ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

Katedra aplikované elektroniky a telekomunikací

DIPLOMOVÁ PRÁCE

Vícestavové rozmítané modulace

Jan Hošek **2016**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jan HOŠEK**
Osobní číslo:	**E14N0098P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Telekomunikační a multimediální systémy**
Název tématu:	**Vícestavové rozmítané modulace**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Popište jednotlivé způsoby vytváření více stavových rozmítaných modulací.
2. Realizujte základní modulátor a demodulátor rozmítané modulace pro nejméně 64 stavový signál v akustickém kmitočtovém pásmu.
3. Analyzujte chybovost demodulace v AWGN kanálu a v AWGN kanálu s úzkopásmovým rušením.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah kvalifikační práce:	**40 - 60 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Seznam odborné literatury:	

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

Vedoucí diplomové práce:	**Ing. Ivo Veřtát, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Datum zadání diplomové práce:	**15. října 2015**
Termín odevzdání diplomové práce:	**16. května 2016**

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

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. října 2015

Abstrakt

Práce je zaměřená na realizaci vícestavových rozmítaných modulací v akustickém pásmu. V práci jsou uvedeny dvě metody rozmítané modulace. Jedná se o metodu půlení frekvenčního pásma a půlení časového intervalu. Modulační symbol u rozmítaných modulací je reprezentován změnou frekvence a konstantní amplitudou. Vícestavové rozmítané modulace jsou zde řešeny, protože spadají do skupiny modulací s rozprostřeným spektrem, konstantní modulační obálkou a při zvyšování počtu stavů i vysokou energetickou účinností danou dobrými korelačními vlastnostmi frekvenčně rozmítaných signálů. Těchto vlastností je s výhodou využito ve chvíli, kdy komunikační systém satelitu nemá dostatek energie. Konstantní modulační obálka umožňuje použití nelineárních zesilovačů s vysokou energetickou účinností. Celkově tak mohou být tyto modulace perspektivní pro komunikační systémy malých satelitů.

Řešením diplomové práce je funkční prototyp softwarového modulátoru a demodulátoru s možností nastavení šířky přenosového pásma, počtem stavů modulace a doby trvání jednoho modulačního symbolu pracujícím v reálném čase v akustickém pásmu Takovéto řešení umožňuje jejich přímé navázání na modulační vstupy a výstupy běžné radiokomunikační techniky a jejich odvysílání a příjem. Je zde ověřena závislost mezi kvalitou přijímaného signálu a chybovostí přenosu v AWGN kanálu a v AWGN kanálu s úzkopásmovým rušením při různém počtu stavů.

Klíčová slova

rozmítané modulace, rozprostřené spektrum, satelitní komunikace, více stavové modulace

HOŠEK, Jan. High order chirp modulations. [Vícestavové rozmítané modulace]. Pilsen, 2016. Master thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Ing. Ivo Veřtát, Ph.D.

Abstract

This thesis is focused on the realization of the M-ary chirp modulation in the acoustic range. The thesis is presented two methods of the chirp modulation. There is the method halving the frequency band and halving time interval. The modulation symbol of the chirp modulation is represented by a change of frequency and constant amplitude. The M-ary chirp modulations are solved here because they belong to the group spread spectrum modulation, constant amplitude and in increasing the number of states also high energy efficiency due to the good correlation properties of the frequency chirps signals. These properties are used when the satellite communication system hasn't enough energy. Constant modulation allows use the nonlinear amplifiers with the high energy efficiency. These modulations may be perspective for communication systems of the small satellites.

The solution of the diploma thesis is a functional prototype software modulator and demodulator with adjustable bandwidth, modulation number of states and the duration of one modulation symbol working in real time and in acoustic band. This solution allows the direct coupling to the modulation inputs and outputs ordinary radio communication technology and them transmission and receiving. There is verified dependence between the quality of the received signal and bit error rate in an AWGN channel and the AWGN channel with narrowband interference in a number of different states.

Key words

chirp modulation, spread spectrum, bit error rate

Prohlášení

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

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 16.5.2016 Jan Hošek

……..

Poděkování

Tato práce vznikla za podpory studentské grantové soutěžen SGS-2015-002.

Obsah

Úvod

V práci je řešena problematika realizace více stavových rozmítaných modulací v akustickém pásmu pro využití v energeticky úsporné a na interference odolné rádiové komunikaci se satelitem. U rozmítaných modulací je přenášená informace reprezentována lineární změnou frekvence pro každý přenášený modulační symbol. Jsou zde uvedeny dvě metody rozmítané modulace. V první metodě je základem pro dvoustavovou modulaci jeden pár lineárně rozmítaného signálu, kde je symbol 0 reprezentován kladnou hodnotou rozmítání, tedy nárůstem frekvence a symbol 1 zápornou hodnotou rozmítání, tedy poklesem frekvence. Nárůst nebo pokles frekvence je po celou dobu trvání modulačního symbolu konstantní. Vytvoření vícestavové modulace je docíleno rozdělením frekvenčního pásma na více sub-pásem. V druhé metodě je jeden modulační symbol rozdělen v polovině trvání modulačního symbolu, kde se mění rychlost rozmítání. Symbol 0 je reprezentován jinou kombinací rychlostí rozmítání v první a druhé polovině trvání symbolu než symbol 1. Vytvoření vícestavové modulace je docíleno vytvořením více odlišných rychlostí rozmítání.

Důvodem vytvoření vícestavové rozmítané modulace je ten, že spadá do skupiny modulací s rozprostřeným spektrem, konstantní modulační obálkou a při zvyšování počtu stavů i vysokou energetickou účinností danou dobrými korelačními vlastnostmi frekvenčně rozmítaných signálů. Ta je důležitá v situacích, kdy komunikační systém satelitu nemá dostatek energie. Celkově tak mohou být tyto modulace perspektivní pro komunikační systémy malých satelitů.

Pro realizaci modulátoru a demodulátoru včetně synchronizace byl použit Digital Signal Processing toolbox výpočetního prostředí Matlab. Při synchronizaci signálu se vychází z dobrých korelačních vlastností frekvenčně rozmítaného signálu, kdy k synchronizaci bude sloužit jeden modulační symbol na začátku každého přenášeného rámce, který bude hledán v posloupnosti dat. Od něj pak bude odpočítáván předem stanovený počet informačních symbolů samotné zprávy.

Výsledkem řešení diplomové práce je funkční prototyp softwarového modulátoru a demodulátoru pracujícím v reálném čase v akustickém pásmu s možností nastavení šířky přenosového pásma, počtem stavů modulace a doby trvání jednoho modulačního symbolu. Takovéto řešení umožňuje jejich přímé navázání na modulační vstupy a výstupy běžné radiokomunikační techniky a jejich odvysílání a příjem.

1 Teoretický rozbor

1.1 Teoretický rozbor rozpočtu radiového spojení

Pro stanovení výkonu užitečného signálu na vstupu přijímače pozemní stanice lze vyjít z rovnice 1.1, která zahrnuje veškeré ztráty a zisky podél cesty vysílaného signálu. Výsledná vypočtená hodnota užitečného přijímaného signálu, zde označena písmenem C vychází v jednotkách dB_W.

$$C = P_A - L_{RF} + G_{Ant1} - L_{Ant1} - L_0 - L_{Iono} - L_{Atm} - L_w + G_{Ant2} - L_{Ant2} - L_P - G_S \qquad (1.1)$$

Rovnice obsahuje mnoho členů, které zde budou podrobněji rozebrány. Prvním členem na pravé straně rovnice je P_A, který představuje výstupní výkon zesilovače na vysílací straně. Ztráty mezi zesilovačem a vysílací anténou jsou označeny jako L_{RF}. Zisk vysílací antény G_{Ant1}, je korigován o útlum vlivem nepřesného směrování vysílací antény L_{Ant1}. Dalším členem rovnice jsou ztráty šířením signálu ve volném prostředí L_0 udávané v dB. Člen L_{Iono} zahrnuje ztráty způsobené útlumem a odrazem signálu v ionosféře. Další ztráty L_{Atm} jsou způsobené absorpcí signálu v plynech a látkách atmosféry. Posledním členem, který vyjadřuje útlum signálu mezi vysílací a přijímací anténou je L_W, který v sobě zahrnuje ztráty způsobené deštěm, oblačností a troposférických scintilací. Člen, který do rozpočtu radiového spojení přináší zisk je zisk přijímací antény G_{Ant2}. Opět zde musí být zahrnuty ztráty nepřesným směrováním antény L_{Ant2}. Vzhledem k tomu, že nelze zaručit přesné natočení satelitu na oběžné dráze, které způsobí rozdílnou polarizaci dopadajícího signálu a přijímací antény, je třeba započítat do rozpočtu radiového spojení ještě polarizační ztráty L_P. Do rozpočtu radiového spojení, je zahrnuta i systémová rezerva.

- Výstupní úroveň výkonu zesilovače P_A [dB_W] - Vzhledem k omezenému množství energie, které lze získat pomocí solárních panelů, je třeba používat zesilovače s vysokou účinností. Aby bylo možné tyto zesilovače použít, je nutné používat takový typ modulace, která má konstantní modulační obálku.
- Ztráty mezi zesilovačem a vysílací anténou L_{RF} [dB] - Do tohoto členu jsou zahrnuty ztráty vložným útlumem mezi zesilovačem a vysílací anténou a ztráty způsobené impedančním nepřizpůsobením.
- Zisk vysílací antény G_{Ant1} [dBi] - U pikosatelitu je kvůli omezeným rozměrům a hmotnostního limitu obvykle absence systému, který by zajišťoval polohování

pikosatelitu. Z toho důvodu není možné použít vysoce směrové antény s velkým ziskem. Používají se tedy hlavně půlvlnné dipóly a monopóly, které mají nízký zisk.

- Ztráty nepřesným směrováním vysílací antény L_{Ant1} [dB] - Tyto ztráty vyplívají z absence polohovacího systémy, který by zajišťoval natočení pikosatelitu k pozemnímu segmentu. Vzhledem k tomu bude docházet k periodickým únikům, ve chvíli kdy bude anténa natočena k pozemnímu segmentu svým minimem vyzařovací charakteristiky. Tyto ztráty je ideální vyjádřit statisticky, kdy je vyhodnocena velikost poklesu signálu po 95 % nebo 99 % z celkové doby komunikace.
- Ztráty šířením signálu ve volném prostředí L_0 [dB] - Tyto ztráty umožňují vypočítat hodnotu užitečného přijímaného signálu C ze zadaného výstupního výkonu zesilovače P_A, zisku vysílací a přijímací antény G_{Ant1}, G_{Ant2} bez nutnosti znát efektivní plochu antény. Závisí na použité frekvenci a druhé mocnině komunikační vzdálenosti.
- Ztráty způsobené útlumem a odrazem signálu v ionosféře L_{Iono} [dB] - Mohou být významné pro kmitočty do 3 GHz a je třeba s nimi počítat do frekvence zhruba 12 GHz. Navíc se mohou během dne, ale i ročního období výrazně měnit.
- Ztráty v atmosféře L_{Atm} [dB] - Tyto ztráty jsou zejména způsobeny v dolních vrstvách atmosféry díky absorpci ve vodních parách
- Ztráty způsobené deštěm, oblačností a troposférických scintilací L_W [dB] - Velmi záleží na pozici pozemního segmentu a průměrnému ročnímu úhrnu srážek. Velikost útlumu bude značně narůstat s intenzitou srážek, proto je třeba znát maximální intenzitu deště. Ztráty v oblačnosti lze vyčíslit ze znalosti statistického modelu oblačnosti pro dané místo. Ztráty způsobené rychlými změnami v atmosféře jsou způsobeny změnou indexu atmosférické refrakce podél komunikační cesty.
- Zisk přijímací antény G_{Ant2} [dBi] - U pozemní stanice může být anténa mnohem rozměrnější a pomocí mechanického rotátoru nebo fázováním antény lze měnit polohu maxima vyzařování v závislosti na změně pozice pikosatelitu. Z těchto důvodů, zisk přijímací antény nabývá mnohem větších hodnot, než zisk vysílací antény pikosatelitu.
- Ztráty nepřesným směrováním přijímací antény L_{Ant2} [dB] - Jelikož přijímací anténa má mnohem užší hlavní lalok vyzařovací charakteristiky, i nepatrné vyosení maxima může znamenat velký pokles zisku přijímací antény.
- Polarizační ztráty L_P [dB] - Jsou opět důsledkem absence polohovacího systému pikosatelitu. Vzhledem k tomu, že pikosatelit může různě rotovat, je nejlepším řešením použití lineární polarizace u antény pikosatelitu a kruhové polarizace u antény

pozemního segmentu. Toto řešení vnáší trvalý útlum 3 dB, ale zamezuje mnohem větším ztrátám, které by nastaly při nevhodném natočení pikosatelitu, pokud by obě antény použily stejnou polarizaci. [2], [9]

- Systémová rezerva G_S [dB] - Systémová rezerva se zahrnuje do rozpočtu radiového spojení pro pokrytí dalších přídavných ztrát, které se mohou během přenosu objevit.

Ačkoliv není vysílán žádný signál, lze na vstupu přijímače naměřit určitou intenzitu signálu, kterou představuje šum. Spektrální výkonová hustota šumu lze vypočítat pomocí rovnice 1.2,

$$N_0 = 10 \cdot \log_{10}(k) + 10 \cdot \log_{10}(T_S) \tag{1.2}$$

kde k představuje Boltzmannovu konstantu a T_S celkovou šumovou teplotu systému v Kelvinech, která v sobě zahrnuje šum přijímací antény, napájecího vedení a šum přijímače. Spektrální výkonová hustota šumu je normována k šířce pásma 1 Hz.

Pro kvalitní příjem, je důležité vyjádřit odstup přijímaného užitečného signálu C od spektrální výkonové hustoty šumu N_0, který se získá pomocí rovnice 1.3. Tento způsob vyjádření nezávisí na typu použité modulace.

$$\frac{C}{N_0} = C\left[dB_W\right] - N_0\left[dB_W/Hz\right] \tag{1.3}$$

Při znalosti použitého typu modulace je známá modulační rychlost v_m , která vyjadřuje počet změn za sekundu je vyjádřena v jednotkách Baud [Bd]. Z modulační rychlosti a znalosti počtu stavů modulace, lze snad vyjádřit přenosovou rychlost v_p. Přenosová rychlost udává počet přenesených bitů za jednotku času a je rovna:

$$v_p = v_m \cdot \log_2(n) \tag{1.3}$$

Kde n je počet stavů modulace. Z přenosové rychlosti v_p [bit/s] a znalosti odstupu výkonu užitečného signálu od spektrální výkonové hustoty šumu C/N_0 [dBw/Hz] lze stanovit poměr E_b/N_0 [dB], který představuje energii potřebnou na přenesení 1 bitu informace při dosažené spektrální výkonové hustotě šumu.

$$\frac{E_b}{N_0} = \frac{C}{N_0} - 10 \cdot \log_{10}(v_p) \tag{1.4}$$

Vyjádřením poměru E_b/N_0 v rovnici 1.1 se získá rozpočet radiového spojení, vyjadřující, jak velká musí být energie jednoho bitu při spektrální výkonové hustotě šumu na přijímači, aby bylo možné přijatý signál demodulovat s požadovanou chybovostí.

$$\frac{E_b}{N_0} = P_A - L_{RF} + G_{Ant1} - L_{Ant1} - L_0 - L_{Iono} - L_{Atm} - L_W + \ldots \quad (1.5)$$

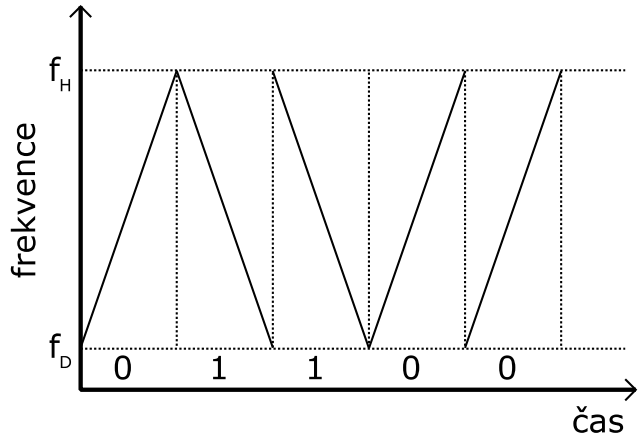
$$\ldots + G_{Ant2} - L_{Ant2} - L_P - G_S - N_0 - 10 \cdot \log_{10}(v_p)$$

Pokud vlivem vnějších okolností (např. nerozvinutá anténa satelitu, porucha napájení, vynucené omezení vysílaného výkonu kvůli podceněnému chlazení koncového stupně vysílače satelitu) přijímáme nižší E_b/N_0 a tedy s vyšší bitovou chybovostí, lze přejít na energeticky účinnější modulaci a dosáhnout původní požadované hranice chybovosti přenosu. Za tímto účelem jsou v práci řešeny vícestavové rozmítané modulace.

1.2 Dvoustavové rozmítané modulace

Rozmítané modulace patří do skupiny modulací s rozprostřeným spektrem, které mohou mít vysokou energetickou účinnost. Ta je důležitá v situacích, kdy komunikační systém nemá dostatek energie, nebo jsou během rádiové komunikace horší podmínky, než se očekávaly. Díky rozprostřenému spektru jsou tyto modulace odolné vůči selektivním únikům a úzkopásmovým interferencím. Patří také do skupiny modulací s konstantní modulační obálkou, díky čemuž je možné použít na zesílení nelineární zesilovač s vysokou účinností.

