky čemuž se určí výchozí protokol jako TCP. [7]

Na digramu níže můžeme vidět jak modul *Socket* funguje:



Obrázek 4.2: Vývojový diagram *TCP soketu* [7] [8]

Na začátku API zavolá server, aby nastavil „naslouchací“ (*listen()*) soket. Tento soket naslouchá pro příchozí spojení od klienta. Když se klient připojí, server zavolá *accept()* k přijetí připojení. Klient zavolá *connect()* k navázání spojení se serverem. Data se vyměňují mezi klientem a serverem pomocí volání *send()* a *recv()*. [7]

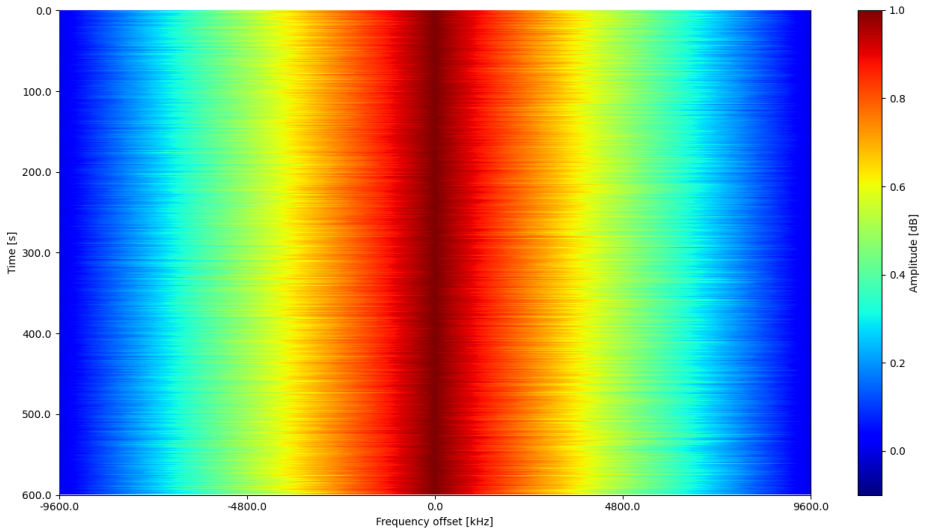
Poté, co se spojí program se serverem, zavolá se funkce, která bude v

určeném časovém intervalu přijímat data. Jako první se najde začátek dat a zkontroluje se, že přišly data správné délky. Až poté se můžou začít zpracovávat a vykreslovat. Před vykreslením se provede FFT shift, což je matematická operace, která posune nulové frekvenční složky signálu do středu spektra.

4.5.2 Zobrazování grafu

Samotné zobrazení grafu je nastavené pomocí modulu Matplotlib. První se vytvoří okno pomocí *figure*, do kterého se bude graf vykreslovat. Data přicházejí po řádcích, které se postupně skládají do matice. Příchozí data by měly mít délku rovnající se mocnině dvou, protože by data měly, už před příchodem, projít rychlou Fourierovou transformací. Tyto data se poté vykreslí jako obrázek, s tím, že každý bod má nějakou hodnotu, kterou mi dává velikost výkonového spektra v tom bodě. Tuto velikost zobrazuji jako barvu v dané barevné mapě.

Pro představu takový graf může vypadat takto:



Obrázek 4.3: Fiktivní waterfall graf s náhodným hodnotami.

Všimněme si, že horizontální osa má hodnoty mezi −9600 a 9600 kHz, i přesto, že jsem psal, že by data měly mít hodnotu, která se rovná mocnině dvou. Je to díky tomu, že v tomto případě má sice osa jen 1024 bodů, ale tyto body odpovídají danému frekvenčnímu offsetu. Toto je dáno už vlastnostmi přicházejícího signálu a je třeba to tedy vědět ještě před tím, než se začne graf vykreslovat.