Rozmítané modulace jsou takové modulace, kde je modulační symbol reprezentován změnou frekvence. Změna frekvence může být lineární nebo exponenciální. V této práci je řešena modulace s lineární změnou frekvence a tak zde bude popsán pouze princip vytvoření tohoto typu modulace. Základní principy, jak vytvořit nebo jakým způsobem reprezentovat jednotlivé symboly, lze však využít i pro exponenciální změnu frekvence. Lineární dvoustavové rozmítané modulace jsou tvořeny pomocí lineární změny frekvence pro jednotlivé modulační symboly. Vytváření základních dvou modulačních symbolů 0 a 1 lze dosáhnout několika způsoby. První možností je zanechání celého frekvenčního pásma pro oba symboly 0 i 1. Rozlišení symbolů 0 a 1 je docíleno pomocí kladné a záporné hodnoty rozmítání nosné frekvence jednotlivých symbolů. Symbol 0 je reprezentován kladnou hodnotou rozmítání, tedy nárůstem frekvence a symbol 1 je reprezentován zápornou hodnotou rozmítání, tedy poklesem frekvence.



Obr. 1 Druhý způsob

Z obrázku 1 je vidět, že pro oba symboly 0 a 1 je stejná jak dolní a horní frekvence, tak i doba rozmítání. Symboly lze matematicky popsat pomocí rovnic 1.6 a 1.7.

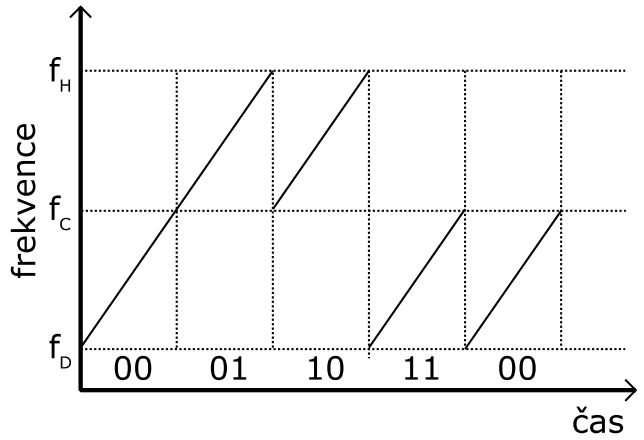
$$y_{up} = \sin\left[2 \cdot \pi \cdot \left(f_{down} \cdot t + \frac{k}{2} \cdot t^2\right)\right] \tag{1.6}$$

$$y_{down} = \sin\left[2 \cdot \pi \cdot \left(f_{up} \cdot t - \frac{k}{2} \cdot t^2\right)\right] \tag{1.7}$$

kde k je velikost rychlosti rozmítání frekvence, který lze vypočíst pomocí rovnice 1.8

$$k = \frac{f_{up} - f_{down}}{t} \tag{1.8}$$

Druhý způsob, jak vytvořit dva symboly je rozdělit frekvenční pásmo na dvě poloviny. Princip je zobrazen na obrázku 2.



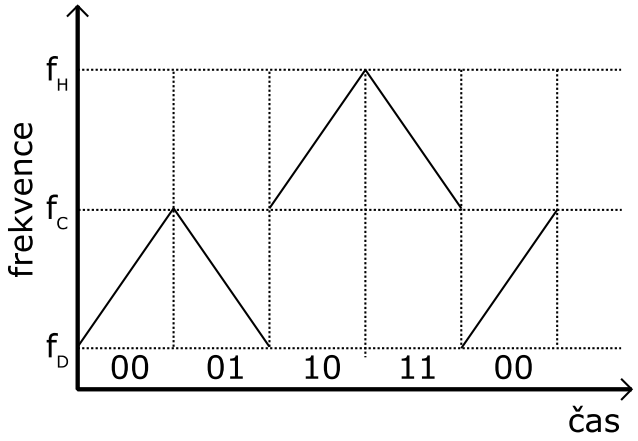
Obr. 2 První způsob vytvoření rozmítané modulace

Z obrázku 2 je vidět, že rozdělení frekvenčního pásma na dvě poloviny umožní vytvoření dvoustavové modulace s jedinou rychlostí frekvenčního rozmítání. Pro symboly 0 a 1 je rozdíl mezi koncovou a počáteční frekvencí stejný, což znamená, že mají stejnou hodnotu rychlosti

frekvenčního rozmítání. Jediný rozdíl je v umístění symbolů ve frekvenčním pásmu. Symbol 0 je reprezentován konstantním nárůstem frekvence od dolní mezní frekvence f_D do střední frekvence f_C. Symbol 1 je reprezentován stejným konstantním nárůstem frekvence od střední frekvence f_C do horní mezní frekvence f_H. Matematický popis modulace je formálně stejný s rovnicí 1.6.

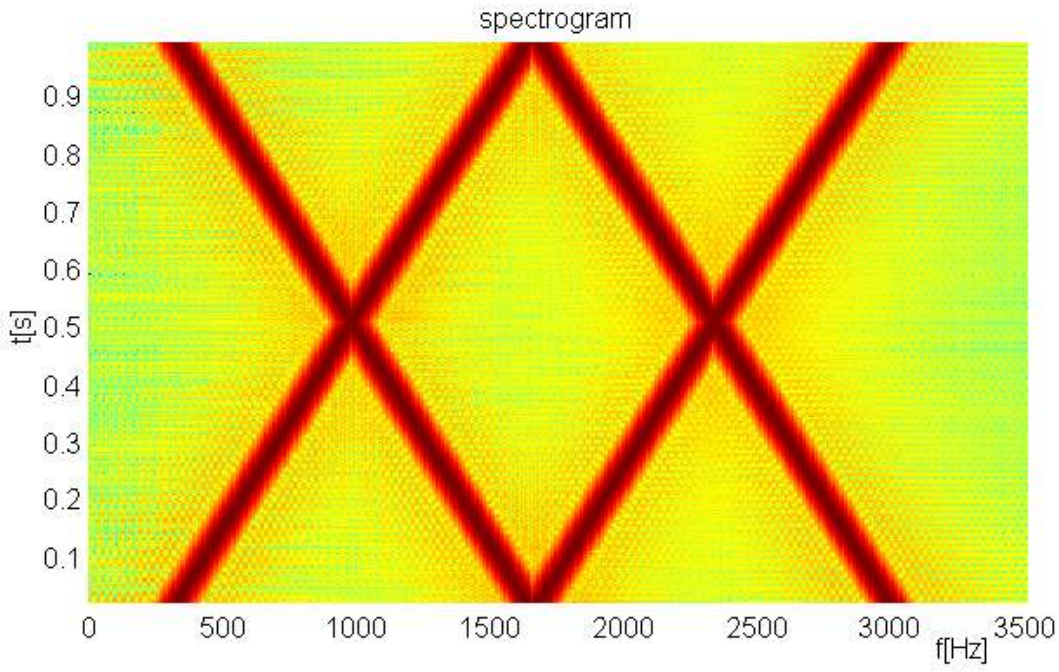
1.3 Vícestavové rozmítané modulace

Obecně lze vyjít z principu druhé metody. Rozdělením frekvenčního pásma na více subpásem vznikne odpovídající počet stavů. Skombinováním s první metodou, tedy že pro každé subpásmo bude jeden symbol s kladnou hodnotou rychlosti frekvenčního rozmítání a jeden symbol se zápornou hodnotou rychlosti frekvenčního rozmítání, se s každým dělením frekvenčního pásma zvýší počet stavů dvojnásobně. Princip je zobrazen na obrázku 3 pro čtyřstavovou modulaci.



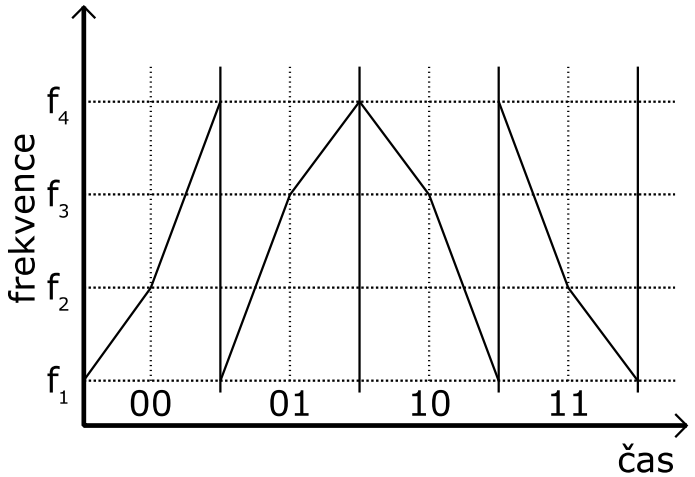
Obr. 3 Princip 4 - stavové rozmítané modulace

Symboly 00 a 10 jsou reprezentovány kladnou hodnotou rychlosti frekvenčního rozmítání a symboly 01 a 11 jsou reprezentovány zápornou hodnotou rychlosti frekvenčního rozmítání. Rozpůlením obou subpásem by vznikly 4 subpásma, čemuž by odpovídalo 8 stavů. Na obrázku 4 je zobrazen spektrogram pro čtyřstavovou modulaci se všemi symboly zakreslenými do stejné časové pozice, který dává lepší představu o tom, jak jsou jednotlivé modulační symboly frekvenčně umístěny.



Obr. 4 Spectrogram 4 - stavové modulace

Další možností je vytvoření více možných symbolů využitím rozdílné rychlosti rozmítání frekvence. Z toho vychází druhá metoda, kdy se modulační symbol rozdělí v polovině doby trvání modulačního symbolu, kde se změní rychlost rozmítání frekvence. Jeden modulační symbol je tedy reprezentován dvěma odlišnými rychlostmi rozmítání frekvence, které se mění v polovině trvání modulačního symbolu. Všechny symboly obsahují frekvence z celého frekvenčního pásma, každý symbol má však jedinečnou dvojici rychlostí rozmítání frekvence. Vytvoření více stavové modulace je docíleno vytvořením více odlišných rychlostí rozmítání frekvence s opětovným využitím kladné i záporné hodnoty rozmítání. Princip pro čtyřstavovou modulaci je zobrazen na obrázku 5.



Obr. 5 Princip 4 - stavové rozmítané modulace

Symboly lze matematicky popsat pomocí rovnic 1.10 a 1.11,

$$y_{K1} = \sqrt{\frac{2 \cdot E}{T}} \cdot \cos\left(\omega_c \cdot t - \pi \cdot \alpha_K \cdot t^2\right) \tag{1.10}$$

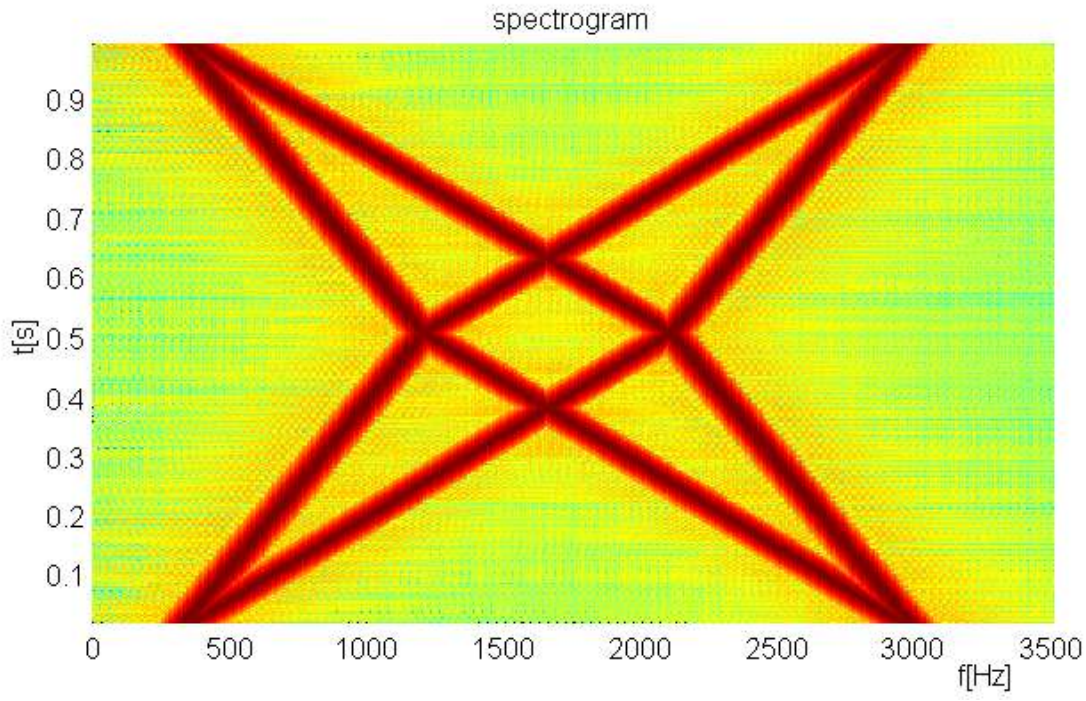
$$y_{K2} = \sqrt{\frac{2 \cdot E}{T}} \cdot \cos\left(\omega_c \cdot t + \pi \cdot \alpha_K \cdot t^2\right) \tag{1.11}$$

$$\alpha_K = \frac{M \cdot \Delta f}{T/2} \tag{1.12}$$

$$\alpha_K^{'} = \frac{(M-1)-K}{T/2} \cdot \Delta f \tag{1.13}$$

kde K je počet stavů modulace K=1, 2, ... M, E představuje energii signálu během trvání celého symbolu T, $\omega_c=2\pi f_c$ je úhlová frekvence nosné vlny. Proměnná α_K a $\alpha_K^{'}$ určuje jaký bude mít modulační symbol frekvenční sklon. Δf je rozdíl mezi koncovou a počáteční frekvencí v každé polovině trvání modulačního symbolu.

Na obrázku 6 je zobrazen spectrogram pro čtyřstavovou modulaci Opět se všemi symboly zobrazenými v jedné časové pozici. Na svislé ose je čas. Je zde jasně vidět, jak se v polovině doby trvání modulačního symbolu změní sklon rozmítání. [1], [8]



Obr. 6 Spectrogram 4 - stavové modulace

2 Realizace modulátoru a demodulátoru

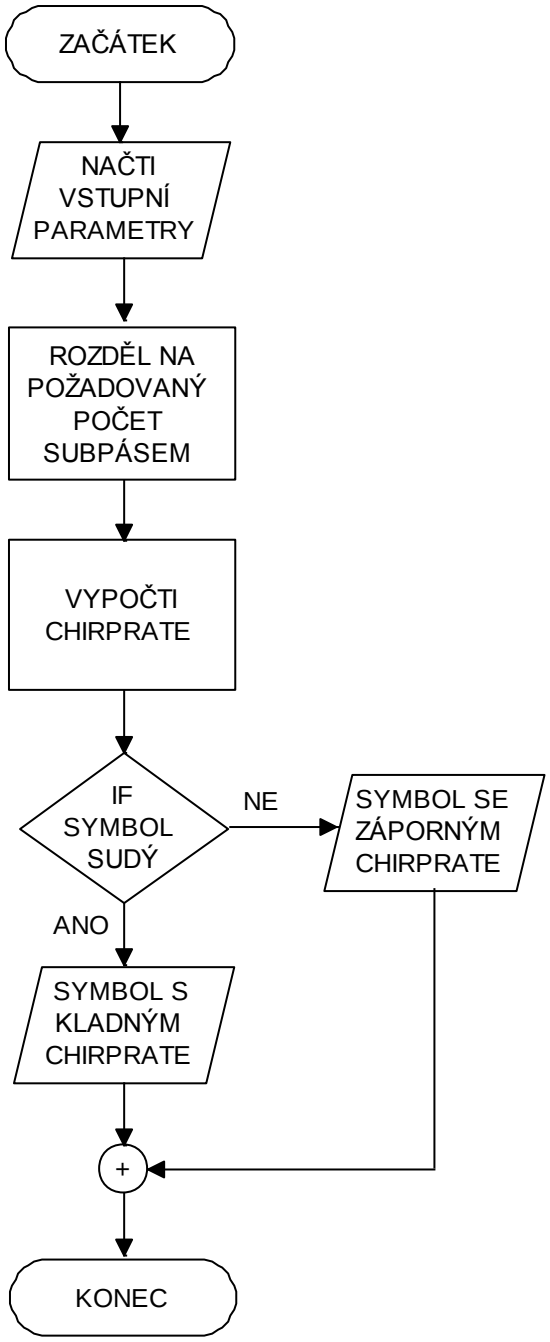
2.1 Softwarové řešení vícestavového modulátoru – Metoda s půlením frekvenčního pásma

Softwarové řešení vícestavové modulace bylo realizováno v prostředí Matlab. Modulace byla vytvořena jako funkce s jednou výstupní proměnou "y", kterou je modulační symbol. Funkce požaduje několik vstupních parametrů. Prvním vstupním parametrem je parametr "t", kterým se zadává doba trvání jednoho modulačního symbolu. Druhým parametrem je "symbol", což je dekadické vyjádření modulačního symbolu. Rozmezí zadávaných hodnot je 0 .. (M-1). Kde "M" je parametr, kterým se zadává počet stavů modulace. Parametr "fup" udává horní mezní frekvenci a parametr "fdown" udává dolní mezní frekvenci. Posledním parametrem je "Fs", který definuje vzorkovací frekvenci.

```
function y=chirpM_B_mod_time(t,symbol,fup,fdown,M,Fs)
```

Na obrázku 7 je zobrazen vývojový diagram pro tuto modulaci. Vývojový diagram popisuje postup při modulaci. Po načtení všech vstupních parametrů se vypočítá rychlost rozmítání frekvence, která je oproti formální rovnici 1.12 a 1.13 upravena. Rychlost rozmítání frekvence se musí měnit nejen se změnou doby trvání modulačního symbolu, ale také podle počtů stavů, který udává parametr "M". Parametr "M" je dělen dvěma, protože pro každé subpásmo náleží dva modulační symboly. Jeden symbol s kladnou rychlostí rozmítání frekvence a druhý symbol se zápornou rychlostí rozmítání frekvence. Níže uveden výpočet rychlosti rozmítání frekvence z kódu pro vytvoření modulace.

```
chirprate=((fup-fdown)/(M/2))*(1/t);
```



Obr. 7 Vývojový diagram pro první metodu modulace

Po vypočtení rychlosti rozmítání frekvence je třeba rozlišit, který modulační symbol se má vytvářet. Jak již bylo řečeno výše, to jaký symbol se má vytvořit je zadáno parametrem "symbol", který je zadán dekadicky. Pro dekadicky zadané hodnoty symbolů 0, 1, 2, 3 odpovídá binární vyjádření 00, 01, 10, 11. Je třeba rozlišit, zda se má vytvářet lichý nebo sudý symbol. Což lze v matlabu pomocí funkce mod(x,2), která vrací zbytek po dělení čísla x číslem 2. Pro sudá čísla tedy vrací hodnotu 0 a pro lichá hodnotu 1. Pokud bude výsledek funkce 0, bude se generovat symbol s kladnou rychlostí rozmítání frekvence. Pokud bude výsledek 1, bude se generovat symbol se zápornou rychlostí rozmítání frekvence. Níže je

ukázka kódu, při porovnání s rovnicí x, je vidět rozdíl v implementaci, který je z důvodu vytvoření jednotlivých subpásem a rozdělení jednotlivých symbolů.

```
switch mod(symbol,2)
    %rozdeleni pro liche 1 a pro sude 0 symboly
    case 0
    y=sin(2*pi.*((fdown+((fup-fdown)/(M/2))*...
      floor(symbol/2)).*time+(chirprate/2).*time.^2));
    %kladny sklon
    case 1
    y=sin(2*pi.*((fup-((fup-fdown)/(M/2))*...
        floor((M-symbol)/2)).*time+(-chirprate/2)...
        .*time.^2)); %zaporny sklon
end;
```

Konkrétní část výpočtu, která je odlišná od formální rovnice 1.10, je zobrazena v této části kódu.

```
(fdown+((fup-fdown)/(M/2))*floor(symbol/2))
```

Příkazem floor(symbol/2) se zajistí rozdělení do příslušných subpásem, názorně to ukazuje tabulka číslo 1. Zbytek kódu, ((fup-fdown)/(M/2)) určuje velikost subpásma, hodnotu která se přičte k počáteční frekvenci fdown. Celý kus kódu dohromady určuje, v kterém subpásmu se bude konkrétní symbol nacházet. Symbol 0(00) má začínat od dolní mezní frekvence. Jak vyplývá z tabulky číslo 1, pro tento symbol je výsledek 0 a proto se k dolní mezní frekvenci nic nepřičte. Další modulační symbol 2(10) má být posunut o jedno subpásmo, z tabulky 1 je pro symbol 2(10) hodnota výsledku 1. Začátek modulačního symbolu bude tedy posunut o jedno subpásmo. Stejně se bude postupovat pro další modulační sudé symboly. Pro liché modulační symboly je v kódu minimální úprava, která zajistí stejný výsledek příkazu floor( ), který zaokrouhluje na nejbližší nižší celý číslo.

```
((fup-fdown)/(dimension/2))*floor((M-symbol)/2)
```

číslo symbolu	floor(symbol/2)	číslo symbolu	floor((M-symbol)/2)
0	0	1	0
2	1	3	1
4	2	5	2
6	3	7	3

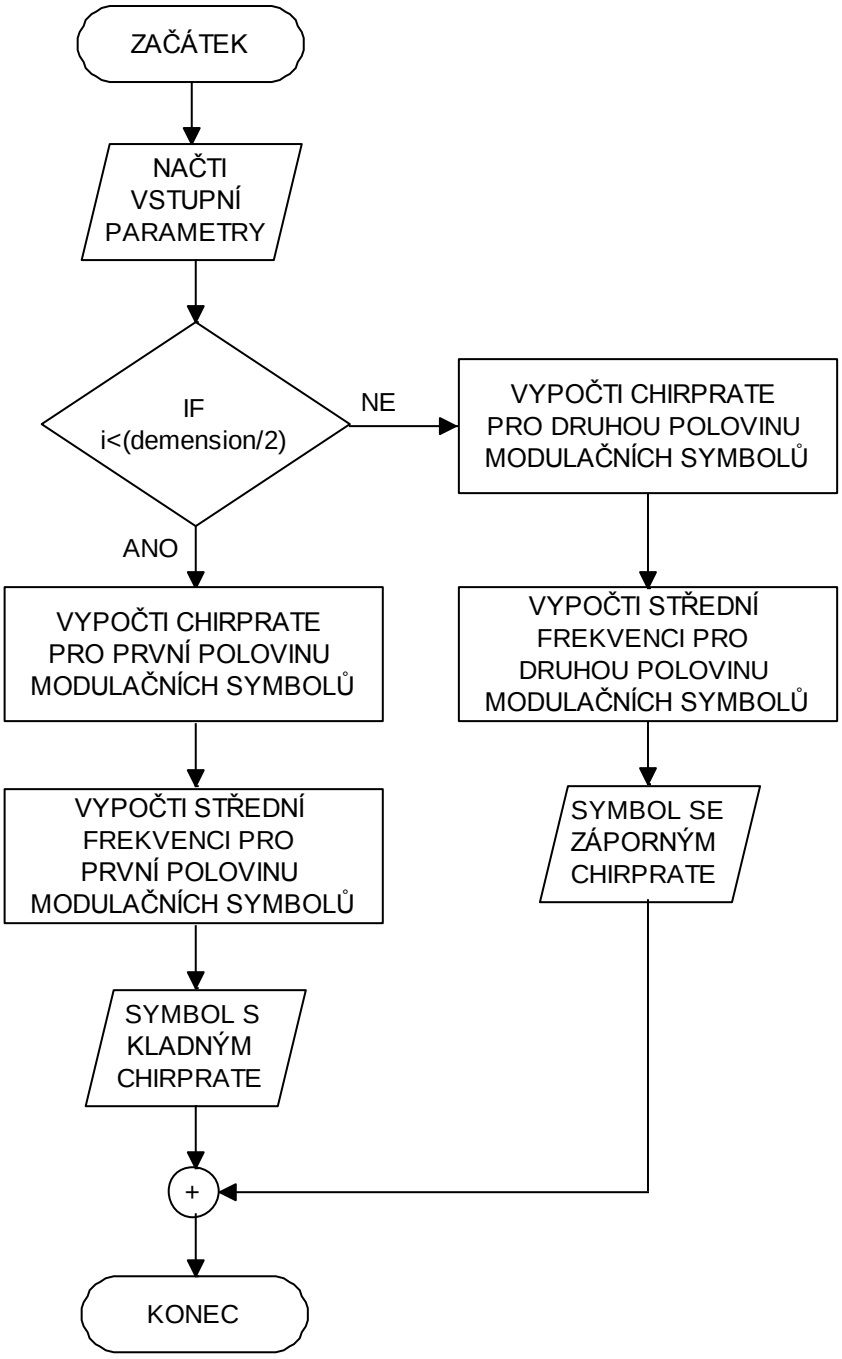
Tab. 1 Rozdělení do subpásem

2.2 Softwarové řešení vícestavového modulátoru – Metoda s půlením časového intervalu

Softwarové řešení vícestavové modulace pro druhou metodu je složitější, protože je třeba vytvořit pro každý symbol dvě rozdílné rychlosti rozmítání frekvence. Modulace byla opět vytvořena jako funkce s jednou výstupní proměnou "y", kterou je modulační symbol. Funkce požaduje několik vstupních parametrů. Prvním vstupním parametrem je parametr "t", kterým se zadává doba trvání jednoho modulačního symbolu. Druhým parametrem je "symbol", což je dekadické vyjádření modulačního symbolu. Rozmezí zadávaných hodnot je 0 .. (M-1). Kde "M" je parametr, kterým se zadává počet stavů modulace. Parametr "fup" udává horní mezní frekvenci a parametr "fdown" udává dolní mezní frekvenci. Posledním parametrem je "Fs", který definuje vzorkovací frekvenci.

```
function y=chirpM_mod_time(t,symbol,fup,fdown,M,Fs)
```

Z popisu vícestavové modulace je patrný rozdíl, oproti první metodě. U výpočtu chirprate (rychlosti rozmítání frekvence) v první metodě nezáleželo na tom, o jaký symbol se jedná. Hodnota rychlosti rozmítání frekvence byla pořád stejná, jediný rozdíl byl ve znaménku podle toho, jestli se jednalo o sudý nebo lichý symbol. U druhé metody tento postup aplikovat nelze, každý symbol má dvě rozdílné hodnoty rychlosti rozmítání frekvence, které se skokově změní v polovině trvání modulačního symbolu. Dalším rozdílem je rozdělení symbolů na dvě poloviny, kde pro dolní polovinu je použito kladné rychlosti rozmítání frekvence a pro druhou záporné. Pro případ čtyřstavové modulace jsou symboly 0(00) a 1(01) generovány s kladnou hodnotou rozmítání rychlosti frekvence a symboly 2(10) a 3(11) generovány se zápornou hodnotou rozmítání rychlosti. Rozdíl je vidět i porovnáním vývojových diagramů pro obě metody. Před samotným generováním je třeba ještě vypočíst střední frekvenci. Je to frekvence, při které dochází ke změně rychlosti rozmítání frekvence a je to frekvence, která je dosažena v polovině trvání modulačního symbolu. Je zřejmé, že podle aktuální rychlosti rozmítání frekvence se bude střední frekvence pro jednotlivé modulační symboly měnit. Název střední frekvence, je tedy z hlediska doby trvání modulačního symbolu, nikoliv šířky pásma modulace.



Obr. 8 Vývojový diagram pro druhou metodu modulace

Kód pro výpočet chirpratu (rychlosti rozmítání frekvence) je zobrazen níže. Je vidět rozdělení pro dolní a horní polovinu modulačních symbolů, kde pro každý symbol jsou počítány dvě rozdílné hodnoty rychlosti rozmítání frekvence. Zároveň je patrné, že hodnota rychlosti rozmítání frekvence pro druhou polovinu modulačních symbolů je pouze zápornou hodnotou rychlosti rozmítání frekvence pro první polovinu modulačních symbolů. Při bližším zkoumání, je patrná podobnost s výpočtem rychlosti rozmítání frekvence pro první metodu.

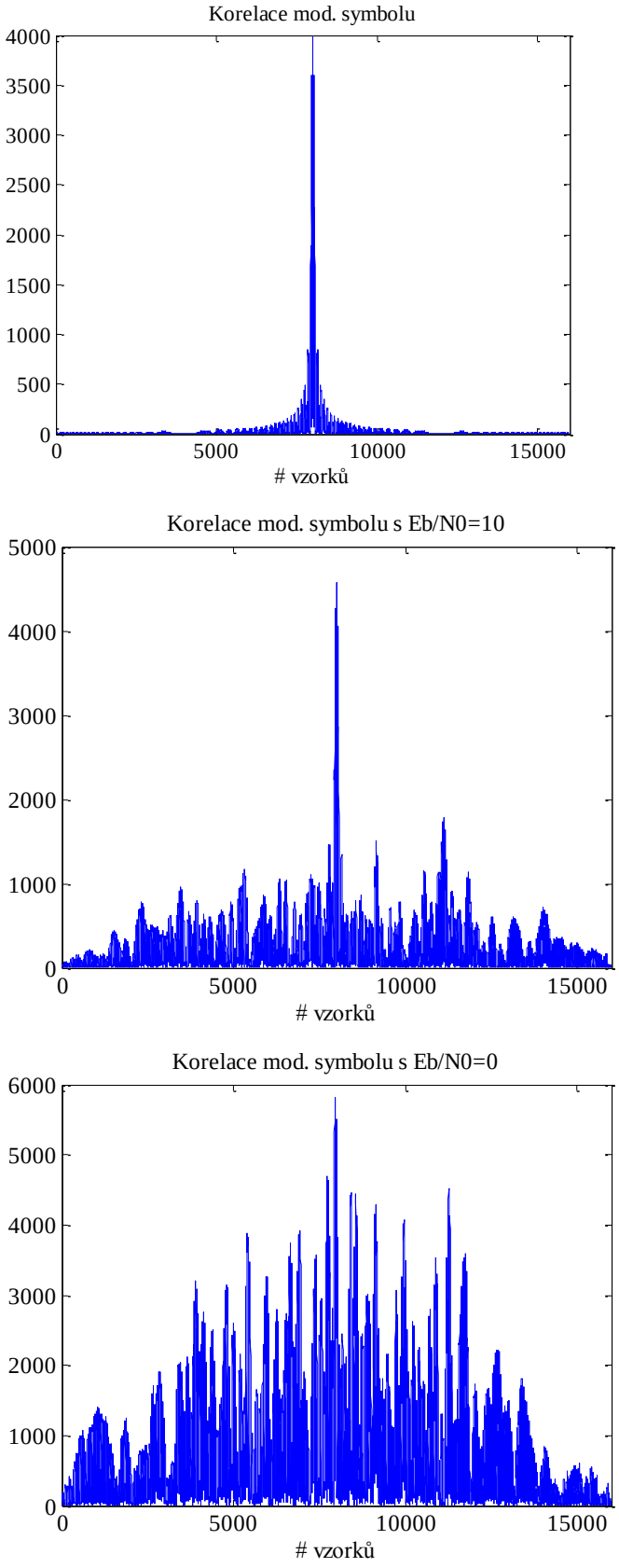
```
for i=1:(dimension/2) %chirprate pro dolní polovinu
    chirprate1(i)=i*((fup-fdown)/((dimension/2)+1))/(0.5*t);
    chirprate2(i)=((dimension/2)-i+1)*((fupfdown)/...
                  ((dimension/2)+1))/(0.5*t);
end;

for i=(dimension/2)+1:dimension % chirprate pro horní polovinu
    chirprate1(i)=-chirprate1(i-(dimension/2));
    chirprate2(i)=-chirprate2(i-(dimension/2));
end;
```

Hodnota rychlosti rozmítání frekvence pro první polovinu trvání modulačního symbolu, je násobkem pořadí symbolu. Pro druhou polovinu trvání modulačního symbolu je hodnota rychlosti rozmítání frekvence v obráceným pořadí.

2.3 Softwarové řešení demodulátoru

Při demodulování signálu se vychází z dobrých korelačních vlastností frekvenčně rozmítaných modulací. Z důvodu snížení výpočetní náročnosti, se symboly pro výpočet korelace s demodulovaným symbolem nevytvářejí při každém demodulování symbolu, ale jsou načteny v matici jako vstupní proměnná. Druhou vstupní proměnnou je vektor dat, který má být demodulován. Výsledkem korelace je matice korelací, z které se vybere maximum pro jednotlivé korelace vektoru dat s jednotlivými modulačními symboly. Pro snížení výpočetního výkonu, se provede hledání maxima pouze ± 20 vzorků, kolem předpokládané pozice maxima. Z tohoto vektoru je opět zjištěno maximum a pozice, kde je maximum nalezeno, odpovídá přijatému modulačnímu symbolu. Tato hodnota je funkcí vrácena zpět.

Výsledek korelace závisí na kvalitě demodulovaného signálu. Pro několik hodnot poměru E_b/N_0 byl zjištěn výsledek korelace. Je možné sledovat, jak pro zhoršující se kvalitu signálu se zhoršuje výsledek korelace. Pro nezašuměný signál je výsledek korelace ostré maximum. Viz obrázek číslo 9a. Pro signál s poměrem E_b/N_0=10 je již patrné rozostření maxima korelace, zobrazeno na obrázku 9b a pro signál s poměrem E_b/N_0=0 lze stále nalézt korelační maximum na správné pozici, ale výsledkem korelace již není ostré maximum.