Nastavení parametrů

První věc, než začnu vkreslovat graf, je třeba znát vlastnosti přicházejícího signálu. V okně *Options* se dají nastavit všechny důležité parametry. Tyto parametry jsou:

- Délka spektra - délka značí to, kolik bodů má každý přijatý řádek. Jak již bylo řečeno, tato hodnota by měla být v mocninách dvou.
- Hloubka spektra - hloubka značí to, kolik řádků spektra zobrazuji.
- Offset frekvence - říká v jakých hodnotách je přicházející frekvence.
- Vzorkovací perioda - určuje, jak často se data přijímají.
- Minimální a maximální amplituda - říká v jakých hodnotách se objevuje příchozí amplituda, toto je potřeba, abychom věděli, jaký bude rozsah barevné mapy.

Všechny tyto parametry je třeba nastavit, ještě před tím, než začnu přijímat data, jinak nebude graf ukazovat správné hodnoty.

4.5.3 Výřez grafu

Další funkcí je výřez grafu. Ořezávání je uděláno jednoduše tak, že jsem vytvořil čtyři okna, do kterých se dají zadat hodnoty, ve kterých se má graf oříznout. Samotná matice, de které se přijímají data se nezmění, jen se vykresluje jen její část.

Jako první si musím převést hodnoty zadaného frekvenčního offsetu do hodnoty délky spektra, abych ořízl graf ve správné hodnotě. To udělám tak, že si rozdělím hodnoty frekvenčního offsetu na posloupnost hodnot, jejichž počet je stejný jako délka spektra. Například pokud budu mít délku spektra 1024 a frekvenční offset od -9600 do 9600 kHz, vytvořím si posloupnost 1024 hodnot od -9600 do 9600 kHz. Poté najdu, ke které z těchto hodnot má moje zadaná hodnoty nejblíže. Když toto vím, můžu zjistit kolikátá tato hodnota je v posloupnosti, tedy jaké hodnotě délky spektra odpovídá a kde tedy graf oříznout. Například, pokud zadám hodnotu -9600, zjistím, že odpovídá hodnotě délky spektra 0. Frekvenční offset 0 Hz by odpovídal hodnotě délky spektra 511 a frekvenční offset 9600 Hz by odpovídal hodnotě délky spektra

1023. Také se musí po oříznutí grafu změnit všechny popisky grafu os, aby seděli pro nově zobrazený graf.

Řádky grafu budou vždy odpovídat času, tedy že každý řádek se rovná jedné sekundě. Ořezávání v časové ose se tedy nijak převádět nemusí.

Po kliknutí tlačítka *Reset* se celý graf vrátí do původní podoby.

4.5.4 Interpolace spektra

Protože data se vykreslují do grafu jako obrázek, může celkem jednoduše graf interpolovat. Obrázky jsou reprezentovány samostatnými pixely, buď vykreslené na obrazovce, nebo v souboru. Když data, která tvoří obrázek, mají jiné rozlišení než jejich zobrazení na obrazovce, může docházet k aliasingovým efektům. Jak jsou patrné, závisí na tom, jak moc se při změně rozlišení odehrává podvzorkování.