Obr. 9 Výsledky korelací pro různá E_b/N_0

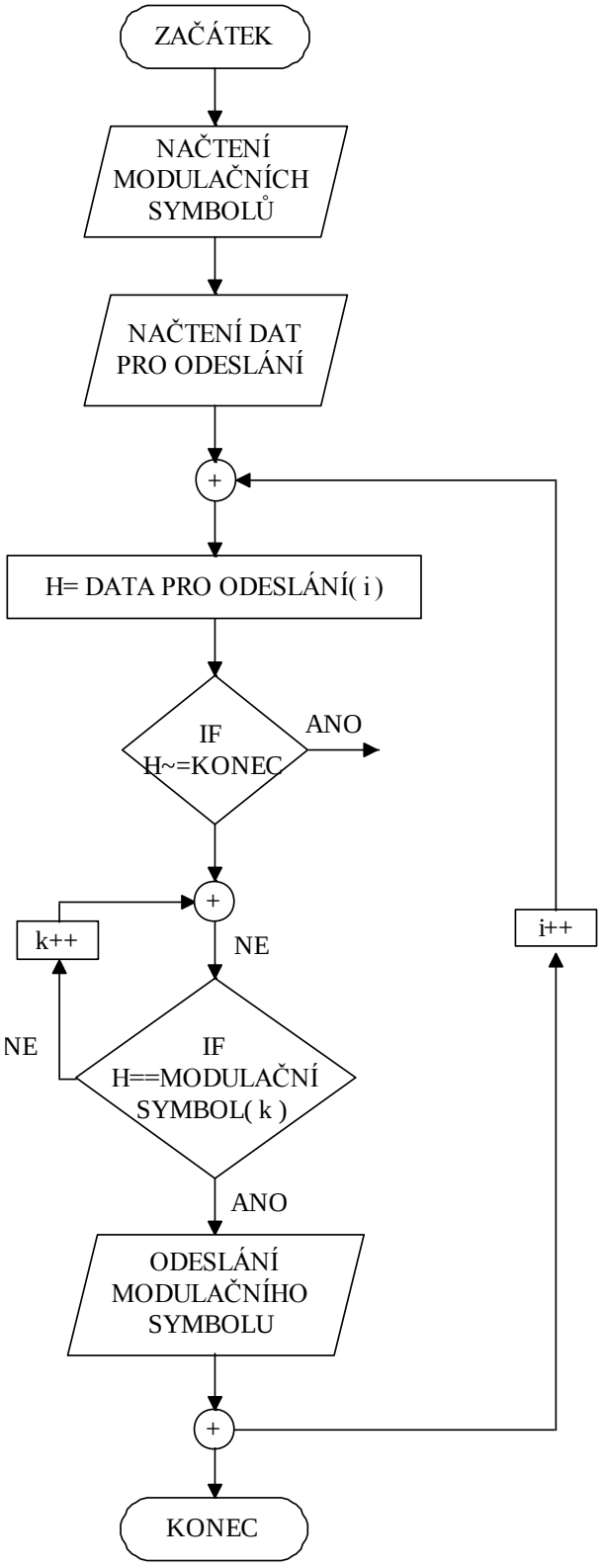
3 Realizace modulátoru a demodulátoru pracujícím v reálném čase

Před samotnou realizací se bylo třeba zamyslet, jak bude komunikace vypadat. Vycházelo se ze známých vlastností frekvenčně rozmítaného signálu. Bylo třeba se rozhodnout, jak se bude signál synchronizovat, kolik k tomu bude použito bitů a dlouhý bude jeden přenosový rámec. Při synchronizaci signálu se využilo dobrých korelačních vlastností frekvenčně rozmítaného signálu, kdy k synchronizaci bude sloužit jeden modulační symbol na začátku každého přenášeného rámce, který bude hledán v posloupnosti dat. Od něj pak bude odpočítáván předem stanovený počet informačních symbolů samotné zprávy. Délka jednoho rámce závisí hlavně na stabilitě oscilátoru u vysílače a je možné ji měnit.

Při realizaci vysílání v reálném čase byl kladen důraz na co nejmenší výpočetní náročnost, aby nedocházelo ke zpoždění vlivem nedostatečného výpočetního výkonu. Z toho důvodu není pro jednotlivý modulační symbol volána funkce, která byla popsána v kapitole 3.1 a 3.2, ale na začátku programu jsou všechny modulační symboly načteny. Volání funkce "chirpM_B_mod_time" nebo "chirpM_mod_time" podle použitého typu modulace by mělo velké nároky na výpočetní výkon. Navíc to není nezbytné, protože všechny symboly, které mohou být použity pro modulaci signálu, jsou známé předem. Proto je možné vytvořit všechny symboly předem a uložit je například do .wav souboru a takto uložené modulační symboly, pak pouze načíst a následně přehrát.

3.1 Realizace vysílání vícestavové rozmítané modulace

Princip programu je ukázán na obrázku 10, kde je vytvořen vývojový diagram pro tento program. Jak bylo řečeno výše, na začátku programu jsou načteny jednotlivé modulační symboly. Po načtení dat, které mají být odeslány, se data uloží do proměnné "H" a jsou postupně odeslány (přehrány). Odesílání dat se provádí, dokud nejsou odeslána všechna data. Pokud nejsou odeslána všechna data, porovnává se hodnota modulačního symbolu v proměnné "H" s jednotlivými modulačními symboly a odpovídající je odeslán. Vše se cyklicky opakuje, dokud není odeslán poslední.



Obr. 10 Vývojový diagram pro odesílání

Pro načtení modulačních symbolů složí objekt dsp.AudioFileReader(nastavení parametrů), který slouží k načtení audio souborů. Nejprve je třeba objekt AudioFileReader definovat a nastavit jeho vlastnosti. Což se provádí přes vstupní parametry tohoto objektu.

Vstupních parametrů je několik. Filename, PlayCount, SampleRate, SamplesPerFrame a OutputDataType. Ne všechny parametry jsou vyžadovány. Pro správné definování, který soubor se načte, je třeba zadat minimálně "Filename" neboli název souboru, který se má načíst. Ukázka načtení audio souboru s názvem "Symbol_0.wav" je vidět na následující řádce.

```
AFR_0=dsp.AudioFileReader('Filename','Symbol_0.wav',...
     'SamplesPerFrame',8000,'OutputDataType','single');
```

Audio soubor obsahuje 8000 vzorků. Pro přehrání celého souboru na jednou se nastaví "SamplePerFrame", což znamená počet vzorků v jednom rámci, na hodnotu 8000. Dále je vidět nastavení "OutputDataTypes", což znamená typ výstupních dat, na "single", defaultní nastavení je "double". [3]

Odesílání dát je realizováno jako přehrání modulačního symbolu. Toto provedení je zvoleno z důvodu, že amatérské radiostanice obsahují zvukový vstup a výstup. Pro přehrání zvukového záznamu slouží objekt dsp.AudioPlayer(nastavení parametrů). Nejprve je třeba definovat AudioPlayer objekt a nastavit parametry pro přehrávání, které se zadávají jako vstupní parametry tohoto objektu. Mezi hlavní parametry patří DeviceName, SapleRate, BufferSizeSource, BufferSize, QueueDuration. Další parametry je možné nalézt v nápovědě programu Matlab po zadání příkazu "help dsp.AudioPlayer". Hlavní parametry a jejich nastavení bude následně podrobněji rozebráno. Prvním parametrem "DeviceName" se specifikuje zařízení, které bude sloužit k přehrání zvukových záznamů. Výchozím nastavením je standardní výstupní zařízení počítače. Parametr "SampleRate" nastavuje počet vzorků, které se přehrají za vteřinu. Ve výchozím nastavení je nastaveno 44100 Hz. Parametr "BufferSizeSource" určuje, jak je určena velikost vyrovnávací paměti, defaultně je nastaven na "Auto", kdy se velikost vyrovnávací paměti určuje na základě vzorkovací frekvence. Vlastní velikost vyrovnávací paměti, lze zadat při nastavení "BufferSizeSource" na "Property", kde se velikost vyrovnávací paměti zadá pomocí parametru "BufferSize". Parametr "QueueDuration" nastavuje velikost fronty, kterým lze řídit kompromis mezi zpožděním a výpadkem dat. Po definování a nastavení vlastností objektu, je třeba v programu volat příkaz "step( )", kterým se přehraje zvukový soubor podle vlastností dsp.AudioPlayeru. Práce s objektem dsp.AudioPlayer je nastíněna níže.

```
AFR_0=dsp.AudioFileReader('Filename','Symbol_0.wav',...
     'SamplesPerFrame',8000,'OutputDataType','single');

AP = dsp.AudioPlayer('SampleRate', AFR_0.SampleRate,...
     'QueueDuration',0);

while ~isDone(AFR_0)

     audioIn=step(AFR_0);
     step(AP,audioIn);

end
```

Na první řádce je zmíněná definice objektu AudioFileReaderu, kterou vznikne proměnná AFR_0 s definovanými vlastnostmi. Na druhé řádce je definice objektu AudioPlayeru, kterou vznikne proměnná AP s definovanými vlastnostmi. V samotném těle cyklu jsou dva příkazy, první načte do proměnné audioIn zvukový soubor. Tento příkaz je nezbytný, protože na první řádce se pouze definovaly vlastnosti. Samotný zvukový soubor se načetl až do proměnné audioIn příkazem step( ). Druhý příkaz přehraje obsah proměnný audioIn pomocí AudioPlayeru. Z úseku programu je vidět několik souvislostí, tou hlavní je, že příkaz step( ) má specifický chování pro každý objekt z DSP toolboxu. Vlastnosti jednoho objektu lze přenést i na další objekt, jak lze vidět v nastavení vzorkovací frekvence u dsp.AudioPlayeru. [4]

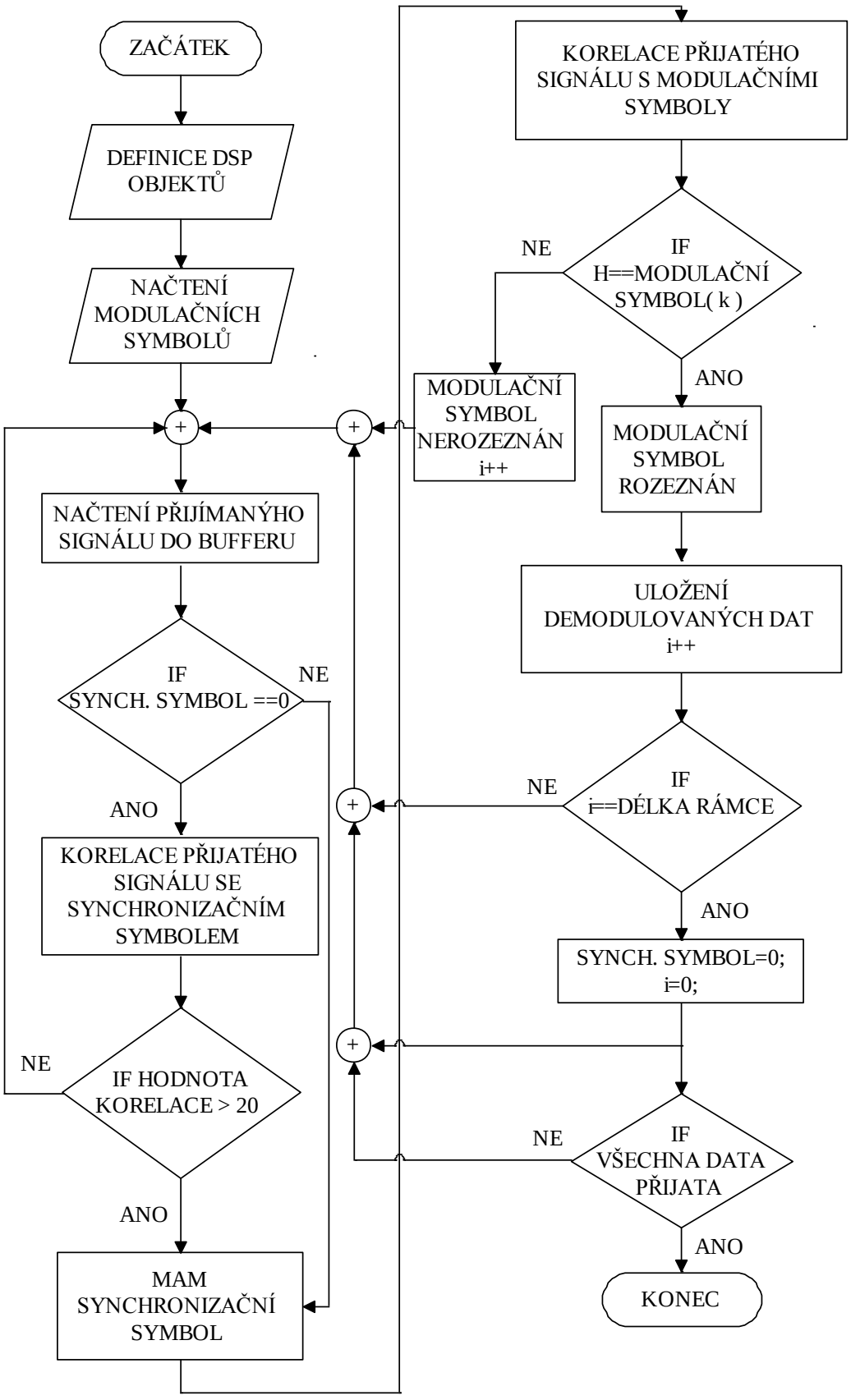
Pro záznam modulačních funkcí byla vytvořena funkce zaznam_mod_symbolu(t, fup, fdown, M, Fs). Kde "t" je doba trvání modulačního symbolu, "fup" je horní mezní frekvence, "fdown" je dolní mezní frekvence, "M" definuje počet stavů modulace a "Fs" je vzorkovací frekvence.

3.2 Realizace přijmu vícestavové rozmítané modulace

U softwarové realizace přijímače je opět kladen důraz na minimalizaci výpočetní náročnost programu. U přijímače je nejnáročnější na výpočetní výkon provádění korelací, pro zjištění podobnosti přijatého signálu s modulačními signály. Z tohoto důvodu je snaha omezit velikost dat, neboli počet vzorků, které se budou korelovat. Ze zadání je určeno vytvořit základní modulátor a demodulátor v akustickém pásmu. Není tedy přesně specifikovaná dolní a horní mezní frekvence, musí však ležet v akustickém pásmu. Zvolená dolní mezní frekvence je 300 Hz a horní mezní frekvence je 3000 Hz. Takto zvolená šířka pásma tedy není nezbytná a je možné ji změnit podle svých požadavků. Čím větší ovšem šířka pásma bude, tím větší bude muset být vzorkovací frekvence a tím větší počet vzorků se bude počítat při korelaci. Pro takto zvolenou šířku pásma byla zvolena vzorkovací frekvence 8000 Hz. Pro modulační

symbol, který bude trvat jednu vteřinu, tedy bude náležet 8000 vzorků. Princip softwarové realizace přijímače je znázorněn na obrázku 11.

U přijímače se na začátku definují veškeré použité DSP objekty, následuje načtení všech modulačních symbolů, s kterými bude počítána korelace přijímaného signálu. Přijímaný signál je načten do bufferu dat. Následuje podmínka, která testuje, zda je již synchronizační symbol nalezen nebo nikoli. Po startu programu je tato pomocná proměnná nastavena na nulu, to znamená, synchronizační symbol nenalezen. Z bufferu jsou data vyčítány a porovnávány se synchronizačním symbolem. Pokud hodnota korelace přesáhne hodnotu dvacet, považuje se synchronizační symbol za nalezený. Zde se program větví na dvě části. Pokud synchronizační symbol v přijatém signálu nebyl, tak se hledá cyklicky dál, až do nalezení synchronizačního symbolu. Pokud je synchronizační symbol nalezen, pokračuje se v programu dál, kde se v dalším kroku provádí korelace přijatého signálu s modulačními symboly. Zde se opět program větví na dvě části, podle toho zda byl modulační symbol nalezen nebo nikoli. Pokud nebyl nalezen, vrací se program zpět k načtení dalšího signálu do bufferu. Zároveň je zvětšena o jedničku pomocná proměnná, která počítá, kolik se demodulovalo symbolů. Pokud se modulační symbol nalezl, zjistí se, o jaký symbol se jedná a tento symbol je uložen do demodulovaných dat. Zároveň se opět zvětší o jedničku pomocná proměnná, která počítá, kolik se demodulovalo symbolů. Následuje podmínka, která testuje, zda již byl přijat celý rámec dat. Pokud ne, program se zpátky vrací k načtení dalším dat z bufferu a počítá se další korelace přijatého signálu s modulačními symboly. Pokud byl již přijat celý rámec, vynuluje se proměnná indikující nalezení synchronizačního symbolu a hledá se znovu synchronizační symbol. Zároveň se vynuluje proměnná, která počítá počet demodulovaných symbolů. Následuje podmínka, která testuje, zda byly již přijaty všechny data. Pokud ne, pokračuje se v příjmu. Pokud ano je program ukončen.