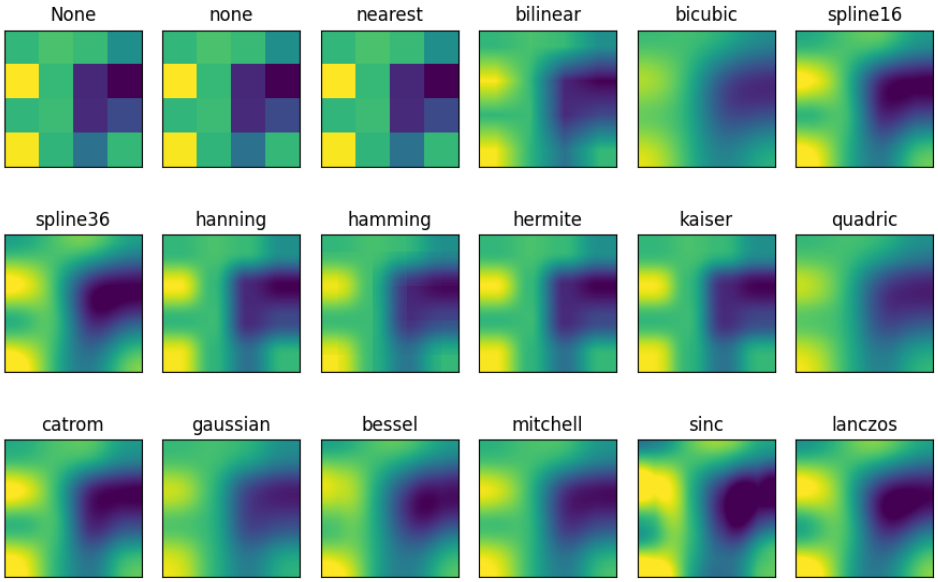
Při podvzorkování dat je aliasing redukován nejprve vyhlazováním a poté podvzorkováním vyhlazených dat. V Matplotlib se dá provést vyhlazení před mapováním dat do barev, nebo můžeme provést vyhlazení na datech RGB(A) ve finálním obrázku.

Výchozí interpolace obrázků v Matplotlib je "antialiased"a je aplikována na data. Tato funkce využívá hanningovou interpolaci dat pro snížení aliasingu ve většině situací. [20]

V programu se ve funkcích jen vybere, kterou interpolaci chceme použít a program jí rovnou aplikuje. Interpolací je na výběr mnoho, proto jsou pro představu na obrázku číslo 4.4 zobrazené některé z nich. Také je možné nastavit, jestli chceme interpolovat už data, před tím než se vykreslí, nebo jen interpolovat vykreslený graf, jako jakýkoli jiný obrázek. Tato funkce se nastavuje pod *Interpolation stage*.

4.5.5 Kurzor

Kurzor dělá jednoduše to, poté co se aktivuje, že zobrazí data z grafu v místě, kam uživatel klikne myší. Kurzor v programu je spojení dvou funkcí. Jedna



Obrázek 4.4: Přehled interpolačních metod. [9]

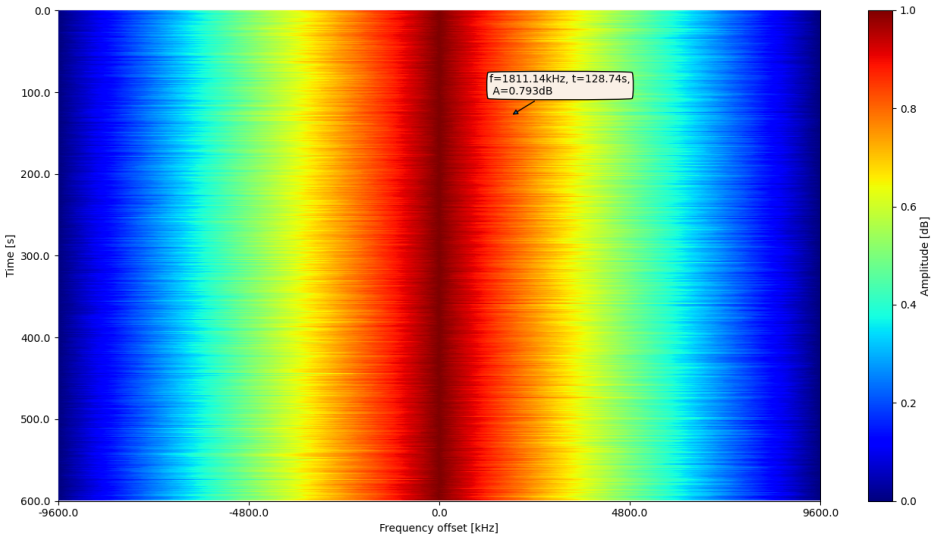
mi vytvoří malé okno, do kterého se vkládají zjištěná data. Toto jsem udělal pomocí anotací grafu. Pomocí anotace se dá umístit text na libovolnou pozici v grafu. Běžným případem je použití textu k popsání některých prvků v grafu. V anotaci je třeba vzít v úvahu dva body, umístění popisovaného bodu reprezentované argumentem *xy* a umístění textu *xytext*. [**?**]

A druhá část, která data z grafu získá. K tomu se používá funkce zvaná *Event handling*, což je API, která je součástí *Matplotlib*, pro interakci s *figure* pomocí stisknutí kláves a pohybů myši. Já jsem použil funkci *button_press_event*, která dokáže zjistit do jaké části grafu jsem myší kliknul. Stejně jako u výřezu grafu se musí data převést, protože jinak by, z horizontální osy, zobrazovali jen délku spektra. Data se poté jen zobrazí v anotaci. [21]

Anotace může ve výsledku vypadat jako na obrázku číslo 4.5.

4.5.6 Barevná mapa

Funkce nastavení barevné mapy dělá jen to, že specifikuje jakou by měl graf používat barevnou mapu. *Matplotlib* má řadu vestavěných barevných map. Smyslem výběru barevné mapy je nastavit dobrou a čitelnou reprezentaci amplitudy v grafu. V seznamu barevných map, v programu, jsou vybrané



Obrázek 4.5: Ukázka anotace kurzoru.

některé mapy, které by měly být vhodné pro zobrazení amplitudového spektra. To, kterou barevnou mapu si uživatel vybere závisí na osobní preferenci.

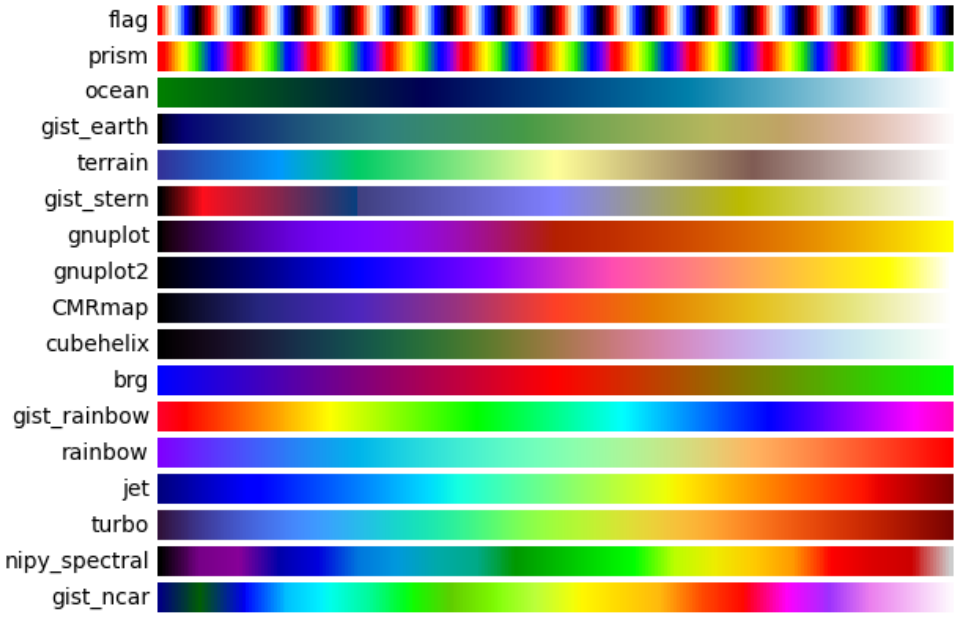
Na obrázku číslo 4.6 uvádím některé barevné mapy, které má *Matplotlib* k dispozici. Ne všechny jsou vhodné pro zobrazení amplitudového spektra, tudíž jsou v programu zahrnuty jen některé:

4.5.7 Dimenze grafu

V programu je doplněná i funkce pro nastavení, v jakých dimenzích se graf bude vykreslovat. B běžným případě nám stačí jen graf se dvěma dimenzemi. Tedy klasický spektrogram, kde je na jedné ose frekvenční offset, na druhé čas a amplituda je dána barevnou mapou. Graf je ale možné vykreslit i ve třech dimenzích, kdy se osy frekvenčního offsetu a času nemění, ale amplitudu vykresluji do třetí osy.