Obr. 11 Vývojový diagram pro přijem signálu

Na začátku programu se definují objekty dsp.AudioRecorder, dsp.Buffer a dsp.Crosscorrelator. Objekt dsp.AudioRecorder je použit pro příjem akustického signálu. Mezi hlavní parametry, které se dají nastavit, patří DeviceName, SapleRate, NumChannels, DeviceDataType, BufferSizeSource, BufferSize, QueueDuration, SamplesPerFrame, OutputDataType. Některé z těchto parametrů již byly popsány a jejich význam se u tohoto objektu nezměnil. Parametr "NumChannels" nastavuje počet audio kanálů, ve výchozím nastavení je nastaven na dva kanály. Parametrem "OutputDataType" se nastavuje datový typ, ve výchozím nastavení je nastaven na "double". U objektu dsp.Buffer lze nastavit Length, OveralapLength, InitialConditions, FrameBasedProcessing. Parametrem "Length" lze nastavit délka bufferu, neboli počet vzorků které se budou do bufferu načítat. Parametrem "OverlapLength" se nastavuje počet vzorků, které se budou cyklicky přepisovat po zaplnění bufferu. Parametr "InitialConditions" nastavuje počáteční výstup bufferu pro případy nenulové latence. Posledním parametrem je "FrameBasedProcessing", je-li tento parametr nastaven na hodnotu "true", M vstupních dat z jediného vstupu je zpracováváno po rámcích. Je-li hodnota tohoto parametru "false", každý vstupní prvek je považován za samostatný kanál, to znamená, že M vstupních dat je zpracováváno po jednotlivých prvcích do M kanálů. Výchozí hodnota je "true." Pro počítání korelačních funkcí, je použit objekt dsp.Crosscorrelator, kde lze nastavit, v jaké oblasti se bude výpočet provádět. "Time Domain" znamená počítání v časové oblasti, "Frequency Domain" počítání ve frekvenční oblasti a poslední možností je "Fastest", při které se pracuje v oblasti, která minimalizuje počet výpočtů. [5], [6], [7]

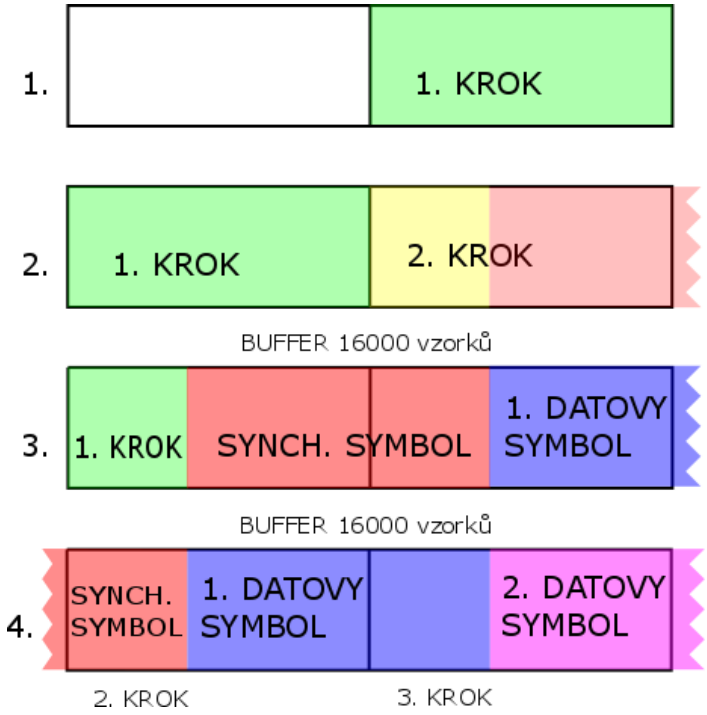
Po definování všech objektů, se načtou audio soubory všech modulačních symbolů. Ty jsou načteny pro potřebu vypočtení korelace se vstupním signálem. Vstupní signál je načten pomocí objektu dsp.AudioRecorder do bufferu. Nastavení objektu dsp.AudioRecorder je zobrazeno zde.

```
M=dsp.AudioRecorder('SamplesPerFrame',8000,'BufferSizeSource',
'Property','BufferSize',8000,'QueueDuration',0,'NumChannels',1
,'OutputDataType','double');
Mic.DeviceName='Default';
Mic.SampleRate=Fs;
```

Z nastavení "SamplesPerFrame" na 8000 vzorků, je nastaveno z důvodu, že jeden modulační symbol je vytvořen při vzorkovací frekvenci 8000 Hz a jeho doba trvání je 1 sekunda, obsahuje tedy 8000 vzorků. Nastavení objektu dsp.bufferu je zobrazeno zde.

```
InBuff = dsp.Buffer(16000,8000);
```

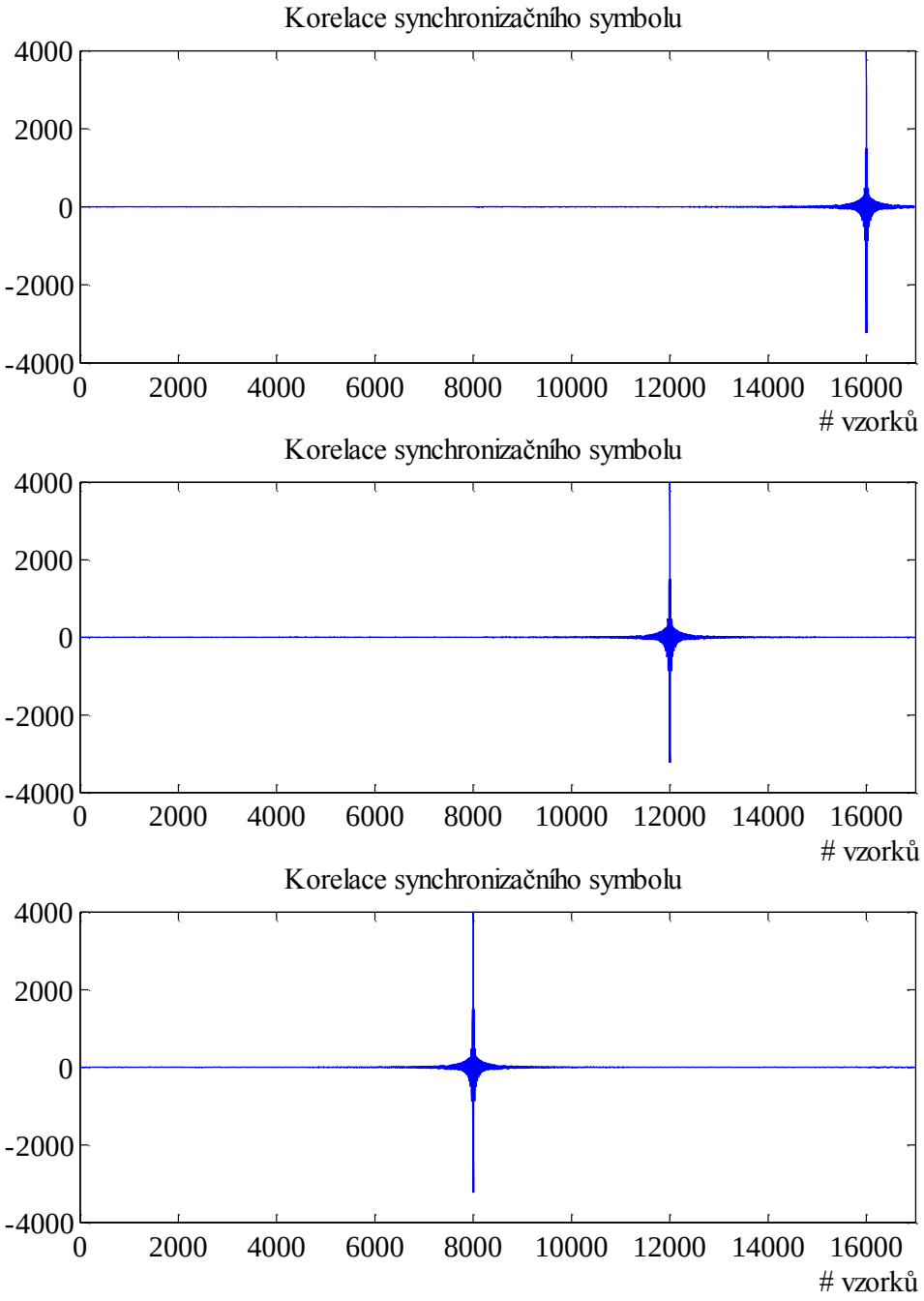
Velikost bufferu je 16000 vzorků. Každým krokem programu se načte do bufferu 8000 vzorků. Jak vyplývá z nastavení AudioRecorderu a bufferu, zapisuje se do buferu vždy jeden celý rámec. Velikost bufferu na 16000 vzorků je zvolena záměrně, aby bufferu obsahoval dva celé rámce. Důvod takto zvolené velikosti objasňuje obrázek 12.



Obr. 12 Načítání dat do bufferu

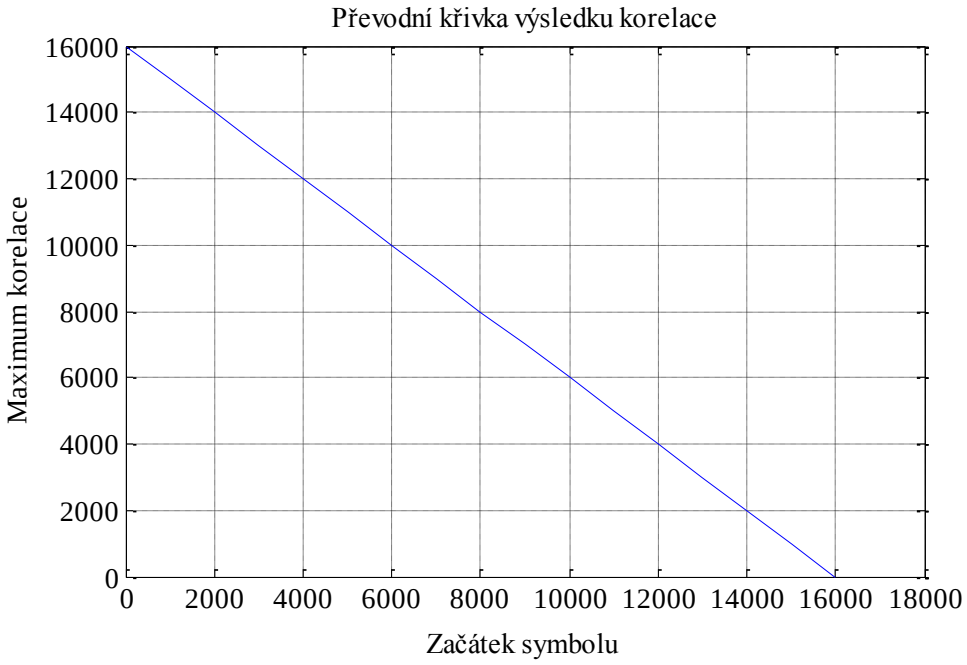
Do bufferu jsou cyklicky každou vteřinu načtena přijímaná data v podobě 8000 vzorků. Buffer je plněn od zadu, při dalším načtení dat se všech 8000 vzorků, tedy celý jeden rámec, přesune o jednu pozici doleva, jak je znázorněno posunem zeleného rámce na obrázku číslo 10. Tento rámec představuje neznámý přijatý signál. V dalším cyklu se načte další rámec (žlutý), který bude obsahovat synchronizační symbol (červený). Obecně se však nedá předpokládat, že přijatý synchronizační symbol se bude nacházet přesně v přijatém rámci. Jak je znázorněno na obrázku číslo 10, začátek synchronizačního symbolu se může nacházet kdekoliv v přijatém rámci a nemusí být přijat celý. V dalším kroku se však předchozí rámec posune doleva a načte se další přijatý rámec, kde už bude synchronizační symbol celý. To je důvod, proč je zvolená velikost bufferu 16000 vzorků neboli dva rámce. Ať je spuštěn příjem kdykoliv, nezávisle na spuštění odesílání, vždy bude v bufferu celý synchronizační symbol.

V každém cyklu programu se provádí výpočet korelace mezi obsahem bufferu a synchronizačním symbolem. Ve chvíli, kdy je synchronizační symbol obsažen v bufferu objeví se ostré korelační maximum. Pozice korelačního maxima závisí na poloze synchronizačního symbolu v bufferu. Na obrázku číslo 13 je ukázka výsledků korelací pro tři pozice synchronizačního symbolu. V prvním případě, se synchronizační symbol nacházel na pozici 1 až 8000. V druhém případě, se synchronizační symbol nacházel na pozici 4001 až 12000 a v třetím případě, se synchronizační symbol nacházel na pozici 8001 až 16000.



Obr. 13 Výsledek korelace pro různou pozici synchronizačního symbolu

Jinak řečeno, symbol se postupně nacházel na začátku, u prostřed a na konci bufferu. Pozice maxima korelace neodpovídá začátku ani konci synchronizačního symbolu. Souvislost mezi pozicí synchronizačního symbolu a výsledkem korelace je uvedena na obrázku 14.



Obr. 14 Převodní křivka výsledku korelace na začátek synchronizačního symbolu

Rovnici přímky lze vyjádřit pomocí rovnice 3.1. Z rovnice 3.1 se po vyjádření proměnné x, která odpovídá začátku synchronizačního symbolu, zjistí pozice začátku synchronizačního symbolu.

$$y = -x + 16001 \tag{3.1}$$

Z obrázku číslo 10 je vidět, že pozice začátku synchronizačního symbolu je v dalším kroku začátkem modulačního symbolu, který nese užitečnou informaci. Z obrázků číslo 11 a 12 lze zjistit ještě jednu důležitou vlastnost. Pokud se v bufferu nachází pouze část synchronizačního symbolu a maximum korelace bude dostatečně vysoké, bude program považovat, že nalezení synchronizačního symbolu bylo úspěšné. V takovém případě, by se všem v dalším cyklu prováděla korelace pouze s částí přijatého modulačního symbolu, což by mělo negativní vliv a chybovost přenosu. Existují dvě možnosti, jak tento problém vyřešit. První možností je uložit výsledek korelace a porovnat ho s výsledkem korelace z následujícího cyklu. V okamžiku kdy bude synchronizační symbol v bufferu celý, bude maximum korelace větší. Druhým

způsobem je sledovat pozici maxima korelace a z toho dopočítat začátek synchronizačního symbolu. Pokud se bude začátek synchronizačního symbolu nacházet až v druhé polovině bufferu, znamená to, že není v bufferu celý. V takovém případě program počká na další cyklus, kdy se načte i zbytek synchronizačního symbolu do bufferu. Tento způsob je méně náročný z hlediska výpočetní náročnosti, a proto je zde použit.

Po nalezení synchronizačního symbolu následují modulační symboly, které nesou užitečnou informaci. Oproti hledání synchronizačního symbolu, je nyní nutné provádět korelaci přijatého signálu se všemi modulačními symboly. To je velmi náročné na výpočetní výkon a bylo nezbytné najít řešení, jak snížit tuto náročnost na minimum. První snížení výpočetní náročnosti vyplývá z obrázku 11. Vzhledem k tomu, že počet vzorků jednoho modulačního symbolu je známý, v tomto případě 8000 a pozice modulačního symbolu v bufferu je po nalezení synchronizačního symbolu také známá. Je zbytečné počítat korelaci z celého bufferu. Výpočet korelace je tedy zúžen pouze na pozici, kde se modulační symbol nachází. Dalším ušetřením výpočetního výkonu se dosáhne v případě, že se nebude hledat maximum korelace z celého rozsahu výsledku korelace, ale pouze v předpokládaném úseku, kde se bude nacházet. Oproti hledání maxima při hledání synchronizačního symbolu, kde se prováděla korelace z 16000 vzorků uložených v bufferu, bude maximum korelace na jiné pozici. To je dáno snížením počtu vzorků, pro které se bude korelace počítat.

```
dataIn = step(InBuff, step(Mic));

PomXcorr= abs(step(xcorr,symbolyMod(:,(1:15)),...
          dataIn(aktualSymbol:(aktualSymbol+7999))));

[PomMax,PozMax]=max(PomXcorr(7950:8050,:));
```

Tento úsek kódu na první řádce načítá do bufferu přijímané data pomocí AudioRecorderu. Na další řádce je vidět provádění korelace se všemi modulačními symboly kromě synchronizačního symbolu. Zároveň je patrné omezení výpočtu korelace na přijatá data v bufferu pouze na pozici, kde se nachází modulační symbol. Na dalším řádku je zjišťování maximální hodnoty korelace. Hledání maxima se provádí 50 vzorků kolem předpokládané pozice maxima. Tento interval je zvolen, protože oscilátor vysílače nemusí být naprosto stabilní.

Maximální hodnoty korelace jsou ukládány do vektoru, kde se provede další hledání maximální hodnoty, tentokrát z maximálních hodnot pro jednotlivé modulační symboly. Pozice, kde je nalezeno maximum odpovídá přijatému modulačnímu symbolu. Výrazného

snížení výpočetní náročnosti by se dosáhlo, pokud by dsp funkce korelátoru umožňovala omezit rozsah možných posunů, pro které je korelace počítána. Tato vlastnost však není podporována

4 Testování chybovosti modulace

4.1 Softwarové řešení testování bitové chybovosti

Pro testování chybovosti přenosu, byl vytvořen program, který na základě zvoleného počtu stavů, doby trvání modulačního symbolu, dolní mezní frekvence a horní mezní frekvence přidá k vytvořenému modulačnímu symbolu AWGN šum, čímž je modelován průchod signálu AWGN kanálem. Takto zašumělý modulační symbol je demodulován. Pro demodulování signálu, byla vytvořena v Matlabu funkce dem_chirp, která očekává dvě vstupní proměnné. První proměnnou je vektorem se zašuměným modulačním symbolem a druhou proměnnou je matice všech modulačních symbolů.

```
out=dem_chirp(awgn(signal(:,j),snr,0),symbolyMod);
```

Funkce dem_chirp vypočte korelaci zašuměného signálu se všemi modulačními symboly a do matice uloží maximální dosaženou hodnotu korelace na očekávané pozici. Z této matice se zjistí maximální hodnota a podle pozice maxima je určeno o jaký modulační symbol se jedná. Tuto pozici funkce vrací zpět do programu pro testování chybovosti. Zde se porovnává, zda se odesílaný modulační symbol rovná demodulovanému modulačnímu symbolu. Pokud se symboly rovnají, odesílá se další zašuměný modulační symbol. Pokud se modulační symboly nerovnají, převede se dekadické číslo modulačního symbolu na binární a zjistí se, v kolika bitech se symboly liší.

```
if (out~=j)
   y=sum(ne(dec2bin(j,log2(chirp_dim)),...
     dec2bin(out,log2(chirp_dim))));

   BER(q)=BER(q)+y;
   c=BER(q)
   ber=(log2(chirp_dim)*c/count);
   SER(q)=SER(q)+1;
   d=SER(q);
   pocet=count

end;
```

Tento počet se uloží do proměnné *y*, v které je uložen počet chyb pro aktuální demodulovaný symbol. Tato hodnota se přičte k celkové chybovosti, kterou představuje proměnná BER(Bit

Error Rate) pro danou hodnotu E_b/N_0. Proměnná SER (Symbol Error Rate) představuje symbolovou chybovost a proto je vždy k aktuální hodnotě přičtena jednička v případě, že se symboly liší. Bitová a symbolová chybovost byla testována pro hodnoty E_b/N_0 od -4 do 10 s krokem 0,5. Podle aktuální hodnoty E_b/N_0 byla vypočítána velikost poměru signál-šum, což je vstupní parametr pro přidání AWGN šumu k modulačnímu symbolu.

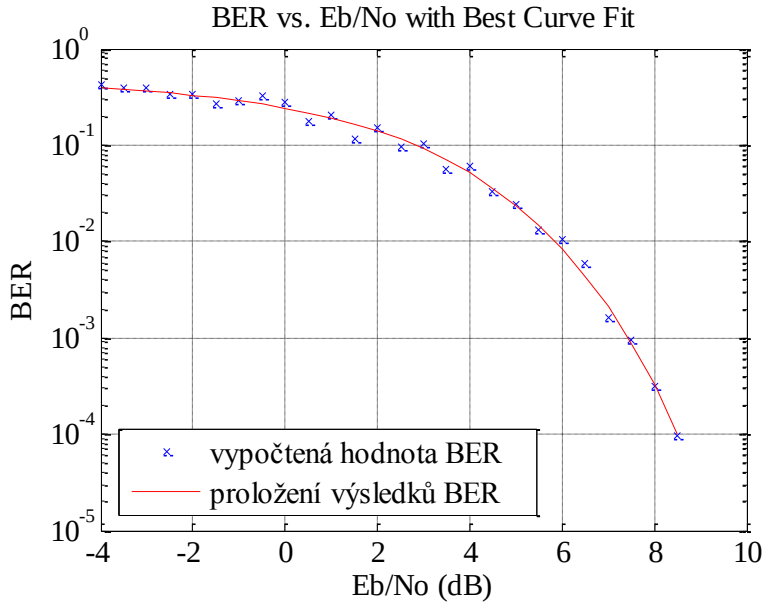
Velikost bitové chybovosti je pro konkrétní hodnotu E_b/N_0 zastavena v případě, že velikost chybovosti je nižší než 10^{-5}. Vzhledem k vysoké časové náročnosti je výpočet chybovosti pro konkrétní hodnotu E_b/N_0 zastaven, pokud počet chyb přesáhl hodnotu 15 nebo počet odeslaných bitů přesáhl hodnotu 2000000. Obě zastavovací podmínky vedou k nepřesnému stanovení chybovosti, byly však využity pro dosažení výsledků chybovosti v kratším čase.

4.2 Dosažené výsledky testování chybovosti v AWGN kanálu

Bitová chybovost byla testována pro 8, 16, 32 a 64 stavů. Pro testování chybovosti byla zvolena první metoda modulace, kde se pro dosažení většího počtu stavů dosahuje půlením frekvenčního pásma.

8 - stavová modulace

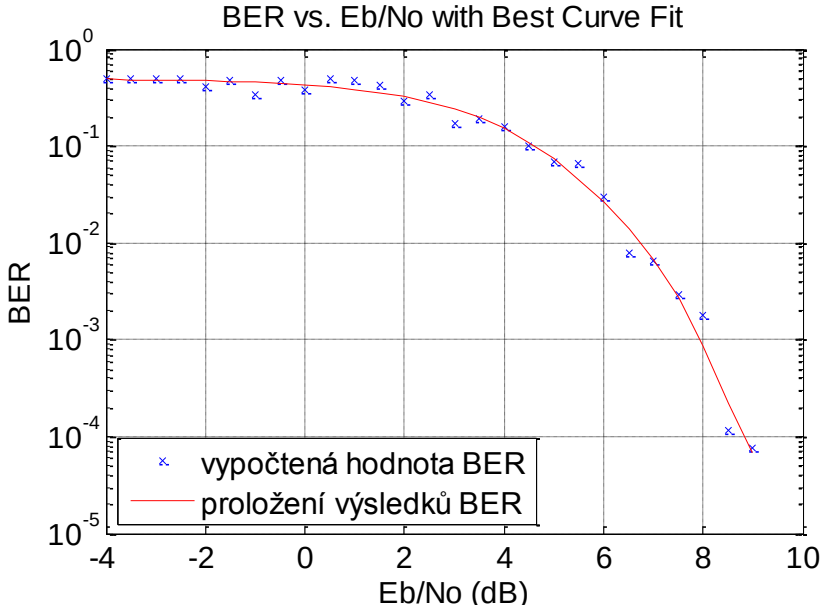
Pro 8 - stavovou modulaci se dosáhlo hodnoty BER menší než 10^{-4} pro hodnotu E_b/N_0=8.5 dB. Od hodnoty E_b/N_0=7,5 je chybovost nižší než 10^{-3}. Pro lepší představu a zobrazení chybovosti byla použita funkce berfit.



Obr. 15 Simulace BER pro 8 - stavovou rozmítanou modulaci

16 - stavová modulace

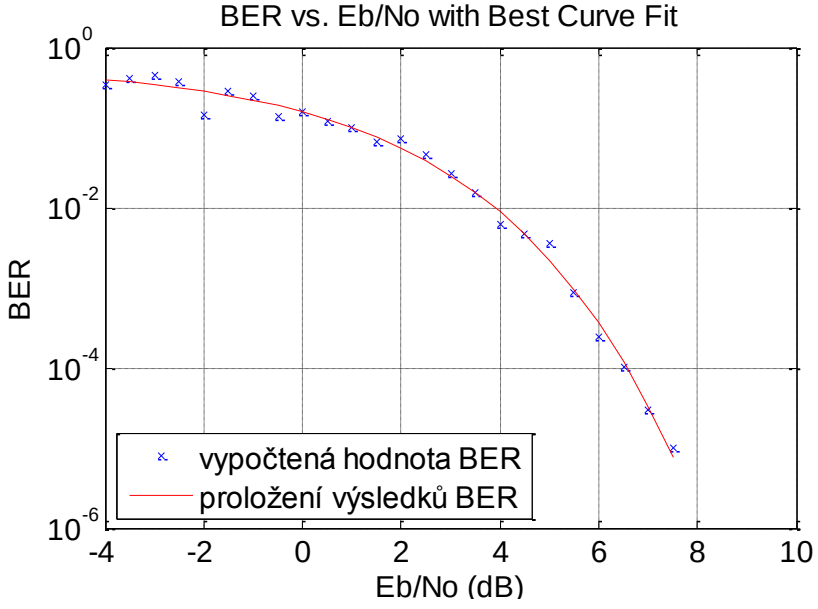
Pro 16 - stavovou modulaci je dosažena chybovost menší než 10^{-3} od hodnoty E_b/N_0 mezi 7,5 až 8,5 dB. Nejnižší chybovosti $7{,}75 \cdot 10^{-4}$ je dosaženo pro hodnotu E_b/N_0 kolem 9 dB. Zde se projevuje chyba, která je zde způsobena z důvodu zastavení počítání chybovosti při dosažení 15 chyb pro konkrétní hodnotu E_b/N_0.



Obr. 16 Simulace BER pro 16 - stavovou rozmítanou modulaci

32 - stavová modulace

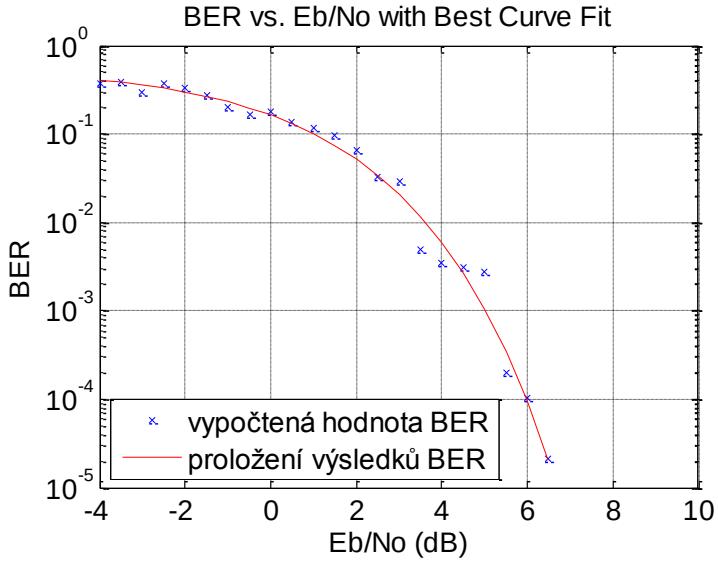
Pro 32 - stavovou modulaci je dosažena chybovost menší než 10^{-3} od hodnoty E_b/N_0 mezi 5 až 6 dB. Chybovost menší než 10^{-5} je dosaženo pro hodnotu E_b/N_0=7,5 dB.



Obr. 17 Simulace BER pro 32 - stavovou rozmítanou modulaci

64 - stavová modulace

Pro 64 - stavovou modulaci je podle předpokladů dosažena největší energetická účinnost. Chybovost menší než 10^{-3} je dosažena od hodnoty E_b/N_0 mezi 4,5 až 5,5 dB. Nejnižší chybovosti $2{,}11 \cdot 10^{-5}$ je dosaženo pro hodnotu E_b/N_0=6,5 dB.



Obr. 18 Simulace BER pro 64 - stavovou rozmítanou modulaci

Při zvýšení počtu zaznamenávaných chyb je pravděpodobné, že výsledná bitová chybovost, by byla ještě o něco nižší. Z grafů je vidět velký rozptyl vypočítaných hodnot bitové chybovosti okolo průměrné hodnoty, což značí, že dosažené výsledky jsou značně nepřesné. Pro dosažení výsledků, které by bylo možné použít, pro stanovení E_b/N_0 pro požadovanou bitovou chybovost, by bylo nezbytné zvednout počet chyb, pro které je chybovost počítána. Zvolená hodnota 15, která byla zvolena z důvodu ušetření času pro testování chybovosti, je nedostatečná. Tento fakt, je vidět ve výsledku bitové chybovosti pro 16 stavů, kde jsou dosažené výsledky horší než pro 8 stavovou modulaci. Tento výsledek odporuje předpokladu, že při zvyšování stavů, se zvyšuje energetická účinnost. Neznamená to ovšem chybný předpoklad, pouze to potvrzuje, že je nutné bitovou chybovost testovat pro větší počet zaznamenaných chyb.

5 Závěr

V této diplomové práci byly řešeny způsoby vytvoření vícestavových rozmítaných modulací. Byla zde rozebrána teorie vytváření jednotlivých frekvenčně rozmítaných symbolů. Při této příležitosti byly v programu Matlab vytvořeny dva typy modulací. Prvním způsobem je metoda půlení frekvenční pásma a druhým způsobem je metoda s půlením časového intervalu. Pro obě metody byl v prostředí Matlab vytvořen modulátor pro možnost testování vlastností vícestavových rozmítaných modulací. Modulátor byl vytvořen jako funkce, kde vstupními proměnnými jsou doba trvání modulačního symbolu, dolní mezní frekvence, horní mezní frekvence, počet stavů modulace a vzorkovací frekvence.

Pro možnost testování chybovosti v reálném čase, byl vytvořen modulátor a demodulátor pracující v reálném čase. K tomu byl v prostředí Matlab využit DSP toolbox. Z důvodu omezení výpočetní náročnosti, se zde jednotlivé modulační symboly nevytvářejí pro každý symbol zvlášť. To je možné, díky předem známému počtu stavů, které budou při komunikaci použity. Jsou tedy vytvořeny předem pomocí funkce zaznam_mod_symbolu. Takto vytvořené symboly jsou na začátku načteny v podobě audio souborů a pro příslušné modulační symboly přehrány. Realizace demodulátoru je opět velmi náročná na výpočetní výkon. Z toho důvodu jsou všechny symboly na začátku načteny v podobě audio souborů a následně porovnávány s přijatým signálem. Pro demodulaci symbolů je využito výborných korelačních vlastností frekvenčně rozmítaného signálu. Jistým nedostatkem, který zvyšuje výpočetní náročnost, je absence možnosti omezení rozsahu posunů, pro které je korelace počítána. Tento nedostatek se výrazně projevuje u vyššího počtu stavů modulace.

Při testování chybovosti pro vyšší počet stavů modulace v reálném čase se ukázal nedostatečný výpočetní výkon počítačů ve školní laboratoři. Časová náročnost při testování bitové chybovosti v reálném čase u takto pomalé modulace, kde jeden modulační symbol trvá jednu vteřinu, je velmi výrazná. Při bitové chybovosti demodulace 10^{-5} je třeba odeslat 200 000 bitů, pro zachycení jedné chyby. Tomu při testování v reálném čase odpovídá 200 000 vteřin pro jednu chybu a jednu hodnotu E_b/N_0. Pokud se ukončí testování, při dosažení 20 chyb, což je stále velmi hrubé, je časová náročnost delší než 46 dní pro jednu hodnotu E_b/N_0. Testování chybovosti je však funkční, a pokud by bylo potřeba zjistit bitovou chybovost při komunikaci v reálném čase, je možné ho využít. Při testování se došlo k závěru, že zvolený způsob stanovení synchronizačního bitu je pro nízké hodnoty E_b/N_0, kde jsou symboly silně zašuměny nevyhovující. Pokud by se měla uskutečnit komunikace i při vyšší chybovosti než 10^{-4}, bylo by nutné zvolit jiný komplexnější způsob. Z důvodu velké časové náročnosti a nedostatečného výpočetního výkonu byl vytvořen nový program pro testování, který

netestoval bitovou chybovost v reálném čase. Přesto časová náročnost při testování chybovosti, byla větší než 48 hodin, záleželo na počtu stavů modulace. Z toho důvodu bylo testování prováděno pro menší počet chyb, které vedlo k určité nepřesnosti v dosažených výsledcích. Pro korektnější dosažení výsledků, by bylo za potřebí, zvýšit počet chyb, pro který je počítána chybovost na hodnotu větší než 50.

Z výsledků bitové chybovosti pro 8, 16, 32 a 64 stavů je patrné zvyšování energetické účinnosti, při zvyšování počtu stavů modulace. Pro 8 - stavovou modulaci je dosaženo bitové chybovosti menší než 10^{-3} od hodnoty E_b/N_0 7,5 dB, zatím co pro 64 - stavovou modulaci je dosaženo bitové chybovosti menší než 10^{-3} při hodnotě E_b/N_0 od 4,5 dB. Výsledné grafy bitové chybovosti i přes poměrné hrubé testování, lze využít pro představu, s jakou bitovou chybovostí se bude demodulovat přijímaný signál při konkrétní hodnotě E_b/N_0. Z rovnice 1.5, lze potom získat představu, s kolika stavovou modulací je nutné realizovat komunikaci pro stanovenou míru bitové chybovosti v případě snížení E_b/N_0.

Z důvodu velké časové náročnosti při testování bitové chybovosti, se nestihlo stanovit bitovou chybovost při úzkopásmovým rušením v AWGN kanálu. Princip testování je však totožný s výše uvedeným testováním chybovosti pro AWGN kanál a výsledky budou prezentovány při obhajobě diplomové práce.

Seznam literatury a informačních zdrojů

[1] CHENGBING HE, MAOHUA RAN, QINWEI MENG, JIANGUO HUANG. Underwater Acoustic Communications using M-ary Chirp-DPSK Modulation.

[2] VEŘTÁT, Ivo. *Efektivní komunikační systém pikosatelitů*. Plzeň, 2011. Disertační práce. Západočeská univerzita v Plzni. Vedoucí práce Doc. Ing. Jiří Masopust, Csc..

[3] The MathWorks, Inc. dsp.AudioFileReader System object. [online]. United States: The MathWorks, Inc. © 1994-2016 [cit. 2016-05-03]. Dostupné z: http://www.mathworks.com/help/dsp/ref/dsp.audiofilereader-class.html

[4] The MathWorks, Inc. dsp.AudioPlayer System object. [online]. United States: The MathWorks, Inc. © 1994-2016 [cit. 2016-05-03]. Dostupné z: http://www.mathworks.com/help/dsp/ref/dsp.audioplayer-class.html

[5] The MathWorks, Inc. dsp.AudioRecorder System object. [online]. United States: The MathWorks, Inc. © 1994-2016 [cit. 2016-05-03]. Dostupné z: http://www.mathworks.com/help/dsp/ref/dsp.audiorecorder-class.html

[6] The MathWorks, Inc. dsp.Buffer object. [online]. United States: The MathWorks, Inc. © 1994-2016 [cit. 2016-05-03]. Dostupné z: http://www.mathworks.com/help/dsp/ref/dsp.buffer-class.html

[7] The MathWorks, Inc. dsp.Crosscorrelator System object. [online]. United States: The MathWorks, Inc. © 1994-2016 [cit. 2016-05-03]. Dostupné z: http://www.mathworks.com/help/dsp/ref/dsp.crosscorrelator-class.html

[8] CEBOLLADA LÓPEZ, Sergio. *M-ary chirp modulation for low power bacup radio communication with picosatellites*. Plzeň, 2014. Diplomová práce. Západočeská univerzita v Plzni. Vedoucí práce Ing, Ivo Veřtát, Ph.D.

[9] ŽALUD, Václav. *Moderní radioelektrotechnika*. Praha: BEN - technická literatura, 2000. ISBN 80-86056-47-3.