Vykreslování grafu se třemi dimenzemi je časově náročnější, proto není možné ho vykreslovat se stejnou *vzorkovací periodou* jako graf se dvěma osami. Všechny funkce, kromě kurzoru, které tady popisuji jdou použít jak na graf se třemi dimenzemi, tak pro graf se dvěma dimenzemi.

Na obrázcích číslo 4.7 a 4.8 můžeme vidět, jak vypadají grafy, se stejnými vstupními daty, ale různým počtem dimenzí.



Obrázek 4.6: Přehled barevných map. [10]

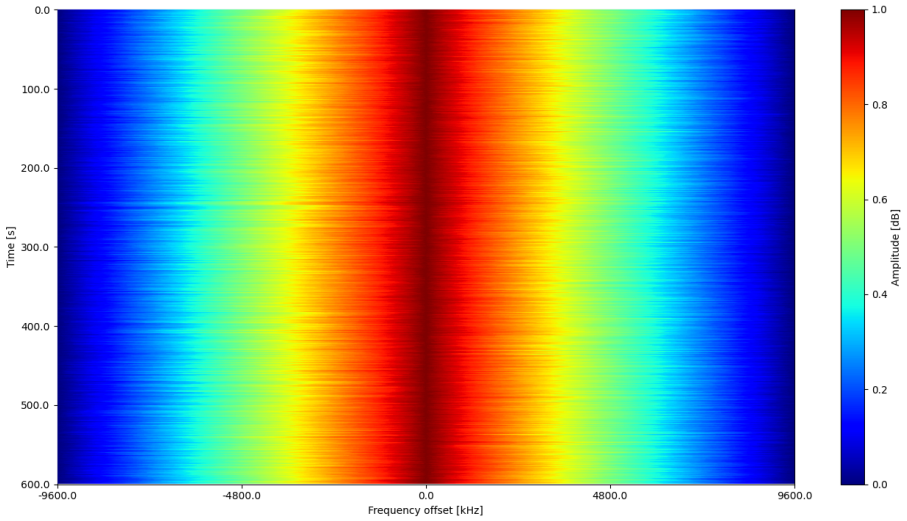
4.5.8 Počet vykreslených řádků

Další funkcí je něco, co jsem v programu popsal jako *Row number*. Toto číslo mi říká nic jiného, než jak často se má graf vykreslovat. Zatímco *Vzorkovací perioda* mi říká, jak často přijímám data, *Row number* říká po kolika přijatých řádcích se mají všechny najednou vykreslit.

Tato funkce existuje z jediného důvodu, a to z takového, že přijímání dat a skládání je do matice je mnohem rychlejší než tyto řádky vykreslovat. Zvýšením *Rown number* tedy udělám proces vykreslování rychlejší a budu méně zatěžovat výpočetní výkon počítače.

4.5.9 Ukládání

Přes funkci ukládání se dá graf uložit jako obrázek nebo rovnou jako soubor dat. Po kliknutí na *Menu* v horní liště programu se otevře výběr, ve kterém je možnost *Save as ...*. Po vybrání této možnosti se otevře okno *FileDialog*, což je modul obsažený v *Tkinter*. V okně se vyplní jméno souboru, kam a v jakém formátu ho chceme uložit. Program dokáže, buď ukládat graf jako obrázek v několika formátech, výslovně: *Jpeg, png, tiff, svg a rgba*, nebo ukládat graf přímo jako soubor hodnot do formátu *csv*.



Obrázek 4.7: Vykreslený graf fiktivních dat se dvěma dimenzemi.