Seznam zkratek a specifických názvů

BER	bitová chybovost
SER	symbolová chybovost
E_b/N_0	poměr potřebné energie na přenesení 1 bitu informace ku spektrální výkonové hustotě šumu
chirprate	označení rychlosti rozmítání frekvence
AWGN	model sdělovacího kanálu na který působý gaussovský šum

Seznam symbolů

P_A	výstupní úroveň výkonu zesilovače
C	hodnota užitečného přijímaného signálu
L_{RF}	ztráty mezi zesilovačem a vysílací anténou
G_{Ant1}	zisk vysílací antény
L_{Ant1}	ztráty nepřesným směrováním vysílací antény
L_0	ztráty šířením signálu ve volném prostředí
L_{Iono}	ztráty způsobené útlumem a odrazem signálu v ionosféře
L_{Atm}	ztráty v atmosféře
L_W	ztráty způsobené deštěm, oblačností a troposférických scintilací
G_{Ant2}	zisk přijímací antény
L_{Ant2}	ztráty nepřesným směrováním přijímací antény
L_P	polarizační ztráty
G_S	systémová rezerva
N_0	spektrální výkonová hustota šumu
C/N_0	odstup přijímaného užitečného signálu od spektrální výkonové hustoty šumu
v_p	přenosová rychlost
n	počet stavů modulace
y_{up}	symbol s kladnou hodnotou rychlosti rozmítání frekvence
y_{down}	symbol se zápornou hodnotou rychlosti rozmítání frekvence
f_{up}	horní mezní frekvence
f_{down}	dolní mezní frekvence
k	rychlost rozmítání frekvence
E	energie signálu během trvání celého symbolu
y_{K1}	symbol s kladnou hodnotou rychlosti rozmítání frekvence
y_{K2}	symbol se zápornou hodnotou rychlosti rozmítání frekvence
M	počet stavů modulace
α_K	rychlost rozmítání frekvence pro kladnou hodnotu rychlosti rozmítání frekvence
α'_K	rychlost rozmítání frekvence pro zápornou hodnotu rychlosti rozmítání frekvence

Příloha

Funkce modulátoru pro metodu půlení frekvenčního pásma

```
function y=chirpM_B_mod_time(t,symbol,fup,fdown,M,Fs)
%t - doba trvani modulacniho symbolu
%symbol - aktualni modulovany symbol
%fup - horni mezni fekvence
%fdown - dolni mezni fekvence
%M - pocet stavu modulace
%Fs - vzorkovaci frekvece

time=(0:1/Fs:t-(1/Fs)); % generovani vektoru casu


chirprate=((fup-fdown)/(M/2))*(1/t);
%rychlost rozmitani frekvence se zmenou podle casu

switch mod(symbol,2)%pro liché 1 a pro sudé 0
    case 0
    y=sin(2*pi.*((fdown+((fup-fdown)/(M/2))*floor...
     (symbol/2)).*time+(chirprate/2).*time.^2));
    case 1
    y=sin(2*pi.*((fup-((fup-fdown)/(M/2))*floor...
     ((M-symbol)/2)).*time+(-chirprate/2).*time.^2));
end;
```

Funkce modulátoru pro metodu půlení časového intervalu

```
function y=chirpM_mod_time(t,symbol,fup,fdown,M,Fs)
%t - doba trvani modulacniho symbolu
%symbol - aktualni modulovany symbol
%fup - horni mezni fekvence
%fdown - dolni mezni fekvence
%M - pocet stavu modulace
%Fs - vzorkovaci frekvece


time=(0:1/Fs:(t/2)-(1/Fs)); % generovani vektoru casu

for i=1:(M/2)
    chirprate1(i)=i*((fup-fdown)/((M/2)+1))/(0.5*t);
    chirprate2(i)=((M/2)-i+1)*((fup-fdown)/((M/2)+1))/(0.5*t);
end;
for i=(M/2)+1:M
    chirprate1(i)=-chirprate1(i-(M/2));
    chirprate2(i)=-chirprate2(i-(M/2));
end;
for i=1:(M/2)
```

```
    freq_middle(i)=fdown+i*((fup-fdown)/((M/2)+1));
%frekvencni stred, tam kde se meni rychlost rozmitani
%frekvence
end;
for i=(M/2)+1:M
    freq_middle(i)=freq_middle((M)-i+1);
%frekvencni stred, tam kde se meni rychlost rozmitani
%frekvence
end;
switch fix(symbol/(M/2))
%fix zaokrouhluje dolu->prvni 4 symboly jsou case 0 druhy
%ctyri case 1 =>rozdeleni sklonu
  case 0
  y=[sin(2*pi.*(fdown.*time+(chirprate1(symbol+1)/2).*...
time.^2))sin(2*pi.*(fdown*(t/2)+(chirprate1(symbol+1)/2)*...
(t/2)^2)+2*pi.*((freq_middle(symbol+1)).*time+(chirprate2...
(symbol+1)/2).*time.^2))];
  case 1
  y=[sin(2*pi.*(fup.*time+(chirprate1(symbol+1)/2).*...
time.^2))sin(2*pi.*(fup*(t/2)+(chirprate1(symbol+1)/2)*...
(t/2)^2)+2*pi.*((freq_middle(symbol+1)).*time+(chirprate2...
(symbol+1)/2).*time.^2))];
end;
```

Funkce pro vytváření modulačních symbolů

```
function  zaznam_mod_symbolu( t,fup,fdown,M,Fs,typ )
%t - doba trvani modulacniho symbolu
%symbol - aktualni modulovany symbol
%fup - horni mezni fekvence
%fdown - dolni mezni fekvence
%M - pocet stavu modulace
%Fs - vzorkovaci frekvece

switch typ
    case 1
for i=0:M-1
y=chirpM_mod_time(t,i,fup,fdown,M,Fs);    %volani funkce
switch i                                  %pojmenovani podle
nahraneho symbolu
    case 0
        filename = 'Symbol_0.wav';
    case 1
        filename = 'Symbol_1.wav';
    case 2
        filename = 'Symbol_2.wav';
    case 3
        filename = 'Symbol_3.wav';
                .
                .
                .
```

```
    case 63
        filename = 'Symbol_63.wav';
    otherwise
        disp('chyba pri zaznamu symbolu');
end
audiowrite(filename,y,Fs);%zapsani do wav.souboru
end
 case 2
for i=0:M-1
y=chirpM_B_mod_time(t,i,fup,fdown,M,Fs);
%volani funkce
switch i  %pojmenovani podle nahraneho symbolu
    case 0
        filename = 'Symbol_0.wav';
    case 1
        filename = 'Symbol_1.wav';
    case 2
        filename = 'Symbol_2.wav';
    case 3
        filename = 'Symbol_3.wav';
                .
                .
                .
    case 63
        filename = 'Symbol_63.wav';
    otherwise
        disp('chyba pri zaznamu symbolu');
end
audiowrite(filename,y,Fs); %zapsani do wav.souboru
end
end
```

Funkce pro demodulaci signálu

```
function [ Symbol ] = dem_chirp(y)

%nacteni jednotlivych symbolu
%
AFR_0=dsp.AudioFileReader('Symbol_0.wav','SamplesPerFrame',800
0,'OutputDataType','single');   %nacteni jednotlivych symbolu
AFR_1=dsp.AudioFileReader('Symbol_1.wav','SamplesPerFrame',800
0,'OutputDataType','single');
AFR_2=dsp.AudioFileReader('Symbol_2.wav','SamplesPerFrame',800
0,'OutputDataType','single');
AFR_3=dsp.AudioFileReader('Symbol_3.wav','SamplesPerFrame',800
0,'OutputDataType','single');
                                .
                                .
                                .
AFR_63=dsp.AudioFileReader('Symbol_63.wav','SamplesPerFrame',8
000,'OutputDataType','single');
```

```
xcorr = dsp.Crosscorrelator('Method','Fastest');

symbolyMod=zeros(8000,63);  %modulacni symboly
    symbolyMod(:,1)=step(AFR_1);
    symbolyMod(:,2)=step(AFR_2);
    symbolyMod(:,3)=step(AFR_3);
                .
                .
                .
    symbolyMod(:,64)=step(AFR_0);

PomSym=zeros(1,63);  %vytvareni pomocnych matic
PomMax=zeros(1,63);
Symbol=0;

                PomXcorr=abs(step(xcorr,symbolyMod,y));
%
                PomMax=max(PomXcorr);

%
                        PomSym(1,1)=PomMax(1,1);
                        PomSym(1,2)=PomMax(1,2);
                        PomSym(1,3)=PomMax(1,3);
                                .
                                .
                                .
                        PomSym(1,63)=PomMax(1,63);
    %hledani max hodnoty, ta je na pozici I
                        [AMP,S] = max(PomSym);


                        switch S    %roztrideni modulacnich
symbolu a vlozeni do vektoru "data"
                            case 1  %symbol urcen podle pozice
maxima
                                Symbol=1;
                            case 2
                                Symbol=2;
                            case 3
                                .
                                .
                                .
                            case 63
                                Symbol=63;
                            otherwise
                            disp('symbol nerozeznan');
                        end

                end
```

Program pro demodulaci v reálném čase

```
clear all;
close all;
%   AFR_0 az AFR_7 jsou audiosoubory vzorovych modulacnich
souboru

%   InBuff je vstupni buffer dat
%   audioIn do teto promenne se ukladaji nactena data z
mikrofonu
%   dataIn do teto promenne se ukladaji nactena data do
bufferu
%   xcorr
%   dataIn promena do ktere se nactou data z bufferu
Fs=8000;

AFR_0=dsp.AudioFileReader('Symbol_0.wav','SamplesPerFrame',800
0,'OutputDataType','single');   %nacteni jednotlivych symbolu
                        .
                        .
                        .
AFR_63=dsp.AudioFileReader('Symbol_63.wav','SamplesPerFrame',8
000,'OutputDataType','single');

Mic=dsp.AudioRecorder('SamplesPerFrame',8000,'BufferSizeSource
','Property','BufferSize',8000,'QueueDuration',0,'NumChannels'
,1,'OutputDataType','double'); %pocet vzorku v jednom ramci
Mic.DeviceName='Default';
Mic.SampleRate=Fs;

InBuff = dsp.Buffer(16000,8000);
%velikost bufferu 16000, prevzorkovani 8000
xcorr = dsp.Crosscorrelator;   %('Method','Fastest')
dataSymbolu=ones(31,1);    %velikost ramce

% data_ref=step(AFR_0);   %synchronizacni symbol

symbolyMod=zeros(8000,64);  %modulacni symboly
    symbolyMod(:,1)=step(AFR_1);
    symbolyMod(:,2)=step(AFR_2);
                .
                .
                .
    symbolyMod(:,64)=step(AFR_0); %synchronizacni symbol
PomSym=zeros(1,63);
PomMax=zeros(1,63);
DATA_RAMCE=zeros(31,2);
for a=0:1
ramec=a
tic;
s=0;
```

```
release(xcorr);

while ((toc<100)&&(s==0));

dataIn = step(InBuff, step(Mic));
%nacteny prvni dva ramce z mikrofonu


 if (s==0)
    y = step(xcorr,symbolyMod(:,16),dataIn); %funguje
%hledani max hodnoty, ta je na pozici I

    [mv,I] = max(abs(y));
       poziceMax =I;
       delayMax = poziceMax/Fs;
       aktualSymbol=-I+16001
         if ((aktualSymbol<8000)&&(aktualSymbol>0))
%pokud je aktualSymbol vetsi nez 8000 tak je za polovinou
bufferu, to znamena ze mi prijde v pristim stepu
             nextSymbol=-I+24001;
             c=mv/mean(abs(y))
             figure(1);
             plot(y); title('Correlated output')


             if(c>(20))
               s=1;
               maxim=y(I);
               disp('povedlo se najit synchronizacni symbol');

%********kod pro urceni modulacnich symbolu*********

            frameCounter=1;     %pocitadlo ramce
            release(xcorr);
%             release(Mic);
               while frameCounter < 32   %127 cislo podle
delky ramce

               dataIn = step(InBuff, step(Mic));


PomXcorr=abs(step(xcorr,symbolyMod(:,(1:15)),...
          dataIn(aktualSymbol:(aktualSymbol+7999))));
%vypocet korelace pouze z bufferu kde se nachazi modulacni
symbol
                [PomMax,PozMax]=max(PomXcorr(7950:8050,:));
%protoze symboly mam zarovnany a porovnavam 8000vzorku s 8000
vzorky a maximmum corelace bude vždy na pozici 8000

               figure(2);
```

```
                plot(PomXcorr(7500:15500,:));
                title('Correlated symbols')
%porovnani jedn. korelaci v grafu

%
PomSym(1,1)=PomMax(1,1)/mean(abs(PomXcorr(I:I+8000,1)));
%normovani pro urceni prijimanyho symbolu
                PomSym(1,1)=PomMax(1,1)/mean(abs(PomXcorr...
                           ([1:7990,8010:15999],1)));
                    .
                    .
                    .

                PomSym(1,63)=PomMax(1,63)/mean(abs(PomXcorr...
                            ([1:7990,8010:15999],63)));

                [AMP,S] = max(PomSym)     %hledani max hodnoty,
ta je na pozici I

                 if (AMP<20)
                   S=65; %pokud bude hodnota corelace mensi
nez 20, tak bude symbol povazovany za nerozeznany
                 end

   %roztrideni modulacnich symbolu a vlozeni do vektoru "data"
                    switch S
                        case 1  %symbol urcen podle pozice
                                %maxima
                            dataSymbolu(frameCounter)=1;

                        case 2
                            dataSymbolu(frameCounter)=2;
                        case 3
                            dataSymbolu(frameCounter)=3;
                                   .
                                   .
                                   .
                        case 63
                            dataSymbolu(frameCounter)=63;
                        otherwise
                            dataSymbolu(frameCounter)=NaN;
                            disp('symbol nerozeznan');
                    end
                    frameCounter=(frameCounter+1);
                end
                DATA_RAMCE(:,a+1)=dataSymbolu(1:31,1);
            else
                disp('nepovedlo se najit synchronizacni
symbol');
            end
        else
```

```
            disp('korelace se bude provadet az priste');
        end
 end
end
% a=a+1;
end

release(AFR_0);              % smazani vstupnich souboru
release(AFR_1);
     .
     .
     .
release(AFR_63);
release(Mic);
release(xcorr);
```

Testování chybovosti v reálném času - modulátor

```
close all;
clear all;

%*********nacteni dat pro odeslani*************
load('odeslana_posloupnost.mat','odeslany_data')
%********nacteni mod. symbolu********************

AFR_0=dsp.AudioFileReader('Filename','Symbol_0.wav','SamplesPe
rFrame',8000,'OutputDataType','single');   %nacteni
jednotlivych symbolu
AFR_1=dsp.AudioFileReader('Filename','Symbol_1.wav','SamplesPe
rFrame',8000,'OutputDataType','single');
AFR_2=dsp.AudioFileReader('Filename','Symbol_2.wav','SamplesPe
rFrame',8000,'OutputDataType','single');
                                  .
                                  .
                                  .
AFR_63=dsp.AudioFileReader('Filename','Symbol_63.wav','Samples
PerFrame',8000,'OutputDataType','single');
Fs = 8000;

AP=dsp.AudioPlayer('SampleRate',Fs,'QueueDuration',0);
%inicializace prehravani


ebno=-4:0.5:10;%vector of Eb/No
out=0;
BER(1:length(ebno))=NaN;
SER(1:length(ebno))=NaN;
b=1;
chirp_dim=16;
POCET_CYKLU=length(ebno)
```

```
for q=1:length(ebno)
    en=ebno(q);
    snr=en+10*log10(log2(chirp_dim))-10*log10(Fs/2);
%SNR pro konkrétní hodnotu Eb/N0 a počet stavů modulace
    pocet=0;

    if q<=9 %pro ebno=0
        for i=1:5   %40
% posilani dat
            posloupnost = (odeslany_data(i));

            pocet=pocet+1

            if posloupnost==0
%porovnani symbolu v posloupnosti(vektor H) s odpovidajicim
symbolem
            audioIn=step(AFR_0);
%             Symbol0Snr=awgn(audioIn,snr,0);
            step(AP,audioIn); %a jeho prehrani

            elseif posloupnost==1
            audioIn=step(AFR_1);
            Symbol1Snr=awgn(audioIn,snr,0)./200;
%           step(CasoveZobrazeni, audioIn);
%           step(Spectrum, audioIn);
            step(AP,Symbol1Snr);
                    .
                    .
                    .

            elseif posloupnost==63
            audioIn=step(AFR_63);
            Symbol63Snr=awgn(audioIn,snr,0)./200;
%           step(CasoveZobrazeni, audioIn);
%           step(Spectrum, audioIn);
            step(AP,Symbol63Snr);
            end
        end

    elseif q>9&&q<=17   %ebno od 0 do 4
        for i=1:5   %150
% posilani dat

            posloupnost = (odeslany_data(i));

            pocet=pocet+1

            if posloupnost==0
%porovnani symbolu v posloupnosti(vektor H) s odpovidajicim
symbolem
```

```
            audioIn=0.1*step(AFR_0);
%             Symbol0Snr=awgn(audioIn,snr,0);
            step(AP,audioIn); %a jeho prehrani

            elseif posloupnost==1
            audioIn=0.1*step(AFR_1);
            Symbol1Snr=awgn(audioIn,snr,0)./200;
%           step(CasoveZobrazeni, audioIn);
%           step(Spectrum, audioIn);
            step(AP,Symbol1Snr);

                    .
                    .
                    .

            elseif posloupnost==63
            audioIn=step(AFR_63);
            Symbol63Snr=awgn(audioIn,snr,0)./200;
%           step(CasoveZobrazeni, audioIn);
%           step(Spectrum, audioIn);
            step(AP,Symbol63Snr);

            end
        end
    elseif q>17&&q<=21  %ebno od 4 do 6
        for i=1:5   %450

            posloupnost = (odeslany_data(i));

            pocet=pocet+1

            if posloupnost==0
%porovnani symbolu v posloupnosti(vektor H) s odpovidajicim
symbolem
            audioIn=step(AFR_0);
%             Symbol0Snr=awgn(audioIn,snr,0);
            step(AP,audioIn); %a jeho prehrani

            elseif posloupnost==1
            audioIn=step(AFR_1);
            Symbol1Snr=awgn(audioIn,snr,0)./200;
%           step(CasoveZobrazeni, audioIn);
%           step(Spectrum, audioIn);
            step(AP,Symbol1Snr);
                    .
                    .
                    .
            elseif posloupnost==63
            audioIn=step(AFR_63);
            Symbol63Snr=awgn(audioIn,snr,0)./200;
%           step(CasoveZobrazeni, audioIn);
```

```
%           step(Spectrum, audioIn);
            step(AP,Symbol63Snr);

            end
        end


        elseif q>21&&q<=25  %ebno od 6 do 8
        for i=1: 10 %4800
% Stream data into MATLAB ((~isDone(H))&&(count<15))
            posloupnost = (odeslany_data(i));

            pocet=pocet+1

            if posloupnost==0
%porovnani symbolu v posloupnosti(vektor H) s odpovidajicim
symbolem
            audioIn=step(AFR_0);
%             Symbol0Snr=awgn(audioIn,snr,0);
            step(AP,audioIn); %a jeho prehrani

            elseif posloupnost==1
            audioIn=step(AFR_1);
            Symbol1Snr=awgn(audioIn,snr,0)./200;
%           step(CasoveZobrazeni, audioIn);
%           step(Spectrum, audioIn);
            step(AP,Symbol1Snr);
                    .
                    .
                    .

            elseif posloupnost==63
            audioIn=step(AFR_63);
            Symbol63Snr=awgn(audioIn,snr,0)./200;
%           step(CasoveZobrazeni, audioIn);
%           step(Spectrum, audioIn);
            step(AP,Symbol63Snr);
            end
        end

        elseif q>25&&q<=29  %ebno od 8 do 10
        for i=1:10  %32000
% Stream data into MATLAB ((~isDone(H))&&(count<15))
            posloupnost = (odeslany_data(i));

            pocet=pocet+1

            if posloupnost==0
%porovnani symbolu v posloupnosti(vektor H) s odpovidajicim
symbolem
            audioIn=step(AFR_0);
```

```
%              Symbol0Snr=awgn(audioIn,snr,0);
              step(AP,audioIn); %a jeho prehrani

              elseif posloupnost==1
              audioIn=step(AFR_1);
              Symbol1Snr=awgn(audioIn,snr,0)./200;
%            step(CasoveZobrazeni, audioIn);
%            step(Spectrum, audioIn);
              step(AP,Symbol1Snr);
                     .
                     .
                     .

elseif posloupnost==63
              audioIn=step(AFR_63);
              Symbol63Snr=awgn(audioIn,snr,0)./200;
%            step(CasoveZobrazeni, audioIn);
%            step(Spectrum, audioIn);
              step(AP,Symbol63Snr);
              end
          end


    end
end
pause(AP.QueueDuration);     % čekání na konec přehrání audioIn
release(AFR_0);              % smazání vstupních souborů
release(AFR_1);
release(AFR_2);
      .
      .
      .
release(AFR_63);
release(AP);
```