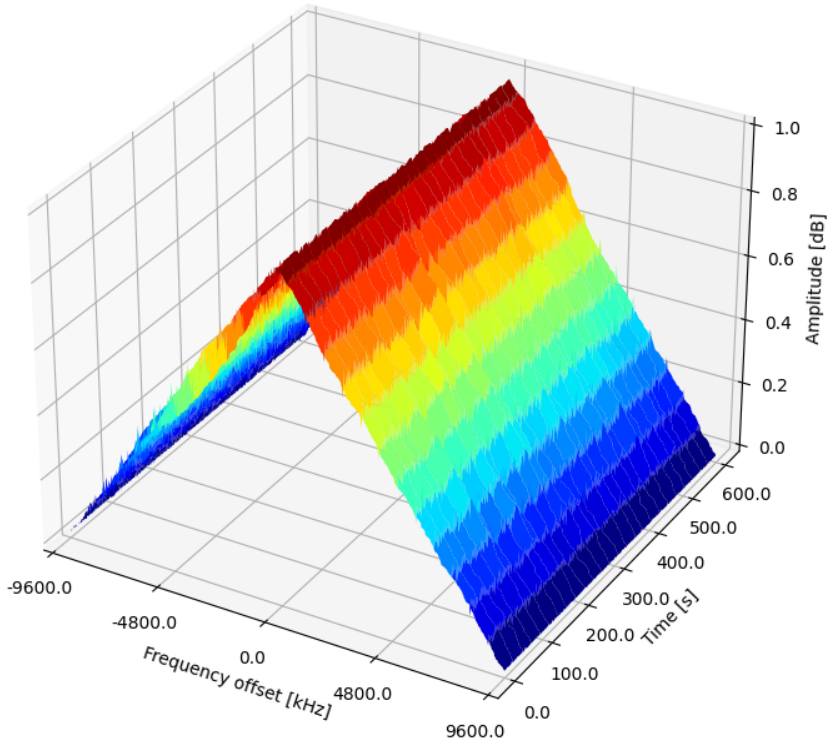
4.5.10 Nastavení

Důležitou funkcí je nastavení. V liště *Menu* je možnost *Options*, která otevře nové okno s nastavením parametrů. Tato funkce již byla z většiny popsána dříve v kapitole. Poté, co nastaví parametry stačí stisknout tlačítko *Set*, které vytvoří nový graf ve *figure* a hodnoty uloží jako proměnné.

Rozlišení grafu se třemi dimenzemi

Další funkce, která se dá nastavit uvnitř okna *Options* je rozlišení grafu se třemi dimenzemi. Jak jsem již zmínil, graf se třemi dimenzemi se vykresluje daleko pomaleji než graf se dvěma dimenzemi. Jeden způsob, jak tento čas zkrátit je snížit kvalitu vykreslovaného grafu.

V tomto nastavení se dají změnit hodnoty *Row count* a *Column count*. Tyto hodnoty představují maximální počet vzorků řádků a sloupců použitých při výpočtu grafu. Čím menší budou tato čísla, tím menší bude kvalita grafu, ale také rychlost vykreslování.



Obrázek 4.8: Vykreslený graf fiktivních dat se třemi dimenzemi.

4.5.11 Ukládání konfigurace

Teď, když známe nastavitelné parametry, můžeme si ukázat ještě jednu funkci. A to funkci ukládání konfigurace. Protože nechci muset nastavovat parametry pokaždé, když spustím program, je možné tyto parametry uložit souboru *INI*, což je konfigurační textový soubor, ve kterém tyto informace budou zapsány. Mimo parametrů se do souboru ukládají i informace pro *Socket*. Tedy IP adresa a adresa soketu.

Toto nám umožňuje modul *ConfigParser* už obsažený v *Pythonu*. Poté, co se parametry uloží pomocí tlačítka *Save Configuration*, se ve složce s kódem programu vytvoří soubor *INI*. Po příštím spuštění programu se tento soubor automaticky načte a nastaví parametry.