Testování chybovosti v reálném času - demodulátor

```
clear all;
close all;
%   AFR_0 az AFR_63 jsou audiosoubory vzorovych modulacnich
souboru

%   InBuff je vstupni buffer dat
%   audioIn do teto promenne se ukladaji nactena data z
mikrofonu
%   dataIn promena do ktere se nactou data z bufferu
load('odeslana_posloupnost.mat','odeslany_data')
ODESLANY_DATA=odeslany_data';
```

```
ebno=-4:0.5:10;%vector of Eb/No
BER(1:length(ebno))=NaN;
SER(1:length(ebno))=NaN;
pocet_zasynchronizovani=0;
CHYBA_DEMOD=0;


Fs=8000;

AFR_0=dsp.AudioFileReader('Symbol_0.wav','SamplesPerFrame',800
0,'OutputDataType','single');   %nacteni jednotlivych symbolu
AFR_1=dsp.AudioFileReader('Symbol_1.wav','SamplesPerFrame',800
0,'OutputDataType','single');
                .
                .
                .
AFR_63=dsp.AudioFileReader('Symbol_63.wav','SamplesPerFrame',8
000,'OutputDataType','single');

Mic=dsp.AudioRecorder('SamplesPerFrame',8000,'BufferSizeSource
','Property','BufferSize',8000,'QueueDuration',0,'NumChannels'
,1,'OutputDataType','double'); %pocet vzorku v jednom ramci
Mic.DeviceName='Default';
Mic.SampleRate=Fs;

InBuff = dsp.Buffer(16000,8000);    %velikost bufferu 16000,
prevzorkovani 8000
xcorr = dsp.Crosscorrelator;   %('Method','Fastest')
dataSymbolu=zeros(320,1);

% data_ref=step(AFR_0);   %synchronizacni symbol

symbolyMod=zeros(8000,64);  %modulacni symboly
    symbolyMod(:,1)=step(AFR_1);
    symbolyMod(:,2)=step(AFR_2);
                .
                .
                .
    symbolyMod(:,63)=step(AFR_63);
    symbolyMod(:,64)=step(AFR_0);
%     synchSymbol=step(AFR_0);
PomSym=zeros(1,63);
PomMax=zeros(1,63);
DATA_RAMCE=zeros(63,2);
counter=0;

chirp_dim=64;
for q=1:length(ebno) %29
    BER(q)=0;
    SER(q)=0;
    s=0;
```

```
    tic;
    release(xcorr);
    EbN0=ebno(q)
    while ((toc<100)&&(s==0));
    % audioIn = step(Mic);
    % dataIn = step(InBuff, audioIn);   %nacteny prvni dva
ramce z mikrofonu

    dataIn = step(InBuff, step(Mic));   %nacteny prvni dva
ramce z mikrofonu


    if (s==0)
        y = step(xcorr,symbolyMod(:,64),dataIn); %funguje
        [mv,I] = max(abs(y));    %hledani max hodnoty, ta je
na pozici I
%         poziceMax =I;
%         delayMax = poziceMax/Fs;
        aktualSymbol=-I+16001
            if ((aktualSymbol<8000)&&(aktualSymbol>0))  %pokud
je aktualSymbol vetsi nez 8000 tak je za polovinou bufferu, to
znamena ze mi prijde v pristim stepu
%                 nextSymbol=-I+24001;
                c=mv/mean(abs(y))
%                 figure(1);
%                 plot(y); title('Correlated output')

                if(c>20)
                    s=1;

pocet_zasynchronizovani=pocet_zasynchronizovani+1;
%pocitadlo uspesne synchronizace
                    disp('povedlo se najit synchronizacni
symbol');

%********kod pro urceni modulacnich symbolu*********

                    frameCounter=1;     %pocitadlo ramce
                    release(xcorr);
%                   release(Mic);
                   if q<=9
                    while frameCounter < 5   %127 cislo podle
delky ramce
                        counter=counter+1;
                        dataIn = step(InBuff, step(Mic));


PomXcorr=abs(step(xcorr,symbolyMod(:,(1:63)),dataIn(aktualSymb
ol:(aktualSymbol+7999)))); %vypocet korelace pouze z bufferu
kde se nachazi modulacni symbol
```

```
[PomMax,PozMax]=max(PomXcorr(7950:8050,:));   %protoze symboly
mam zarovnany a porovnavam 8000vzorku s 8000 vzorky a maximmum
corelace bude vždy na pozici 8000


                        PomSym(1,1)=PomMax(1,1);
                        PomSym(1,2)=PomMax(1,2);
                              .
                              .
                              .
                        PomSym(1,63)=PomMax(1,63);

                        [AMP,S] = max(PomSym)
%hledani max hodnoty, ta je na pozici I


                        switch S
%roztrideni modulacnich symbolu a vlozeni do vektoru "data"
                            case 1  %symbol urcen podle pozice
maxima
                                dataSymbolu(frameCounter)=1;
                            case 2
                                dataSymbolu(frameCounter)=2;
                                   .
                                   .
                                   .
                            case 63
                                dataSymbolu(frameCounter)=63;
                            otherwise
                                dataSymbolu(frameCounter)=0;
                                disp('symbol nerozeznan');
                        end

%*************************************************************
*****
if(dataSymbolu(frameCounter)~=ODESLANY_DATA(frameCounter+1))
%porovnani zda se odeslany data rovnaji demodulovanym datum

CHYBA_DEMOD=sum(ne(dec2bin(ODESLANY_DATA(frameCounter+1),...
     log2(chirp_dim)),dec2bin(dataSymbolu(frameCounter),...
     log2(chirp_dim))));
   %prevedeni dek cisla na bin a zjisteni kolik bitu je
rozdilnych
                            BER(q)=BER(q)+CHYBA_DEMOD;
%musim zajistit pricitani q
                            SER(q)=SER(q)+1;    %zjistovani
kolik bylo chybnych symbolu

                        end;
                        frameCounter=(frameCounter+1);
```

```
                    end
                   elseif q>9&&q<=17
                         while frameCounter < 10   %127 cislo
podle delky ramce
                        counter=counter+1;
                        dataIn = step(InBuff, step(Mic));


PomXcorr=abs(step(xcorr,symbolyMod(:,(1:63)),dataIn(aktualSymb
ol:(aktualSymbol+7999)))); %vypocet korelace pouze z bufferu
kde se nachazi modulacni symbol

[PomMax,PozMax]=max(PomXcorr(7950:8050,:));   %protoze symboly
mam zarovnany a porovnavam 8000vzorku s 8000 vzorky a maximmum
corelace bude vždy na pozici 8000


                        PomSym(1,1)=PomMax(1,1);
                        PomSym(1,2)=PomMax(1,2);
                              .
                              .
                              .
                        PomSym(1,63)=PomMax(1,63);

                        [AMP,S] = max(PomSym)    %hledani max
hodnoty, ta je na pozici I


                        switch S    %roztrideni modulacnich
symbolu a vlozeni do vektoru "data"
                            case 1  %symbol urcen podle pozice
maxima
                                dataSymbolu(frameCounter)=1;
                            case 2
                                   .
                                   .
                                   .
                            case 63
                                dataSymbolu(frameCounter)=63;
                            otherwise
                                dataSymbolu(frameCounter)=0;
                                disp('symbol nerozeznan');
                        end

%*************************************************************
*****
if(dataSymbolu(frameCounter)~=ODESLANY_DATA(frameCounter+1))
%porovnani zda se odeslany data rovnaji demodulovanym datum

CHYBA_DEMOD=sum(ne(dec2bin(ODESLANY_DATA(frameCounter+1),...
     log2(chirp_dim)),dec2bin(dataSymbolu(frameCounter),...
```

```
     log2(chirp_dim))));
%prevedeni dek cisla na bin a zjisteni kolik bitu je
rozdilnych
                              BER(q)=BER(q)+CHYBA_DEMOD;
%musim zajistit pricitani q
                              SER(q)=SER(q)+1;    %zjistovani
kolik bylo chybnych symbolu

                          end;
                          frameCounter=(frameCounter+1);
                          end

                     elseif q>17&&q<=21
                         while frameCounter < 15   %127 cislo
podle delky ramce
                          counter=counter+1;
                          dataIn = step(InBuff, step(Mic));


PomXcorr=abs(step(xcorr,symbolyMod(:,(1:63)),dataIn(aktualSymb
ol:(aktualSymbol+7999)))); %vypocet korelace pouze z bufferu
kde se nachazi modulacni symbol
%                          PomMax=max(abs(PomXcorr(8000:end,:)));
%max abs hodnota pro hodnoty korelace 8000:end

[PomMax,PozMax]=max(PomXcorr(7950:8050,:));   %protoze symboly
mam zarovnany a porovnavam 8000vzorku s 8000 vzorky a maximmum
corelace bude vždy na pozici 8000

                          PomSym(1,1)=PomMax(1,1);
                          PomSym(1,2)=PomMax(1,2);
                                .
                                .
                                .
                          PomSym(1,63)=PomMax(1,63);

                          [AMP,S] = max(PomSym)    %hledani max
hodnoty, ta je na pozici I


                          switch S    %roztrideni modulacnich
symbolu a vlozeni do vektoru "data"
                              case 1  %symbol urcen podle pozice
maxima
                                  dataSymbolu(frameCounter)=1;
                              case 2
                                  dataSymbolu(frameCounter)=2;
                                    .
                                    .
                                    .
                              case 63
```

```
                                dataSymbolu(frameCounter)=63;
                            otherwise
                                dataSymbolu(frameCounter)=0;
                                disp('symbol nerozeznan');
                        end

%*************************************************************
*****
if(dataSymbolu(frameCounter)~=ODESLANY_DATA(frameCounter+1))
%porovnani zda se odeslany data rovnaji demodulovanym datum

CHYBA_DEMOD=sum(ne(dec2bin(ODESLANY_DATA(frameCounter+1),...
     log2(chirp_dim)),dec2bin(dataSymbolu(frameCounter),...
     log2(chirp_dim))));
%prevedeni dek cisla na bin a zjisteni kolik bitu je
rozdilnych
                            BER(q)=BER(q)+CHYBA_DEMOD;
%musim zajistit pricitani q
                            SER(q)=SER(q)+1;
%zjistovani kolik bylo chybnych symbolu

                        end;
                        frameCounter=(frameCounter+1);
                        end
                    elseif q>21&&q<=25
                        while frameCounter < 20
                        counter=counter+1;
                        dataIn = step(InBuff, step(Mic));


PomXcorr=abs(step(xcorr,symbolyMod(:,(1:63)),dataIn(aktualSymb
ol:(aktualSymbol+7999)))); %vypocet korelace pouze z bufferu
kde se nachazi modulacni symbol
%                       PomMax=max(abs(PomXcorr(8000:end,:)));
%max abs hodnota pro hodnoty korelace 8000:end

[PomMax,PozMax]=max(PomXcorr(7950:8050,:));   %protoze symboly
mam zarovnany a porovnavam 8000vzorku s 8000 vzorky a maximmum
corelace bude vždy na pozici 8000


                        PomSym(1,1)=PomMax(1,1);
                        PomSym(1,2)=PomMax(1,2);
                              .
                              .
                              .
                        PomSym(1,63)=PomMax(1,63);

                        [AMP,S] = max(PomSym)
%hledani max hodnoty, ta je na pozici I
```

```
                        switch S    %roztrideni modulacnich
symbolu a vlozeni do vektoru "data"
                            case 1  %symbol urcen podle pozice
maxima
                                dataSymbolu(frameCounter)=1;
                            case 2
                                dataSymbolu(frameCounter)=2;
                                   .
                                   .
                                   .
                            case 63
                                dataSymbolu(frameCounter)=63;
                            otherwise
                                dataSymbolu(frameCounter)=0;
                                disp('symbol nerozeznan');
                        end

%***********************************************************
*****

if(dataSymbolu(frameCounter)~=ODESLANY_DATA(frameCounter+1))
%porovnani zda se odeslany data rovnaji demodulovanym datum

CHYBA_DEMOD=sum(ne(dec2bin(ODESLANY_DATA(frameCounter+1),...
     log2(chirp_dim)),dec2bin(dataSymbolu(frameCounter),...
     log2(chirp_dim))));
%prevedeni dek cisla na bin a zjisteni kolik bitu je
rozdilnych
                            BER(q)=BER(q)+CHYBA_DEMOD;
%musim zajistit pricitani q
                            SER(q)=SER(q)+1;    %zjistovani
kolik bylo chybnych symbolu

                        end;
                        frameCounter=(frameCounter+1);
                        end
                    elseif q>25&&q<=29

                       while frameCounter < 20   %127 cislo
podle delky ramce
                        counter=counter+1;
                        dataIn = step(InBuff, step(Mic));


PomXcorr=abs(step(xcorr,symbolyMod(:,(1:63)),dataIn(aktualSymb
ol:(aktualSymbol+7999))));
%vypocet korelace pouze z bufferu kde se nachazi modulacni
%symbol

[PomMax,PozMax]=max(PomXcorr(7950:8050,:));   %protoze symboly
```

```
mam zarovnany a porovnavam 8000vzorku s 8000 vzorky a maximmum
corelace bude vždy na pozici 8000

                        PomSym(1,1)=PomMax(1,1);
                        PomSym(1,2)=PomMax(1,2);
                              .
                              .
                              .
                        PomSym(1,63)=PomMax(1,63);

                        [AMP,S] = max(PomSym)    %hledani max
hodnoty, ta je na pozici I

%                         if (AMP<20)
%                             S=16; %pokud bude hodnota
corelace mensi nez 20, tak bude symbol povazovany za
nerozeznany
%                         end


                        switch S    %roztrideni modulacnich
symbolu a vlozeni do vektoru "data"
                            case 1  %symbol urcen podle pozice
maxima
                                dataSymbolu(frameCounter)=1;
                            case 2
                                dataSymbolu(frameCounter)=2;
                                   .
                                   .
                                   .
                            case 63
                                dataSymbolu(frameCounter)=63;
                            otherwise
                                dataSymbolu(frameCounter)=0;
                                disp('symbol nerozeznan');
                        end

%***********************************************************
*****

if(dataSymbolu(frameCounter)~=ODESLANY_DATA(frameCounter+1))
%porovnani zda se odeslany data rovnaji demodulovanym datum

CHYBA_DEMOD=sum(ne(dec2bin(ODESLANY_DATA(frameCounter+1),...
     log2(chirp_dim)),dec2bin(dataSymbolu(frameCounter),...
     log2(chirp_dim))));
%prevedeni dek cisla na bin a zjisteni kolik bitu je
rozdilnych
                            BER(q)=BER(q)+CHYBA_DEMOD;
%musim zajistit pricitani q
                            SER(q)=SER(q)+1;
```

```
%zjistovani kolik bylo chybnych symbolu

                        end;
                        frameCounter=(frameCounter+1);
                        end

end%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                else
                    disp('nepovedlo se najit synchronizacni
symbol');
                end
            else
                disp('korelace se bude provadet az priste');
            end
    end
    end
    BER(q)=BER(q)/(log2(chirp_dim)*(frameCounter-1));
%     b=BER(q);
    SER(q)=SER(q)/(frameCounter-1);
%     figure(3)
%     semilogy(ebno,BER);drawnow;
%     axis([-4 10 10^-5 1]);
end

release(AFR_0);             % smazani vstupnich souboru
release(AFR_1);
     .
     .
     .
release(AFR_63);
release(Mic);
release(xcorr);
```