4.5.12 Testovací verze

Poslední funkce, která je nutná vysvětlit je něco, co jsem pojmenoval *Testing version*, tedy testovací verze programu. Jediné, co tato funkce udělá je, že vytvoří nové tlačítko *Test* v hlavní části programu. Po stisku tlačítka *Test* se podobně jako při stisku tlačítka *Connect* vykreslovat data do *figure*. Velký rozdíl je ale v tom, že tlačítko *Test* vykresluje jen náhodná data a ne data přijímaná z *TCP/IP*.

Tato data jsou velmi jednoduše generovaná. Jsou to jen data s délkou spektrální délky, která od nuly do poloviny své délky stoupají a poté do konce své délky klesají. To znamená, že mají v grafu okolo středu grafu největší hodnoty a po stranách nejmenší. Data tedy vzdáleně připomínají ty, která bychom měli reálně přijímat.

Celá tato funkce existuje jen pro to, aby bylo možné program testovat, i když nemáme přístup k reálným datům přes *TCP/IP*.

4.6 Vyhodnocení programu

Celý program je ve výsledku funkční. Problém byl s rychlostí vykreslování grafu. *Matplotlib* bohužel není určen pro rychlé vykreslování grafů, proto s tím v programu byly různé problémy. Snažil jsem se to co nejvíce kompenzovat, ale přesto je doba vykreslování poměrně dlouhá. Doba vykreslování grafu se třemi dimenzemi je velmi dlouhá. Ale i doba vykreslování jednoduššího grafu se dvěma dimenzemi je vcelku dlouhá. Na mám stolním počítači s těmito komponenty:

- Procesor - AMD Ryzen 5 1600 šesti-jádrový procesor s frekvencí 3.2 GHz
- Grafická karta - NVIDIA GeForce GTX 1650 SUPER
- RAM - 16 GB
- Disk - NVMe Samsung 980 SSD
- OS - Windows 10 Education

Při tomto výkonu počítače se data přijímají a zapisují do matice přibližně 1.5 ms. A vykreslení trvá přibližně 35 ms. V zadání práce bylo, že by měl

program být schopný vykreslovat 100 řádků za sekundu. Pokud je *Row number* dostatečně velké a vykresluje se jen každých několik sekund, bylo by být možné, na dostatečně výkonném počítači, rychlosti 100 řádků za sekundu dosáhnout. Bohužel na mém počítači jsem této rychlosti nedosáhl. Vykreslování grafu se třemi dimenzemi je ještě několika-násobně pomalejší (okolo 100 ms), proto je tento graf použitelný jen pokud se přijímají data s velkou *vzorkovací periodou*. Tedy například přijímat data jen jenom nebo dvakrát za sekundu.

Funkce, která volá, jak často se mají data přijímat, je napsaná ručně a běží na jiném vlákně než samotné uživatelské rozhraní. Funkce má možnost toho, že pokud funkce ještě vykresluje, ale přitom už se má zavolat znovu, po dokončení vykreslování se funkce zavolá několikrát za sebou, čímž se dožene čas. Pokud je tedy *Row number* moc nízké data se začnou hromadit a začne se vytvářet zpoždění.

4.7 Program v příloze

Program je ve příloze v archivované složce a zároveň, ve druhé příloze, jako samostatný kód. Ve složce je vložené celé virtuální prostředí, ve kterém je program spouštěn. Ve virtuálním prostředí je uložený nejen samotný kód programu a všechny použité moduly, ale i samotný potřebný interpreter programovacího jazyka Python. To znamená, že, pokud se program spustí přes tento interpreter, není nutné znovu stahovat použité moduly.

Kapitola 5

Závěr

Hlavním úkolem práce bylo vytvořit počítačový program, který dokáže přijímat data z TCP/IP a z přijatých dat vytvořit amplitudové spektrum, neboli waterfall graf. Věřím, že tohoto cíle se mi podařilo dosáhnout.

Bohužel má program i své problémy, jak jsem vypsal ve vyhodnocení programu. Určitě by dala vytvořit i lepší verze, která by dokázala data vykreslovat rychleji. Ale to by znamenalo velkou změnu implementace kódu. V budoucnu by mohl být program vylepšen dalšími funkcemi.

Při programování programu jsem se výrazně lépe naučil programovat v jazyce Python a v principech objektového programování. Při tvorbě programu jsem si musel zopakovat velkou část svých znalostí o programování a naučit se úplně nové věci.

Literatura

[1] John G. Proakis and Dimitris G. Manolakis. *Digital signal processing : principles, algorithms, and applications*. Prentice Hall, 1995.

[2] Afshin Samani. *An introduction to signal processing for non-engineers*. CRC Press/Taylor & Francis Group, 2020.

[3] G. D. Bergland. A guided tour of the fast fourier transform. *IEEE Spectrum*, 6(7):41–52, 1969.

[4] Wikipedia contributors. Spectrogram, 2021.

[5] Md Mamunoor Islam. A practical approach to spectrum analyzing unit using rtl-sdr. 2016.

[6] Pavel Puricer, Pavel Kovář, and Miroslav Barta. Modernized solar radio spectrograph in the l band based on software defined radio. *Electronics*, 8:861, 2019.

[7] Nathan Jennings. Socket programming in python (guide). Accessed: 2021-12-28.

[8] Wikipedia contributors. Berkeley sockets, 2021.

[9] Interpolations for imshow. Accessed: 2022-01-02.

[10] Choosing colormaps in matplotlib. Accessed: 2022-01-02.

[11] Wikipedia contributors. Fourier series, 2021.

[12] Wikipedia contributors. Fourier transform, 2021.

[13] Jonah Gamba. Radar signal processing for autonomous driving. In *Radar Signal Processing for Autonomous Driving*. Springer, 2020.

[14] Wikipedia contributors. Python (programming language), 2021.

[15] Python introduction. Accessed: 2022-01-02.

[16] Wikipedia contributors. Tkinter, 2021.

[17] Wikipedia contributors. Matplotlib, 2021.

[18] Wikipedia contributors. Numpy, 2021.

[19] Space weather impacts. Accessed: 2021-12-28.

[20] Image antialiasing. Accessed: 2022-01-02.

[21] Event handling and picking. Accessed: 2022-01-02.

[22] Vratislav Davídek and Pavel Sovka. *Číslicové zpracovaní signalů a implementace*. ČVUT, 1996.

[23] Alan Oppenheim and Alan Willsky. *Signals & systems*. Prentice Hall, 1997.

[24] E Brigham. *The fast Fourier transform and its applications*. Prentice Hall, 1988.

[25] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[26] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Petrýdes** Jméno: **Patrik** Osobní číslo: **461791**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Letectví a kosmonautika**

Studijní obor: **Avionika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Zobrazení Waterfall spektra na PC

Název diplomové práce anglicky:

Display of Waterfall spectrum on PC

Pokyny pro vypracování:

Cílem projektu je vyvinout software pro zobrazení amplitudového spektra na PC ve formě Waterfall pro účely monitorování družicových komunikačních signálů, detekci rušení GNSS a další aplikace.
Vstup spektra bude z SDR přijímače prostřednictvím sériové linky. Software zároveň musí umět spektrum ukládat a uložené spektrum prohlížet. Software musí pracovat v reálném čase na běžném PC. Musí být schopný zobrazit spektrum o maximální délce min. 16384. 100 řádků za sekundu. Software musí umět provádět decimaci a interpolaci spektra. Uživatelské rozhraní musí podporovat nastavení barevného profilu, amplitudového měřítka, zobrazit výřez apod.

Seznam doporučené literatury:

Proakis, J. G.; Monolakis, D. G.: Digital signal processing, 4th. Prentice-Hall, 1996.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Dr. Ing. Pavel Kovář, katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **17.09.2021** Termín odevzdání diplomové práce: **04.01.2022**

Platnost zadání diplomové práce:
do konce zimního semestru 2022/2023

doc. Dr. Ing. Pavel Kovář podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta