VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

LDPC KÓDY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

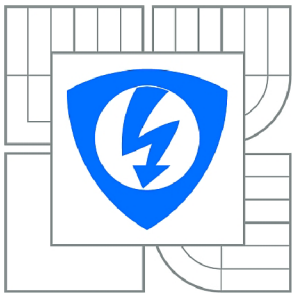
AUTOR PRÁCE
AUTHOR

Bc. ONDŘEJ HROUZA

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

LDPC KÓDY

LDPC CODES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

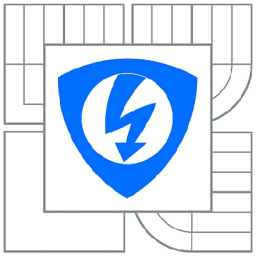
AUTOR PRÁCE — Bc. ONDŘEJ HROUZA

AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL ŠILHAVÝ, Ph.D.

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Ondřej Hrouza ***ID:*** 106476
Ročník: 2 ***Akademický rok:*** 2011/2012

NÁZEV TÉMATU:

LDPC kódy

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku protichybového zabezpečení pomocí LDPC (Low-Density Parity Check) kódů. Zaměřte jak na problematiku vytváření kódu, tak na dekódování zprávy zabezpečené pomocí LDPC kódů. Porovnejte jednotlivé přístupy. V programovém prostředí Matlab vytvořte demonstrační program, který bude možno využít jako výukovou pomůcku a rovněž jako analyzačního nástroje pro porovnání dosažených parametrů kódů. Program bude zahrnovat grafické rozhranní a jednotlivé položky v něm doplňte nápovědou. S pomocí vytvořeného programu realizujte zevrubné srovnání dosažených parametrů (kódového zisku a dalších) pro různé parametry kódů.

DOPORUČENÁ LITERATURA:

[1] Moreira,J.C., Farrel, P.G.. Essentials of Error-Control Coding. JohnWiley, 2006, ISBN: 0-470-02920-X.
[2] Moon, T. K. Error Correction Coding: Mathematical Methods and Algorithms. Wiley-Interscience, 2005, ISBN-13: 978-0070010697.
[3] Lin, S., Costello, D. J.. Error Control Coding: Fundamentals and Applications, second edition, Prentice Hall: Englewood Cliffs, NJ, 2005, ISBN: 0-13-042672-5.
[4] Sweeney, P.. Error control coding: from theory to practice, . Wiley-Interscience, 2002, ISBN: 0-470-84356-X.

Termín zadání: 6.2.2012 ***Termín odevzdání:*** 24.5.2012

Vedoucí práce: Ing. Pavel Šilhavý, Ph.D.
Konzultanti diplomové práce:

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

ABSTRAKT

Práce se zabývá problematikou LDPC kódů. Jsou zde popsány metody vytváření paritní matice, kde je kladen důraz především na strukturované vytváření této matice za použití konečné geometrie: Euklidovské geometrie a projektivní geometrie. Další oblastí, které se práce věnuje je dekódování LDPC kódů. Práce porovnává čtyři dekódovací metody: Hard-Decision algoritmus, Bit-Flipping algoritmus, The Sum-Product algoritmus a Log Likelihood algoritmus, při kterých je kladen důraz především na iterativní dekódovací metody. Praktickým výstupem práce je program LDPC kódy, který vznik v prostředí Matlab. Tento program je rozdělen na dvě části – Výuka LDPC kódů a Simulace LDPC kódů. Na základě výsledků získaných z programu Simulace LDPC kódů je vytvořeno porovnání vytvářecích a dekódovacích metod LDPC kódů. Pro porovnávání dekódovacích metod LDPC kódů byly využity BER charakteristiky a časová závislost jednotlivých metod na různých parametrech LDPC kódu (počet iterací nebo velikost paritní matice).

KLÍČOVÁ SLOVA

LDPC kódy, Euklidovská geometrie, projektivní geometrie, iterativní dekódování, Matlab, BER.

ABSTRACT

The aim of this thesis are problematics about LDPC codes. There are described metods to create parity check matrix, where are important structured metods using finite geometry: Euclidean geometry and projectice geometry. Next area in this thesis is decoding LDPC codes. There are presented four metods: Hard-Decision algorithm, Bit-Flipping algorithm, The Sum-Product algorithm and Log Likelihood algorithm, where is mainly focused on iterative decoding methods. Practical output of this work is program LDPC codes created in environment Matlab. The program is divided to two parts – Practise LDPC codes and Simulation LDPC codes. The result reached by program Simulation LDPC codes is used to create a comparison of creating and decoding methods LDPC codes. For comparison of decoding methods LDPC codes were used BER characteristics and time dependence each method on various parameters LDPC code (number of iteration or size of parity matrix).

KEYWORDS

LDPC codes, Euclidean geometry, projectice geometry, iterative decoding, Matlab, BER.

HROUZA, Ondřej *LDPC kódy*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 80 s. Vedoucí práce byl Ing. Pavel Šilhavý, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „LDPC kódy“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Pavlovi Šilhavému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Cílem diplomové prácí je popsat problematiku protichybového zabezpečení pomocí LDPC (Low-Density Parity Check) kódů. Proto se kapitola 1 věnuje okrajově historii těchto kódů a posléze jsou popsány základní principy při vytváření LDCP kódu. S touto problematikou souvisí především definování pojmu jako paritní matice **H** nebo generující matice **G**, které jsou posléze používány při kódování a dekódování LDPC kódy.

Podle zadání je jedním z cílů práce zaměřit se na problematiku vytváření LDPC kódů. Toto téma je rozebíráno v kapitole 1.3, kde jsou podrobně přiblíženy přístupy při vytváření paritních matic **H**. Práce se zaměřuje především na strukturované metody vytváření těchto matic, tedy na metodu využívající Euklidovskou geometrii (EG) a metodu s použitím projektivní geometrie (PG). Součástí první kapitoly je také podkapitola 1.4 zabývající se grafickým ztvárněním LDPC kódu. Toto zobrazení se nazývá Tannerovi grafy a na příkladě je objasněna tvorba takového grafu.

V kapitole 2 jsou popsány metody kódování a dekódování LDPC kódů. Proces kódování je prezentován v podkapitole 2.1, ale pro tuto práci důležitější jsou dekódovací algoritmy LDPC kódů. Pro možnost porovnání různých přístupů byly vybrány čtyři metody:

- Hard-Decision algoritmus (HD) – neiterativní algoritmus.
- Bit-Flipping algoritmus (BF) – iterativní algoritmus.
- The Sum-Product algoritmus (SD) – iterativní algoritmus.
- Log Likelihood algoritmus (LD) – iterativní algoritmus.

V jednotlivých podkapitolách jsou všechny tyto přístupy podrobně popsány a u některých je přidán také příklad numerického výpočtu.

Další část práce tvoří kapitola 3, ve které je popsána algoritmizace vytvářecích a dekódovacích metod v prostředí Matlab. Tato algoritmizace je nutná pro vytvoření programu LDPC kódy, jenž je prioritním výstupem diplomové práce. Tento program slouží jako výuková pomůcka, tak jako analyzační nástroj pro porovnání jednotlivých dekódovacích metod. Popisem tohoto grafického interface se zabývá kapitola 3.2.

Poslední kapitola 4 popisuje realizaci zevrubného srovnání dosažených výsledků pro různé parametry kódů pomocí vytvořeného programu Simulace LDPC kódů. Zde jsou zobrazeny grafy, které popisují nejrůznější volby kódových parametrů a také časové srovnání vytvářecích a dekódovacích algoritmů. Na základě dosažených výsledků jsou popsány výstupy tohoto srovnání.

1 LDPC KÓDY

První kapitola se věnuje obecně LDPC kódům. Je zde nastíněna historie těchto kódů, konkrétně v podkapitole 1.1. V dalších částech úvodní kapitoly je pojednáváno o základních pojmech souvisejících s LDPC kódy a také je zde obecně popsáno vytváření paritní matice **H** a generující matice **G**, které jsou nepostradatelné při vytváření LDPC kódu.

Podkapitola 1.3 je stěžejní, protože popisuje vytváření paritní matice **H**. Jsou zde popsány základní přístupy k této problematice. Pro práci je dále důležitá především podkapitola 1.3.3, v níž jsou prezentovány dva základní přístupy při využití konečné geometrie: EG a PG. Na konkrétních příkladech je poté demonstrován postup při vytváření.

V poslední části kapitoly o LDPC kódech je pojednáváno o grafické reprezentaci těchto kódů – Tannerových grafech. Tyto grafy poslouží dále v práci při popisu dekódovacích metod.

1.1 Historie

LDPC kódy byly popsány roku 1962 Robertem Gallagerem v jeho dizertační práci [3]. Proto se v některé literatuře nazývají LDPC kódy jako Gallagerovi kódy, například v [14]. Tyto lineární blokové kódy byly na svou dobu výjimečné především tím, že se přibližovaly teoretickému Shannonovu výkonnostnímu limitu (kapacita kanálu) pro protichybové kódy. Dnes jsou LDPC kódy, spolu s Turbo kódy, považovány za jedny z nejlepších protichybových kódování. Navíc je velkou výhodou LDPC kódů, že nejsou patentově chráněny.

Bohužel i přes svůj potenciál nenacházely LDPC kódy ve své době uplatnění. Důvod byl prostý. Na svou dobu měla hardwarová realizace dekodéru velmi vysoké požadavky. Tehdy byla realizace iterativního dekódování možná pouze teoreticky. Proto upadly Gallagerovi kódy na dlouhých necelých dvacet let v zapomnění.

V roce 1981 Michael Tanner definoval grafickou interpretaci paritní matice **H**, o níž je pojednáváno dále v 1.4. Tannerova práce ovšem ještě nevedla zcela ke vzkříšení uoadlých LDPC kódů. To se povedlo až o dalších čtrnáct let později. Profesor univerzity v Cambridgi David MacKay se zasloužil svým výzkumem a silnou podporou těchto kódu o jejich znovuzrození [7]. MacKay dokázal, že LDPC kódy se přibližují Shannonovu kapacitnímu limitu exponenciálně s délkou kódu. Tato skutečnost přispěla k využívání LDPC kódů v několika standardech, jako například IEEE 802.16, IEEE 802.20, IEEE 802.3 nebo DBV-RS2 [10].

1.2 Popis LDPC kódů

LDPC kódy jsou obvykle navrhovány jako binární lineární blokové kódy. Proto je v následující kapitole 1.2.1 ve zkratce popsána konstrukce blokových kódů.

1.2.1 Blokové kódy

Pro účely LDPC kódu je zapotřebí definovat systematický lineární blokový kód $C_b\,(n,k)$, kde n je délka zakódovaná zprávy (délka kódu) a k je délka posloupnosti informačních bitů. Pro vytváření blokových kódů se využívá generující matice $\mathbf{G}$. Tuto matici lze definovat vztahem [11]

$$\mathbf{G} = [\mathbf{P}\ \mathbf{I_k}]\,, \tag{1.1}$$

kde $\mathbf{P}$ je paritní submatice velikosti $k \times (n-k)$ a $\mathbf{I_k}$ je jednotková submatice velikosti $k \times k$. Vzniklá generující matice má rozměry $k \times n$ a slouží ke kódování. Proces kódování popisuje vztah 1.2

$$\mathbf{c} = \mathbf{G}^T \circ \mathbf{k}, \tag{1.2}$$

kde vektor $\mathbf{k}$ chápeme jako posloupnost informačních znaků, které zakódujeme pomocí generující matice $\mathbf{G}$ do vektoru zakódované zprávy $\mathbf{c}$. Pro operaci kódování se používá operace modulo 2, jak je dále ukázáno v kapitole 2.1 zabývající se problematikou kódování LDPC kódů. Při aplikaci kódování podle vztahu 1.2 jsou vektory $\mathbf{c}$ a $\mathbf{k}$ chápány jako sloupcové, ale při obecném popisu nebo při práci s těmito proměnnými v programovém prostředí Matlab je uvažován řádkový vektor.

Dále pro blokové kódy definujeme kontrolní matici $\mathbf{H}$, která se většinou uvádí v systematickém tvaru. Tento tvar lze odvodit z generující matice $\mathbf{G}$ podle vztahu 1.3

$$\mathbf{H} = \left[\mathbf{I_{c-k}}\ \mathbf{P^T}\right]. \tag{1.3}$$

Mezi maticemi $\mathbf{G}$ a $\mathbf{H}$ musí platit ortogonalita, tj.

$$\mathbf{G} \circ \mathbf{H}^T = 0. \tag{1.4}$$

1.2.2 Konstrukce LDPC kódů

Jak již bylo naznačeno, LDPC kódy vycházejí z lineárních blokových kódů. Proto využívají generující matici $\mathbf{G}$ a kontrolní matici $\mathbf{H}$, která je častěji nazývána paritní matice.

Proces kódování je u LDPC kódů obdobný jako u blokových kódu, kde se tedy vychází ze vztahu 1.2. Zakódované slovo $\mathbf{c}$ dostaneme vynásobením (modulo 2) sloupců matice $\mathbf{G}$ s vektorem $\mathbf{k}$ reprezentujícím posloupnost informačních bitů.

Při konstrukci LDPC kódů se klade nejvyšší význam paritní matici **H**. Mezi zakódovaným slovem **c** a paritní maticí **H** musí platit vztah 1.5

$$\mathbf{H} \circ \mathbf{c} = 0. \tag{1.5}$$

Jak napovídá název LDPC (Low Density Parity Check) kódy jsou tvořeny na bázi tzv. „řídké“ matice **H**, tj. matice s většinovým zastoupením nulových prvků. Gallager ve své práci [3] formuloval LDPC kód jako $C_{LDPC}\,(n, o, v)$, kde n je délka kódu, o je počet jedniček na sloupec a v je počet jedniček na řádek paritní matice. Stanovil také podmínku, že by $o \geq 3$.

Podle rozložení jedniček v řídké matici **H** lze LDPC rozdělit do dvou skupin:

- Rovnoměrné (Regular) LDPC kódy,
- Nerovnoměrné (Irregular) LDPC kódy.

Tato práce se bude zabývat pouze skupinou rovnoměrných LDPC kódu, i když je dokázáno, že nerovnoměrné LDPC kódy dosahují lepší výsledky při porovnávání v rámci chybovosti BER (Bit Error Rate) než rovnoměrné LDPC kódy [11]. Výhoda rovnoměrných LDPC kódů spočívá především v jednodušší realizaci dekódovacích metod.

Vytváření paritních matic **H** je klíčové pro tvorbu LDPC kódu. Platí zde mnohá pravidla a také bylo definováno mnoho přístupů, jak dosáhnout co nejlepšího výsledku. Měřítkem kvality každého protichybového kódu je informační rychlost R [1]. Informační rychlost R je podle vztahu 1.6 definována jako poměr přenášených informačních znaků k k celkové délce kódu n

$$R = \frac{k}{n}. \tag{1.6}$$

Protože je konstrukce matic **H** rozsáhlým tématem a pro tuto práci velmi důležitou součástí, je této problematice věnována kapitola 1.3, kde je vše podrobně popsáno a vysvětleno na příkladech.

1.3 Tvorba paritních matic

V této sekci budou podrobně rozebrány přístupy k tvorbě paritní matice **H** LDPC kódů. Tato kapitola je členěna do menších celků, kde každý celek zastupuje danou metodu vytváření. Pro diplomovou práci je důležitá především podkapitola 1.3.3, jenž je věnována využití konečné geometrie při tvorbě paritních matic. Metody využívající konečné geometrie byly využity při tvorbě výukové i simulační aplikace v prostředí Matlab.

[1] v některé literatuře označována jako informační poměr.

1.3.1 Náhodná konstrukce paritní matice

Tvorba paritních matic postavených na náhodné konstrukci je nejjednodušší popisovanou variantou. Často nachází uplatnění při vytváření rozsáhlých matic $\mathbf{H}$, kdy musí každá takto vytvořená matice splňovat určité pravidla. Tyto pravidla jsou následující:

- každý řádek matice musí obsahovat konstantní počet jedniček,
- každý sloupec matice musí obsahovat konstantní počet jedniček,
- matice musí být řídká (musí obsahovat nízký počet jedniček v řádcích i sloupcích),
- libovolné dva řádky v matici musí být lineárně nezávislé,
- v matici nesmí docházet k cyklům délky 4 [2].

Jak je patrno z podmínek pro tvorbu paritních matic, je velice obtížné takovou matici sestavit ručně. Proto se tato metoda aplikuje především softwarově, například v prostředí Matlab.

Metoda návrhu náhodných paritním matic není příliš vhodná pro tvorbu matic o malých rozměrech. Zde často může dojít k nedodržení některé z podmínek, jenž musí být při konstrukci splněny (například není konstantní počet jedniček ve všech sloupcích). Naopak tato metoda nachází obrovské uplatnění při návrhu rozsáhlých paritním matic o rozměrech řádově stovek až tisíce sloupců.

1.3.2 Gallagerova paritní matice

Robert Gallager ve své práci [3] využíval paritní matici, kterou sestavil numericky bez použití výpočetní techniky. Tento přístup je odlišný od tvorby náhodných paritních matic, kdy předpokládá, že se matice skládá z několika submatic

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_\gamma \end{bmatrix},$$

kde γ značí váhu (počet jedniček) všech sloupců ve výsledné matici $\mathbf{H}$. Každý řádek submatice $\mathbf{H}_1 \ldots \mathbf{H}_\gamma$ má konstantní počet jedniček. Submatice $\mathbf{H}_1$ lze považovat za základní. Ostatní submatice vznikají různými permutacemi sloupců matice $\mathbf{H}_1$.

Paritní matice $\mathbf{H}$, kterou Gallager sestavil má rozměry 15×20. Skládá se ze tří submatic, kdy má každá rozměr 5×20 a obsahuje pouze jednu jedničku ve sloupci. Celková váha sloupců matice $\mathbf{H}$ je tedy $\gamma = 3$. Každý řádek submatice obsahuje přesně čtyři jedničky. Jak již bylo zmíněno výše, submatice $\mathbf{H}_2$ a $\mathbf{H}_3$ vnikly

[2] Vysvětleno dále v kapitole 1.4 zabývající se grafickým ztvárněním paritních matic.

různými permutacemi sloupců základní submatice $\mathbf{H}_1$. Výsledná matice $\mathbf{H}$ má poté následující tvar

$$\mathbf{H} = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.$$

Při této konstrukci matice vznikne lineární blokový kód $(20, 7)$ s minimální vzdáleností $d_{min} = 6$ [6, 10].

1.3.3 Geometrická konstrukce paritní matice

Metoda geometrické konstrukce paritních matic lze řadit do skupiny plně strukturovaných vytvářecích metord. Při této konstrukci paritních matic $\mathbf{H}$ je vycházeno z konečné geometrie [6]. Takto vytvořené matice splňují veškeré požadavky definované v 1.3.1. Při vytváření se vychází z Galoisových těles GF, jejichž přehled lze nalézt například v [1].

Při geometrické konstrukcu paritní matice rozlišujeme dva základní přístupy

- Euklidovská geometrie (EG),
- projektivní geometrie (PG).

Takto vznikají EG – LDPC kódy a PG – LDPC kódy. Obě varianty jsou popsány v následujícím textu. Pro každou variantu je uveden příklad, na kterém je přiblížen postup tvorby daného kódu. Metody založené na použití algebraické struktury konečného pole byly v práci využívány při tvorbě výukového i simulačního programu LDPC kódů popsaných v kapitole 3.2.

EG – LDPC kódy

Pomocí Euklidovské geometrie vznikají kódy $EG(m, 2^s)$, kde m označuje dimenzi kódu a proměnná s značí stupeň kódu. Čím vyšší je dimenze a stupeň, tím narůstá délka kódu a úměrně i složitost. Tvorba těchto kódů je podrobně popsána v [6]. V krátkosti lze shrnout, že EG – LDPC kódy vznikají pomocí m-dimenzionální EG nad $GF(2^s)$. EG pak obsahuje 2^{ms} bodů, kde každý bod je reprezentován m-ticemi nad $GF(2^s)$.

Pro účely tvorby LDPC kódů je důležitá skupina $m = 2$ dimenzionálních cyklických kódů. V tomto speciálním případě lze sestavit LDPC kód s následujícími parametry:

- délka kódu
$$n = 2^{2s} - 1, \tag{1.7}$$
- počet paritních bitů
$$n - k = 3^s - 1, \tag{1.8}$$
- počet informačních bitů
$$k = 2^{2s} - 3^s, \tag{1.9}$$
- minimální vzdálenost
$$d_{min} = 2^s + 1, \tag{1.10}$$
- citlivost (poměr jedniček vůči nulám v paritní matici $\mathbf{H}$)
$$r = \frac{2^s}{2^{2s} - 1}. \tag{1.11}$$

Složitost kódu vzrůstá exponenciálně s parametrem s. Narůstá především délka kódu, kde je patrné, že složitější kódy již nelze řešit numericky, ale je zde nutnost využít výpočetní techniky. Složitější kódy pak mají daleko vyšší výkon jak v množství opravitelných chyb, jenž vychází z d_{min}, tak například také v informační rychlosti. V tabulce 1.1 je uveden přehled EG – LDPC kódů až po $s = 7$, kdy jsou u každého kódu uvedeny parametry definované výše a přidány ještě další proměnně ρ a γ, jenž udávají váhu řádku, respektive sloupce v paritní matici.

V geometrii $EG(2, 2^s)$ se vytváří paritní matici $\mathbf{H}_{EG}$ velmi jednoduše. Stačí vytvořit vektor $\mathbf{v}$, jenž je poté cyklicky posouván tak, aby $2^{2s} - 2$ posunů vytvořilo stejný počet řádků. Výsledkem je $2^{2s} - 1$ nezávislých vektorů, které vytvoří čtvercovou matici $\mathbf{H}_{EG}$ o rozměrech $n \times n$. Váha takto vytvořených řádku ρ i sloupců γ je shodná

$$\rho = \gamma = 2^s.$$

Pro přiblížení tvorby paritní matice $\mathbf{H}_{EG}$ a generující matice $\mathbf{G}_{EG}$ je uveden následující příklad:

Tab. 1.1: Přehled EG – LDPC kódů.

s	n	k	d_{min}	ρ	γ	r	R
2	15	7	5	4	4	0,276	0,467
3	63	37	9	8	8	0,127	0,587
4	255	175	17	16	16	0,063	0,686
5	1023	781	33	32	32	0,031	0,763
6	4095	3367	65	64	64	0,017	0,822
7	16383	14197	129	128	128	0,008	0,867

1. **Volba kódu**
   Pro potřebu představení této metody tvorby paritní matice byl vybrán kód $EG\,(2,4)$. Tento kód odpovídá prvnímu řádku v tabulce 1.1, kde lze snadno nalézt veškeré potřebné informace k tomuto kódu.
2. **Vytvoření GF**
   Jak již bylo popsáno výše, EG – LDPC kódy vznikají pomocí m-dimenzionální EG nad $GF\,(2^s)$. EG pak obsahuje 2^{ms} bodů, kde každý bod je reprezentován m-ticemi nad $GF\,(2^s)$.
   Protože se jedná o $m = 2$ dimenzionální $EG\,(2, 2^2)$ nad $GF\,(2^2)$, vycházíme z pole
   $$GF\left(2^{2\cdot 2}\right) \rightarrow GF\,(16)\,.$$
   Toto GF je generováno primitivním polynomem $p(X) = 1 + X + X^4$, jak je například uvedeno v příloze A.1 literatury [1].
   Primitivní polynom $p(X)$ lze rozepsat na kořeny polynomu α [3], jak je předvedeno v tabulce 1.2, kdy je postupováno následovně
   $$p(X) = 1 + X + X^4,$$
   $$p(\alpha) = 1 + \alpha + \alpha^4,$$
   $$\alpha^4 = 1 + \alpha.$$
3. **Nalezení bodů ve vektoru v**
   Nyní přichází na řadu nalezení vytvářecího vektoru **v**. Tento vektor obsahuje kořeny polynomu α z $GF\,(2^{2\times 2})$. Vektoru **v** má délku shodnou s délkou kódu n a obsahuje čtyři body (podle váhy ρ). Tyto body nalezneme podle
   $$\left\{\alpha^{14} + \eta\alpha\right\}, \tag{1.12}$$

[3] V anglické literatuře označovány jako primitivní elementy α

Tab. 1.2: Možné reprezentace kořenu polynomu α.

Mocninná reprezentace	Polynomická reprezentace	Binární reprezentace
0	0	(0000)
1	1	(1000)
α	α	(0100)
α^2	α^2	(0010)
α^3	α^3	(0001)
α^4	$1+\alpha$	(1100)
α^5	$\alpha+\alpha^2$	(0110)
α^6	$\alpha^2+\alpha^3$	(0011)
α^7	$1+\alpha+\alpha^3$	(1101)
α^8	$1+\alpha^2$	(1010)
α^9	$\alpha+\alpha^3$	(0101)
α^{10}	$1+\alpha+\alpha^2$	(1110)
α^{11}	$\alpha+\alpha^2+\alpha^3$	(0111)
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	(1111)
α^{13}	$1+\alpha^2+\alpha^3$	(1011)
α^{14}	$1+\alpha^3$	(1001)

kde $\eta \in GF\left(2^2\right)$. Prvky z $GF\left(2^2\right)$ jsou značeny podle nenulového elementu β, který se volí z $GF\left(2^2\right)$ tak, aby platilo

$$\beta^{2^s-1}+1=0. \tag{1.13}$$

V tomto případě vyhovuje vztahu 1.13

$$\beta=\alpha^5.$$

Poté lze označit prvky $GF\left(2^2\right)$ jako

$$\left\{0,1,\beta,\beta^2\right\}$$

a po dosazení α jsou získány požadované hodnoty η, jenž budou dosazovány do vztahu 1.12

$$\left\{0,1,\alpha^5,\alpha^{10}\right\}.$$

V následujícím kroku jsou již pouze dosazeny do vztahu 1.12 nalezené prvky z $GF\left(2^2\right)$. Tento výpočet je zachycen v následujících rovnicích a je při něm vycházeno z tabulky 1.2.

$$\left\{\alpha^{14}+0\cdot\alpha\right\}=\{1001+0000\}=\{1001\}=\underline{\alpha^{14}}.$$

$$\left\{\alpha^{14} + 1 \cdot \alpha\right\} = \{1001 + 0100\} = \{1101\} = \underline{\alpha^{7}}.$$
$$\left\{\alpha^{14} + \alpha^{5} \cdot \alpha\right\} = \{1001 + 0011\} = \{1010\} = \underline{\alpha^{8}}.$$
$$\left\{\alpha^{14} + \alpha^{10} \cdot \alpha\right\} = \{1001 + 0111\} = \{1110\} = \underline{\alpha^{10}}.$$

Prvky vektoru $\mathbf{v}$ nabývají tyto hodnoty

$$v_i = \begin{cases} 1 & \text{jestliže } \alpha^i \text{ je nalezený bod,} \\ 0 & \text{ve zbylých případech.} \end{cases}$$

Nalezený vektor $\mathbf{v}$ má tedy tvar

$$\mathbf{v} = (000000011010001).$$

4. **Vytvoření $\mathbf{H}_{EG}$**

   Nyní již lze jednoduše vytvořit paritní matici $\mathbf{H}_{EG}$ postupným cyklickým posouváním (rotací) vektoru $\mathbf{v}$. Takto vznikne matice

$$\mathbf{H}_{EG} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

5. **Vytvoření generující matice $\mathbf{G}_{EG}$**

   Generující matice $\mathbf{G}_{EG}$ je opět generována cyklicky pomocí generujícího polynomu $\mathbf{g}_{EG}(X)$. Tento generující polynom se skládá z kořenů polynomu získaných z pole $GF(2^{2s})$. Poté je definováno nezáporné celé číslo h popsané vztahem 1.14

$$h = \delta_0 + \delta_1 2^s, \tag{1.14}$$

   kde $0 \leq \delta_i < 2^s$ pro $0 \leq i < 2$. Následně je definováno $w_{2^s}(h)$ jako suma koeficientů δ pro dané h

$$w_{2^s}(h) = \sum_{i=0}^{1} \delta_i. \tag{1.15}$$

Vzorec 1.16 udává podmínku, podle které jsou určeny kořeny polynomu, ze kterých se skládá hledaný generující polynom $\mathbf{g}_{EG}(X)$.

$$0 < \max_{0 \leq l < s} w_{2^s}\left(h^{(l)}\right) \leq (2^s - 1). \tag{1.16}$$

Po dosazení hodnot do podmínky ze vztahu 1.16 jsou získány následující primitivní elementy

$$\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^6, \alpha^8, \alpha^9, \alpha^{12}.$$

Nyní je důležité vyjádřit α z kořenů v $GF(2^{2s})$ generovaným polynomem $p(X) = 1 + X + X^4$. Tyto kořeny zachycuje tabulka 1.3. Získaným kořenům z 1.16 jsou přiřazeny minimální polynomy z tabulky 1.3 tak, aby vznikl generující polynom $\mathbf{g}(X)$

Tab. 1.3: Minimální polynomy z kořenů v $GF(2^{2s})$.

Kořen polynomu	Minimální polynom
0	X
1	$X + 1$
$\alpha, \alpha^2, \alpha^4, \alpha^8$	$X^4 + X + 1$
$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$X^4 + X^3 + X^2 + X + 1$
α^5, α^{10}	$X^2 + X + 1$
$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$	$X^4 + X^3 + 1$

$$\begin{aligned}\mathbf{g}_{EG}(X) &= \left(1 + X + X^4\right)\left(1 + X + X^2 + X^3 + X^4\right) \\ &= 1 + X^4 + X^6 + X^7 + X^8.\end{aligned}$$

Obdobně jako při tvorbě $\mathbf{H}_{EG}$ i zde vznikne generující matice $\mathbf{G}_{EG}$ cyklickou rotací generujícího vektoru $\mathbf{g}_{EG}(X)$. Počet posuvů je ovšem roven číslu $k - 1$, aby počet řádků generující matice odpovídal počtu informačních znaků k. Aby vygenerovaná matice $\mathbf{G}_{EG}$ vyhovovala podmínce systematického blokového kódu, musí být převedena na tvar definovaný vztahem 1.1. Zde je pouze prohozena paritní submatice $\mathbf{P}$ s jednotkovou submaticí $\mathbf{I}$, což nemá na výslednou funkci významný vliv. Generující matice má poté následující tvar

$$\mathbf{G}_{EG} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

PG – LDPC kódy

Druhou variantou geometrické konstrukce pomocí níž lze vytvořit paritní matice **H** LDPC kódu se nazývá projektivní geometrie. Pomocí PG vznikají kódy $PG\,(m, 2^s)$, kdy pro tvorbu PG – LDPC kódů má největší význam varianta s $m = 2$. Postup vytváření paritní matice $\mathbf{H}_{PG}$ je v mnohém podobný s tvorbou v EG – LDPC. I zde se paritní matice získává postupným cyklickým posouváním vektoru **v**. Kódy s dimenzí $m = 2$ jsou $PG\,(2, 2^s)$ kódy s následujícími parametry:

- délka kódu
$$n = 2^{2s} + 2^s + 1, \tag{1.17}$$
- počet paritních bitů
$$n - k = 3^s + 1, \tag{1.18}$$
- počet informačních bitů
$$k = 2^{2s} + 2^s - 3^s, \tag{1.19}$$
- minimální vzdálenost
$$d_{min} = 2^s + 2, \tag{1.20}$$
- citlivost
$$r = \frac{2^s + 1}{2^{2s} + 2^s + 1}. \tag{1.21}$$

Složitost kódu také zde vzrůstá exponenciálně s parametrem s. Se zvyšující se složitostí $PG\,(2, 2^s)$ kódu pak lze získat daleko vyšší výkon jak v množství opravitelných chyb, tak v informační rychlosti. V tabulce 1.4 je uveden přehled PG – LDPC kódů až po $s = 7$. Z tohoto přehledu jsou zřejmé nárůsty výkonnostních parametrů protichybového kódu se zvyšující se délkou kódu.

Tab. 1.4: Přehled PG – LDPC kódů.

s	n	k	d_{min}	ρ	γ	r	R
2	21	11	6	5	5	0,238	0,524
3	73	45	10	9	9	0,123	0,616
4	273	191	18	17	17	0,062	0,7
5	1057	813	34	33	33	0,031	0,769
6	4161	3431	66	65	65	0,017	0,825
7	16513	14326	130	129	129	0,008	0,868

Varianta PG s dimenzí $m = 2$ se nevolí náhodně. Stejně jako v případě EG vzniká paritní matice $\mathbf{H}_{PG}$ cyklickým posouváním vektoru **v**. Tímto způsobem je získána čtvercová paritní matice o rozměrech $n \times n$, kde je uvažováno n z rovnice 1.17. Délka vektoru **v** tedy odpovídá délce vytvořeného PG – LDPC kódu. Matice

$\mathbf{H}_{PG}$ má konstantní počet jedniček jak v řádcích, tak i ve sloupcích, jak je patrno z tabulky 1.4.

Pro ilustraci postupu vytváření paritní matice $\mathbf{H}_{PG}$ a generující matice $\mathbf{G}_{PG}$ je i zde prezentován příklad:

1. **Volba kódu**
   Pro prezentaci vytváření paritní matice $\mathbf{H}_{PG}$ a generující matice $\mathbf{G}_{PG}$ byl vybrán kód $PG\,(2,4)$. Tento kód odpovídá prvnímu řádku v tabulce 1.4, kde jsou zachyceny veškeré potřebné parametry k tomuto kódu.
2. **Vytvoření GF**
   PG – LDPC kódy vznikají podobně EG – LDPC kódy z elementů GF. Zde je využíváno pole $GF\,(2^{3s})$ obsahující $GF\,(2^s)$ jako podpole.

   $$GF\left(2^{3\times 2}\right) \rightarrow GF\,(64)\,.$$

   Použité $GF\,(64)$ je generováno primitivním polynomem $p(X) = 1 + X + X^6$. Primitivní polynom $p(X)$ lze rozepsat na kořeny polynomu α, kdy je postupováno následovně

   $$p(X) = 1 + X + X^6,$$
   $$p(\alpha) = 1 + \alpha + \alpha^6,$$
   $$\alpha^6 = 1 + \alpha.$$

   Kompletní tabulku rozepsaných hodnot lze nalézt v literatuře [6]. Tabulka je podobná tabulce 1.2. Zde je ovšem použita 6ti bitová reprezentace mocnin α.
3. **Nalezení bodů ve vektoru v**
   V případě PG – LDPC kódů se mírně liší způsob získávání vektoru $\mathbf{v}$, pomocí kterého je následně vytvářena paritní matice $\mathbf{H}$. Protože vektor $\mathbf{v}$ musí mít stejnou délku jako délka kódu n, definujeme maximální hodnotu kořene polynomu α jako α^{n-1}, kde n je definováno vztahem 1.17. Geometrie $PG\,(2,2^2)$ tedy obsahuje 21 bodů:

   $$(\alpha^0)\,,(\alpha^1)\,,(\alpha^2)\,,(\alpha^3)\,,(\alpha^4)\,,(\alpha^5)\,,(\alpha^6)\,,$$
   $$(\alpha^7)\,,(\alpha^8)\,,(\alpha^9)\,,(\alpha^{10})\,,(\alpha^{11})\,,(\alpha^{12})\,,(\alpha^{13})\,,$$
   $$(\alpha^{14})\,,(\alpha^{15})\,,(\alpha^{16})\,,(\alpha^{17})\,,(\alpha^{18})\,,(\alpha^{19})\,,(\alpha^{20})\,.$$

   Z těchto bodů musí být vybráno pět bodů, které ve vektoru $\mathbf{v}$ budou mít hodnotu jedna. Při hledání těchto bodů je vycházeno z bodů (α) a (α^{20}), které jsou dosazeny do vzorce

   $$\left\{\eta_1\alpha + \eta_2\alpha^{20}\right\}, \tag{1.22}$$

   kde η_1 i η_2 jsou brány z

   $$GF\left(2^2\right) = \left\{0, 1, \beta, \beta^2\right\}.$$

Stejně jako v případě EG – LDPC kódů je i zde zapotřebí dodatečně definovat koeficient β, který je v případě PG – LDPC roven

$$\beta = \alpha^n,$$

tedy pro tento příklad je $\beta = \alpha^{21}$. Volba koeficientu β je dalším rozdílem mezi dvěma popisovanými směry geometrické konstrukce paritních matic.
V následujícím kroku je ještě třeba vybrat pět kombinací koeficientů η_1, η_2, jenž vyhovují následujícím podmínkám

- $\eta_1 = \eta_2 \neq 0$,
- $(\eta_1, \eta_2) \neq (k\eta_1, k\eta_2)$, kde $k \in GF\,(2^2)$.

Kombinace koeficientů η_1, η_2 splňující výše definované podmínky a byly vybrány k řešení tohoto příkladu jsou prezentovány v tabulce 1.5.

Tab. 1.5: Kombinace parametrů η_1, η_2.

Parametr η_1	Parametr η_2
1	0
0	1
1	1
1	β
1	β^2

V dalším kroku jsou dosazeny nalezené parametry η_1, η_2 do vztahu 1.22 a vzniknou tak rovnice jejichž výpočtem jsou získány požadované body, které značí polohu jedniček ve vektoru $\mathbf{v}$. Pro ukázku je zde zachycen výpočet těchto rovnic.

$$\left\{1 \cdot \alpha + 0 \cdot \alpha^{20}\right\} = \{010000 + 000000\} = \{010000\} = \underline{\alpha}.$$

$$\left\{0 \cdot \alpha + 1 \cdot \alpha^{20}\right\} = \{000000 + 001111\} = \{001111\} = \underline{\alpha^{20}}.$$

$$\left\{1 \cdot \alpha + 1 \cdot \alpha^{20}\right\} = \{010000 + 001111\} = \{011111\} = \underline{\alpha^{57}}.$$

$$\left\{1 \cdot \alpha + \beta \cdot \alpha^{20}\right\} = \{010000 + 101110\} = \{111110\} = \underline{\alpha^{56}}.$$

$$\left\{1 \cdot \alpha + \beta^2 \cdot \alpha^{20}\right\} = \{010000 + 100001\} = \{110001\} = \underline{\alpha^{11}}.$$

Protože body tvořící geometrie $PG\,(2, 2^2)$ mají maximální index kořene (α^{20}), musí být nalezené body zkráceny o násobky β. Tak je dosaženo indexů s polohou jedniček ve vektoru $\mathbf{v}$.

$$\left\{(\alpha), \left(\alpha^{11}\right), \left(\alpha^{14}\right), \left(\alpha^{15}\right), \left(\alpha^{20}\right)\right\}.$$

Hledaný vektor $\mathbf{v}$ má potom tvar

$$\mathbf{v} = (01000000001001100001)\,.$$

4. **Vytvoření $\mathbf{H}_{EG}$**
   Paritní matice $\mathbf{H}_{EG}$ se tvoří obdobně jako v případě EG – LDPC, tedy postupným cyklickým posouváním vektoru $\mathbf{v}$. Tímto postupem je dosažena požadovaná paritní matice, která má opět čtvercový tvar

$$\mathbf{H}_{PG} = \begin{bmatrix}
0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&1\\
1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0\\
0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0\\
0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1\\
1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1\\
1&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&1&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&1&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1\\
1&0&0&1&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&1&0&0&1&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&1&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&1&0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&1&1&0&0&0&0&1&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&1&1&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&1&0&1&0&0&0\\
0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&1&0&1\\
1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&1&0
\end{bmatrix}$$

5. **Vytvoření generující matice $\mathbf{G}_{EG}$**
   Generující matice $\mathbf{G}_{PG}$ je také generována cyklicky pomocí generujícího polynomu $\mathbf{g}_{PG}(X)$. Protože byl postup vytvoření generujícího polynomu popsán v předchozím příkladě při tvorbě EG – LDPC kódu, je zde prezentována pouze jeho hodnota

$$\mathbf{g}_{PG}(X) = 1 + X^2 + X^4 + X^6 + X^7 + X^{10}.$$

   Stejně jako když byla tvořena paritní matice $\mathbf{H}_{PG}$, tak vznikne generující matice $\mathbf{G}_{PG}$ cyklickou rotací generujícího vektoru $\mathbf{g}_{PG}(X)$. Počet posuvů generujícího vektoru je roven číslu $k-1$, aby počet řádků generující matice odpovídal

počtu informačních znaků k. Pro docílení požadované generující matice geometrie PG vyhovující podmínce systematického blokového kódů je proveden převod do systematické tvaru. Finální tvar generující matice má tedy podobu

$$\mathbf{G}_{PG} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

U uvedené generující matice $\mathbf{G}_{PG}$ je opět otočena paritní a jednotková submatice vůči definici ze vztahu 1.1. To má vliv pouze na pozici posloupnosti informačních bitů $\mathbf{k}$, která bude umístěna na začátku slova $\mathbf{c}$ po provedení operace kódování.

1.4 Tannerovi grafy

Lineární blokové kódy mohou být kromě běžného maticového zobrazení reprezentovány také graficky. Existuje několik variant grafického zobrazení kódů. Jednou z nejznámějších grafických reprezentací je trellis diagram, na které je založen také stejnojmenný dekódovací algoritmus. Další, často využívanou grafickou reprezentací lineárních blokových kódů je Tannerův graf, který zobrazuje vztahy mezi kódovými bity a bity paritních součtů. Pro LDPC kódy je ve většině případů výhradě využívána grafická reprezentace pomocí Tannerova grafu, jenž je pojmenován podle Michaela Tannera, který tento typ grafu poprvé v roce 1981 prezentoval.

Tannerův graf je vlastně graficky ztvárněná paritní matice $\mathbf{H}$. V tomto zobrazení jsou rozlišeny dvě sady uzlů [4]

- bitové uzly c_i,
- součtové uzly s_j.

Počet bitových uzlů je roven délce kódu n a množství součtových uzlů se odvíjí od počtu řádků v paritní matici, která je převáděna do grafické podoby. Hrany grafu jsou vytvářeny pouze mezi jednotlivými skupinami uzlů a to pouze v případě, že

[4] Označení uzlů se v různých literaturách liší.

příslušná pozice v paritní matice je rovna logické jedničce

$$\mathbf{H}(i, j) = 1,$$

kde i značí pozici uzlu c_i a j pozici uzlu s_j.

Tannerovi grafy se vytvářejí především pro to, aby bylo možné lépe popsat dekódovací algoritmy, jenž jsou obsahem následující kapitoly 2. Tyto grafy slouží také pro kontrolu posledního pravidla formulovaného v kapitole 1.3.1, jenž říká, že nesmí docházet k cyklům délky 4 uvnitř paritní matice. Toto pravidle se dá přeformulovat tak, že cesta z libovolného uzlu c_i grafem zpět do výchozího (startovního) uzlu c_i v Tannerově grafu, nesmí být délky 4. Tj. počet hran, po kterých je grafem procházeno při hledání cesty do startovního uzlu nesmí být roven 4. Při správné konstrukci paritní matice $\mathbf{H}$ bývá cesta obvykle 6 hran dlouhá.

Jako ukázka tvorby Tannerova grafu je zde přiložen obrázek B.1, na kterém je zachycena grafická podoba paritní matice $\mathbf{H}_{EG}$ sestrojená v příkladu v kapitole 1.3.3. Tučnou červenou čarou je znázorněn cyklus délky 6, který je pro tuto vytvářecí matici charakteristický. Jak je z obrázku Tannerova grafu viditelné, je jeho zobrazení dosti nepřehledné, a proto se pro rozsáhlejší paritní matice nevyužívá. Toto zobrazení má čistě demonstrační úlohu pro přiblížení některých dekódovací algoritmů LDPC kódů, popsaných v kapitole 2.2.

2 KÓDOVÁNÍ A DEKÓDOVÁNÍ LDPC KÓDŮ

Následující kapitola se zabývá problematikou kódování a dekódování LDPC kódů. Nejprve je v podkapitole 2.1 v krátkosti vysvětlen princip kódování pomocí generující matice **G** a posléze následuje pro práci velice významná podkapitola 2.2, ve které jsou prezentovány celkem čtyři dekódovací algoritmy. Z tohoto počtu jsou tři algoritmy založeny na iterativním dekódování a jeden je pouze jednokrokový dekódovací algoritmus.

2.1 Kódování LDPC kódů

Protože LDPC kódy patří do skupiny lineárních blokových kódů, využívá se při kódování generující matice **G**. Tuto generující matici získáváme z paritní matice **H**, jak bylo ukázáno na příkladech v podkapitole 1.3 zabývající se tvorbou paritních matice. Mezi těmito maticemi musí platit vztah 1.4.

Pro proces kódování pak musí být zvolen vektor **k**, reprezentující vstupní posloupnost informačních bitů. Jako výstup kódování je chápán vektor **c** symbolizující výstup z kodéru LDPC kódu, jenž je připraven k přenosu přes informační kanál. Před vlastním přenosem bývá bitový proud ještě modulován.

Po definování všech potřebných parametrů, lze provést kódování. To je realizováno vektorovým vynásobením sloupců generující matice **G** se vstupní posloupnosti informačních znaků **k** a následným sečtením jednotlivých součinů za pomoci logické funkce XOR. Daná operace je v práci označována jako součin modulo 2.

Proces kódování je předveden na následujícím příkladě.

1. **Volba generující matice G**
   První krok při procesu kódování vychází z volby paritní matice **H**. Metody vytváření této matice jsou shrnuty v kapitole 1.3. Pro účel demonstrace funkce kódování LDPC kódu byla zvolena generující matice $\mathbf{G}_{EG}$ formulovaná v sekci 1.3.3.

$$\mathbf{G}_{EG} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

2. **Zadání vstupního vektoru k**
   Binární vektor **k** reprezentuje přenášenou informaci. Tato informace je z důvodu zabezpečení rozšířena o kontrolní bity na celkovou délku n. Maximální

délka vektoru $\mathbf{k}$ je rovna počtu řádků generující matice. V tomto příkladě byla zvolena hodnota informačních bitů náhodně

$$\mathbf{k} = 1001111.$$

3. **Kódování**
   Kódování probíhá vynásobením vstupního vektoru $\mathbf{k}$ se sloupci matice $\mathbf{G}_{EG}$. Hodnota výstupního bitu vektoru $\mathbf{c}$ je pak získána sečtením modulo 2 jednotlivých součinů

$$\begin{aligned}
\mathbf{c}(1) = \mathbf{k} \circ \mathbf{G}_{EG}(:,1) &= (1\cdot 1) \oplus (0\cdot 0) \oplus (0\cdot 0) \oplus (1\cdot 0) \oplus (1\cdot 0) \oplus (1\cdot 0) \oplus \\
&\oplus (1\cdot 0) = \underline{1} \\
\mathbf{c}(2) = \mathbf{k} \circ \mathbf{G}_{EG}(:,2) &= (1\cdot 0) \oplus (0\cdot 1) \oplus (0\cdot 0) \oplus (1\cdot 0) \oplus (1\cdot 0) \oplus (1\cdot 0) \oplus \\
&\oplus (1\cdot 0) = \underline{0} \\
&\vdots \\
\mathbf{c}(15) = \mathbf{k} \circ \mathbf{G}_{EG}(:,15) &= (1\cdot 0) \oplus (0\cdot 0) \oplus (0\cdot 0) \oplus (1\cdot 0) \oplus (1\cdot 0) \oplus (1\cdot 0) \oplus \\
&\oplus (1\cdot 1) = \underline{1}
\end{aligned}$$

$$\mathbf{c} = 100101000011101.$$

2.2 Dekódování LDPC kódů

V textu následující kapitoly jsou prezentovány různé metody dekódování LDPC kódů. Při popisu dekódovacích algoritmů je využívána grafická realizace paritní matice $\mathbf{H}$ pomocí Tannerových grafů, jenž byly obsahem kapitoly 1.4. Náplní práce jsou tyto dekódovacími algoritmy

- Hard-Decision algoritmus (HD) – neiterativní algoritmus,
- Bit-Flipping algoritmus (BF) – iterativní algoritmus,
- The Sum-Product algoritmus (SD) – iterativní algoritmus,
- Log Likelihood algoritmus (LD) – iterativní algoritmus.

Následující řádky vytváří teoretický základ k těmto algoritmům, kdy je popsána matematická rutina. K algoritmům HD a SD je navíc vytvořen příklad osvětlující postup při dekódování.

2.2.1 Hard-Decision algoritmus

Hard Decision (HD) je dekódovací algoritmus založený na výměně informací mezi bitovými a součtovými uzly v Tannerově grafu [5]. Algoritmus tvrdého rozhodnutí

není iterativním algoritmem, a proto je nejjednodušším popisovaným algoritmem pro dekódování LDPC kódů.

Postup dekódování je následující

- V prvním kroku odešlou všechny bitové uzly c_i informaci o přijatém zakódovaném slovu, jenž je obsaženo ve vektoru $\mathbf{c}$, součtovým uzlům s_j.
- V druhém kroku každý součtový uzel s_j vypočítá paritu pro každý připojený bitový uzel tak, že sečte modulo 2 všechny přijaté bity kromě aktuálního bitu, pro který je počítána parita. Vypočítanou paritu pošlou uzly s_j zpět uzlům c_i.
- Na závěr je pomocí majoritní logiky rozhodnuto o hodnotě příslušného bitu vektoru $\mathbf{c}$. Bity, ze kterých je rozhodováno o nové hodnotě, se získávají z obdržených zpráv od uzlů s_j. Navíc je k nim přidána původní hodnota bitu v obdržené zprávě $\mathbf{c}$.

Pomocí HD dekódovacího algoritmu je možno opravit pouze omezený počet chyb. Tento počet vychází z teorie o blokových kódů a je definován vztahem 2.1 [12]

$$t = \frac{d_{min} - 1}{2}. \tag{2.1}$$

Výhodou tohoto algoritmu může být skutečnost, že lze proces dekódování přerušit již v druhém kroku popsaného postupu, a to v případě, kdy vyjdou všechny paritní součty ekvivalentně. Tímto je jisté, že do zakódovaného slova $\mathbf{c}$ nebyla vlivem přenosu přes informační kanál vnořena žádná chyba a přenos byl tím pádem bezchybný.

Pro názornou ukázku funkce HD algoritmu je určen následující příklad.

1. **Volba LDPC kódu**

   V tomto příkladu je vycházeno z EG – LDPC kódu s paritní a generující maticí z kapitoly 1.3.3. Paritní matice má rozměr $n \times n$, tj. 15×15 a její grafické ztvárnění je zachyceno na obrázku B.1. Jako výstup kódování je použito slovo

   $$\mathbf{c} = 100101000011101$$

   z příkladu na kódováni LDPC kódu, uvedeného v kapitole 2.1.

2. **Přenos přes informační kanál**

   Do slova $\mathbf{c}$ jsou vlivem přenosu přes informační kanál náhodně vneseny $t = 2$ chyby, které je tento kód dle vztahu 2.1 schopen opravit. Tyto chyby mohou být způsobeny například působením AWGN šumu nebo jiného libovolného rušení působícího v čase přenosu na informační kanál. Tyto chyby jsou umístěny na pozice 5 a 8 v kódovém slově

   $$\mathbf{c}_{chyb} = 1001\mathbf{1}110\mathbf{1}0011101.$$

3. **Vertikální krok**

   Vertikální krok HD algoritmu je realizován tak, že bitové uzly c_i odešlou zprávy

o obdržených bitech součtovým uzlům s_j. Tento proces je znázorněno v tabulce 2.1. Index bitového uzlu c_i odpovídá indexování ve vektoru $\mathbf{c}_{chyb}$ vstupujícím do dekodéru.

Tab. 2.1: Vertikální krok v HD algoritmu.

Součtový uzel	Přijaté bity
s_1	$c_8 \to 1 \quad c_9 \to 0 \quad c_{11} \to 1 \quad c_{15} \to 1$
s_2	$c_1 \to 1 \quad c_9 \to 0 \quad c_{10} \to 0 \quad c_{12} \to 1$
s_3	$c_2 \to 0 \quad c_{10} \to 0 \quad c_{11} \to 1 \quad c_{13} \to 1$
s_4	$c_3 \to 0 \quad c_{11} \to 1 \quad c_{12} \to 1 \quad c_{14} \to 0$
s_5	$c_4 \to 1 \quad c_{12} \to 1 \quad c_{13} \to 1 \quad c_{15} \to 1$
s_6	$c_1 \to 1 \quad c_5 \to 1 \quad c_{13} \to 1 \quad c_{14} \to 0$
s_7	$c_2 \to 0 \quad c_6 \to 1 \quad c_{14} \to 0 \quad c_{15} \to 1$
s_8	$c_1 \to 1 \quad c_3 \to 0 \quad c_7 \to 0 \quad c_{15} \to 1$
s_9	$c_1 \to 1 \quad c_2 \to 0 \quad c_4 \to 1 \quad c_8 \to 1$
s_{10}	$c_2 \to 0 \quad c_3 \to 0 \quad c_5 \to 1 \quad c_9 \to 0$
s_{11}	$c_3 \to 0 \quad c_4 \to 1 \quad c_6 \to 1 \quad c_{10} \to 0$
s_{12}	$c_4 \to 1 \quad c_5 \to 1 \quad c_7 \to 0 \quad c_{11} \to 1$
s_{13}	$c_5 \to 1 \quad c_6 \to 1 \quad c_8 \to 1 \quad c_{12} \to 1$
s_{14}	$c_6 \to 1 \quad c_7 \to 0 \quad c_9 \to 0 \quad c_{13} \to 1$
s_{15}	$c_7 \to 0 \quad c_8 \to 1 \quad c_{10} \to 0 \quad c_{14} \to 0$

4. **Horizontální krok**
   Následujícím krokem je realizace horizontálního kroku HD algoritmu. Při jeho vykonávání je postupováno tak, že se ve všech součtových uzlech vypočítají parity podle druhého kroku výše uvedeného postupu dekódování metodou HD. Tyto výsledky jsou odeslány zpět bitovým uzlům, ve kterých je pomocí majoritní logiky rozhodnuto o hodnotě daného dekódovaného bitu zprávy $\mathbf{d}$. Vektor $\mathbf{d}$ je poté výstupem celého procesu dekódování. Proces horizontálního kroku je zachycen v tabulce 2.2, kde jsou přehledně zapsány zprávy přenášené od součtových uzlů. Z těchto hodnot lze při takto krátkém kódu pomocí oka určit hodnotu dekódovaného bitu. Tato hodnota je zaznamenána v posledním sloupci popisované tabulky.
5. **Výsledek dekódování HD algoritmu**
   V posledním kroku je zkontrolována správnost dekódování tak, že je porovnáno zakódované slovo $\mathbf{c}$ s dekódovaným slovem $\mathbf{d}$. V tomto případě se oba vektory rovnají, a proto lze dekódování prohlásit za úspěšné

$$\mathbf{c} = \mathbf{d} = 100101000011101.$$

Tab. 2.2: Horizontální krok HD algoritmu.

Bitový uzel	$\mathbf{c}_{chyb}$	Zprávy od součtových uzlů	$\mathbf{d}$
c_1	1	$s_2 \to 1 \quad s_6 \to 0 \quad s_8 \to 1 \quad s_9 \to 0$	1
c_2	0	$s_3 \to 0 \quad s_7 \to 0 \quad s_9 \to 1 \quad s_{10} \to 1$	0
c_3	0	$s_4 \to 0 \quad s_8 \to 0 \quad s_{10} \to 1 \quad s_{11} \to 0$	0
c_4	1	$s_5 \to 1 \quad s_9 \to 0 \quad s_{11} \to 1 \quad s_{12} \to 0$	1
c_5	1	$s_6 \to 0 \quad s_{10} \to 0 \quad s_{12} \to 0 \quad s_{13} \to 1$	0
c_6	1	$s_7 \to 1 \quad s_{11} \to 1 \quad s_{13} \to 1 \quad s_{14} \to 1$	1
c_7	0	$s_8 \to 0 \quad s_{12} \to 1 \quad s_{14} \to 0 \quad s_{15} \to 1$	0
c_8	1	$s_1 \to 0 \quad s_9 \to 0 \quad s_{13} \to 1 \quad s_{15} \to 0$	0
c_9	0	$s_1 \to 1 \quad s_2 \to 0 \quad s_{10} \to 1 \quad s_{14} \to 0$	0
c_{10}	0	$s_2 \to 0 \quad s_3 \to 0 \quad s_{11} \to 0 \quad s_{15} \to 1$	0
c_{11}	1	$s_1 \to 0 \quad s_3 \to 1 \quad s_4 \to 1 \quad s_{12} \to 0$	1
c_{12}	1	$s_2 \to 1 \quad s_4 \to 1 \quad s_5 \to 1 \quad s_{13} \to 0$	1
c_{13}	1	$s_3 \to 1 \quad s_5 \to 1 \quad s_6 \to 0 \quad s_{14} \to 1$	1
c_{14}	0	$s_4 \to 0 \quad s_6 \to 1 \quad s_7 \to 0 \quad s_{15} \to 1$	0
c_{15}	1	$s_1 \to 0 \quad s_5 \to 1 \quad s_7 \to 1 \quad s_8 \to 1$	1

Na předchozím příkladě byla numericky demonstrována korekční schopnost HD algoritmu, který ovšem nedosahuje výkonu iteračních dekódovacích algoritmů jak bude ukázáno v kapitole 4 zabývající se mimo jiné také komparací dekódovacích algoritmů.

2.2.2 Bit-Flipping algoritmus

Dekódovací algoritmus Bit-Flipping (BF) byl vyvinut Gallagerem stejně jako celé LDPC kódy. Algoritmus se řadí do skupiny iterativních dekódovacích algoritmů a je založen na výpočtu vektoru syndromu $\mathbf{s}$ v paritní matici $\mathbf{H}$. Při výpočtu syndromu se vychází ze vztahu 1.5, jenž říká, že vynásobením modulo 2 zakódovaného slova s paritní matici musí být výsledek roven 0. Toto není splněno, pokud je do zakódovaného slova vložen chybový element. Výpočet jednotlivých bitů syndromu je realizován podle následujícího vztahu

$$\mathbf{s} = \mathbf{c} \circ \mathbf{H}^T. \tag{2.2}$$

Po získání syndromu matice $\mathbf{H}$ může být dekódování ukončeno v případě, kdy je vektor $\mathbf{s}$ nulový. V opačném případě, když je vektor nenulový, se využije k násobení se sloupci paritní matice podle vztahu 2.3

$$S_i = \mathbf{s} \circ \mathbf{H}, \tag{2.3}$$

kde $i = 1, 2, \ldots, n$. Takto vypočítané hodnoty jsou posléze porovnávány. Jestliže nastane situace, kdy je hodnota na pozici

$$S_{i+1} > S_i,$$

je pozice $i+1$ uložena, aby mohla být negována hodnota bitu na odpovídající pozici ve vektoru **c**. V posledním kroku je přepočítán syndrom a pokud je opět nenulový, celý cyklus se opakuje. Počet opakování bývá povinným parametrem algoritmu.

Celý postup dekódování pomocí BF algoritmu je shrnut v následujícím postupu.

1. Výpočet bitů syndromu podle vztahu 2.2. Jestliže je vektor syndromu nulový, je dekódování přerušeno.
2. Nalezení hodnot vektoru **S** dle 2.3.
3. Porovnávání hodnot ve vektoru **S** a uložení pozic, kde je hodnota na pozici $S_{i+1} > S_i$.
4. Přehození hodnot vektoru **c** na definovaných pozicích.
5. Opakování bodů celého cyklu dokud není vypočítaný syndrom z bodu 1 nulový nebo není dosaženo maximálního počtu iterací.

Dekódovací algoritmus BF je díky nízkému počtu vykonávaných operací na jednu iteraci velice rychlým algoritmem. Jeho rychlost je vykoupena sníženým výkonem v korekci chyb, kdy je ovšem výhodné využít vyššího počtu iterací. Vliv počtu iterací na BER charakteristiky algoritmu BF je zkoumán v kapitole 4.2.2.

2.2.3 The Sum-Product algoritmus

The Sum-Product algoritmus (SD) je založen na měkkém rozhodnutí. Algoritmus je postaven na podobném principu jako HD algoritmus, popisovaný v podkapitole 2.2.1, s tím rozdílem, že nepracuje s bity ale s pravděpodobnostmi. Tyto pravděpodobnosti vyjadřují šanci, že zkoumaná hodnota bitu bude log. 1 nebo 0. Algoritmus je iterativní a je chápán jako základní algoritmus pro dekódování LDPC kódů.

V práci není rozebírána základní verze SD algoritmu popsaná v [11], ale je využívána zjednodušená verze SD algoritmu. Tato verze sebou nese řadu zjednodušení, které se pozitivně projeví ve finálním výpočetním výkonu algoritmu. Popis SD dekódovacího algoritmu lze opět efektivně provést podle grafické reprezentace použité paritní matice, tedy Tannerova grafu.

SD algoritmus ve svém základu předpokládá využití přenosu přes AWGN kanál. Tím pádem je nutno výstup z kodéru před vlastním přenosem ještě modulovat, kdy je za tímto účelem v práci využita modulace BPSK. Modulované data jsou poté přenášena přes AWGN kanál se standardním rozptylem $\sigma = 0{,}8$.

V prvním kroku vlastního dekódovaní je pak stanovena dvojice pravděpodobností značící šanci, že obdržený bit má hodnotu log. 1 nebo 0. Pravděpodobnost výskytu

log. 1 je definována vztahem 2.4

$$f_j^1 = \frac{1}{1 + e^{-\frac{2Arx_j}{\sigma^2}}} \tag{2.4}$$

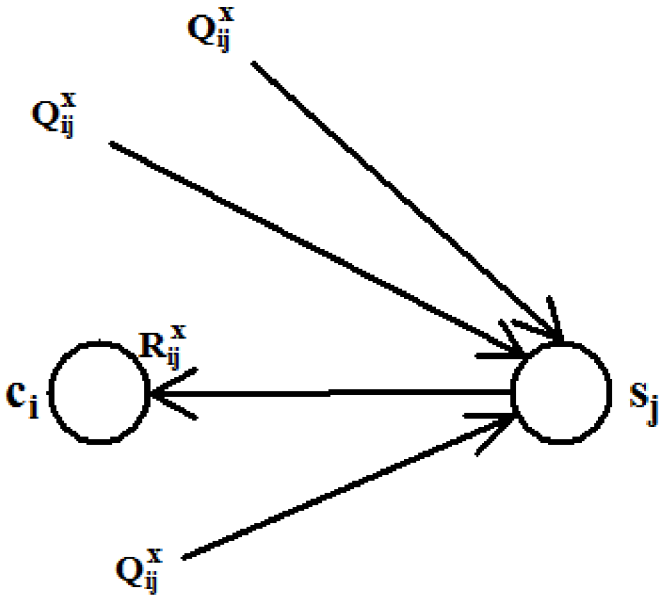
a pravděpodobnost výskytu log. 0 je snadno odvozena podle 2.5

$$f_j^0 = 1 - f_j^1. \tag{2.5}$$

Ve vztahu 2.4 značí veličina $\pm A$ amplitudu signálu při modulaci, rx_j výstup z kanálu v čase j. Následně je třeba definovat koeficienty Q_{ij}^0 a Q_{ij}^1, které zastupují hodnoty bitových uzlů v Tannerově grafu. Mackey a Neal zjistili, že v zjednodušené verzi SD algoritmu lze tyto koeficienty formulovat dle vztahu 2.6

$$Q_{ij}^0 = f_j^0 \text{ a } Q_{ij}^1 = f_j^1. \tag{2.6}$$

Při iterativním SD dekódování se střídavě provádí horizontální a vertikální krok. Při horizontálním kroku jsou vypočítávány koeficienty R_{ij}^0 a R_{ij}^1 v součtových uzlech s Tannerova grafu. Toto situaci obecně vystihuje obrázek 2.1, kde horní index $x \in (0, 1)$.



Obr. 2.1: Horizontální krok SD algoritmu.

Pro výpočet koeficientů R_{ij}^0 a R_{ij}^1 je pak využíváno vztahu 2.7

$$\delta Q_{ij} = Q_{ij}^0 - Q_{ij}^1, \tag{2.7}$$

kde je vypočítán rozdíl koeficientů Q_{ij}^0 a Q_{ij}^1, ze kterého je následně dopočítán koeficient

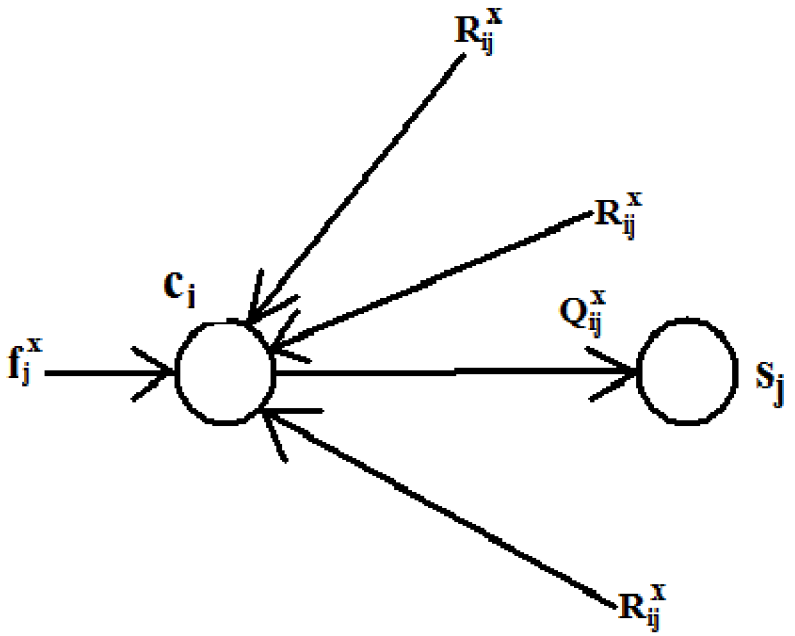
$$\delta R_{ij} = \prod_{j' \in N(i) \setminus j} \delta Q_{ij'}. \tag{2.8}$$

Ve vztahu 2.8 je patrný ekvivalent s dekódovací metodou HD, ve které se počítaly paritní součty také bez aktuálního prvku, pro který byl daná součet dělán. V metodě SD se navíc vychází ze součinů. Koeficienty R_{ij}^0 a R_{ij}^1 jsou již lehce vyjádřitelné pomocí vztahů 2.9 a 2.10

$$R_{ij}^0 = 0{,}5\left(1 + \delta R_{ij}\right), \tag{2.9}$$

$$R_{ij}^1 = 0{,}5\left(1 - \delta R_{ij}\right). \tag{2.10}$$

Po provedení horizontálního kroku se přechází k vertikálnímu kroku. Tento krok je zachycen na obrázku 2.2, kde je vše zachyceno z obecného hlediska.



Obr. 2.2: Vertikální krok SD algoritmu.

Aby bylo možné odvodit nové koeficienty Q_{ij}^0 a Q_{ij}^1, je nutné nejprve vyjádřit koeficienty Q_j^0 a Q_j^1 podle vztahu 2.11

$$Q_j^x = \alpha_{ij} f_j^x \prod_{i \in M(j)} R_{ij}^x. \tag{2.11}$$

Zde značí M bitové uzly c_i z Tannerova grafu, f_j^x vyjadřuje původní rozložení pravděpodobností v přijatém slovu na vstupu dekodéru a koeficient α_{ij} je dopočítán

podle vztahu formulovaného v rovnici 2.12

$$\alpha_{ij} = \frac{1}{f_j^0 \prod_{i \in M(j)} R_{ij}^0 + f_j^1 \prod_{i \in M(j)} R_{ij}^1}. \tag{2.12}$$

Hodnota dekódovaného výstupního vektoru $\mathbf{d}$ je získána lehce podle toho, který parametr Q_j^x je větší. Proto když $Q_j^0 > Q_j^1$ je hodnota $\mathbf{d}(j) = 0$, nebo v opačném případě $\mathbf{d}(j) = 1$.

Pro další iteraci je nutno aktualizovat také koeficienty f_j^0 a f_j^1 podle

$$f_j^0 = Q_j^0 \text{ a } f_j^1 = Q_j^1.$$

Proto lze díky vztahu 2.6 tvrdit, že koeficienty Q_{ij}^0 a Q_{ij}^1 jsou shodné s koeficienty Q_j^0 a Q_j^1.

Pro názornou ukázku funkce SD algoritmu je přiložen následující příklad, na kterém je ukázán postup při numerickém výpočtu první iterace pomocí tohoto dekódovacího algoritmu.

1. **Volba LDPC kódu**
   I tento příklad vychází z EG – LDPC kódu s paritní a generující maticí z kapitoly 1.3.3. Je definována vstupní posloupnost informačních znaků

   $$\mathbf{k} = 1001111,$$

   ze které po kódování vznikne zakódované slovo

   $$\mathbf{c} = 100101000011101$$

   z příkladu na kódováni LDPC kódu, uvedeného v kapitole 2.1. Vektor $\mathbf{c}$ je modulován pomocí BPSK s amplitudou ± 1V a následně je simulován přenos přes AWGN kanál.
2. **Inicializace SD algoritmu**
   Pomocí vztahu 2.4 a 2.5 jsou vypočítány inicializační pravděpodobnosti. Následující tabulka 2.3 ukazuje výsledky výpočtu těchto pravděpodobností a navíc je zde pro zajímavost přidán sloupec tvrdého rozhodnutí, kde je patrné, že vlivem přenosu byly do zprávy $\mathbf{c}$ vneseny $t = 3$ chyby. Aby byla úspěšně dokončena inicializace SD algoritmu je potřeba stanovit koeficienty Q_{ij}^0 a Q_{ij}^1 podle vztahu 2.6 a jejich rozdíl dle 2.7. Výpočet koeficientů Q_{ij}^0 a Q_{ij}^1 je přehledně zachycen dle použitého indexování v přílohách práce, konkrétně v tabulkách C.1, C.2. Kvůli vysokému rozsahu paritní matice $\mathbf{H}_{EG}$ je uvedena pouze část výpočtu, jenž bohatě postačuje k ověření numerického výpočtu.
3. **Horizontální krok**
   Horizontální krok je počítán podle vztahů 2.8, 2.9 a 2.10. Je zde opět podobnost s HD algoritmem, kde se pracuje s řádky paritní matice $\mathbf{H}_{EG}$, jenž je nyní

Tab. 2.3: Výpočet vstupních pravděpodobností SD algoritmu.

Pozice j	Zakódované slovo $\mathbf{c}$	Parametr f_j^0	Parametr f_j^1	Tvrdé rozhodnutí
1	1	0,0812	0,9188	1
2	0	0,9660	0,0340	0
3	0	0,2367	0,7633	1
4	1	0,0010	0,9990	1
5	0	0,2753	0,7247	1
6	1	0,0075	0,9925	1
7	0	0,9983	0,0017	0
8	0	0,7295	0,2705	0
9	0	0,1730	0,8270	1
10	0	0,8363	0,1637	0
11	1	0,0027	0,9973	1
12	1	0,0065	0,9935	1
13	1	0,1043	0,8957	1
14	0	0,8982	0,1018	0
15	1	0,3109	0,6891	1

vyplněna koeficienty Q_{ij}^0 a Q_{ij}^1. Výsledky výpočtů koeficientů R_{ij}^0 a R_{ij}^1 jsou shrnuty v tabulkách C.3 a C.4, které jsou pro přehlednost obsahem přílohy C.2.

4. **Vertikální krok a rozhodnutí o dalších iteracích SD algoritmu**
   Po dokončení vertikálního kroku jsou získány požadované hodnoty vektoru $\mathbf{d}$. Tyto hodnoty ještě nemusí být finální, protože se jedná o první iterační cyklus. Vektor $\mathbf{d}$ je získán v několika krocích, jenž jako celek tvoří vertikální krok. Tyto kroky jsou charakterizovány vztahem 2.11 pro výpočet koeficientů Q_j^0 a Q_j^1. Do vztahu 2.11 je nutno dosadit parametr α_{ij} definovaný vztahem 2.12. V tabulce 2.4 jsou prezentovány vypočítané parametry. Hodnota výstupního dekódovaného slova $\mathbf{d}$ je závislá na koeficientech Q_j^0 a Q_j^1. Protože je parametr α shodný pro všechny řádky ve sloupci, je prezentována pro každý sloupec hodnota α_j.
   Z tabulky 2.4 je patrné, že dekódování bylo úspěšně dokončeno již v prvním iteračním cyklu, protože

$$\mathbf{d} = \mathbf{c} = 100101000011101.$$

   Na tomto konkrétním příkladu dosahuje algoritmus SD korekční schopnosti $t = 3$ chyby již v prvním iteračním cyklu dekódování. Lze tedy konstatovat, že algoritmus SD dosáhl v daném případě lepšího výsledku než–li algoritmus HD.

Tab. 2.4: Výsledky 1. iterace SD algoritmu.

Pozice j	Parametr α_j	Parametr Q_j^0	Parametr Q_j^1	**d**	**c**
1	30,2556	0,1941	0,8059	1	1
2	7,1157	0,9964	0,0036	0	0
3	13,3074	0,9767	0,0233	0	0
4	25,1715	0,0013	0,9987	1	1
5	10,0004	0,9991	0,0009	0	0
6	48,4449	0,0405	0,9595	1	1
7	56,2000	0,9886	0,0114	0	0
8	17,8729	0,9443	0,0557	0	0
9	21,6926	0,9284	0,0716	0	0
10	31,8045	0,8712	0,1288	0	0
11	37,0489	0,0073	0,9927	1	1
12	41,1175	0,0272	0,9728	1	1
13	28,9281	0,0951	0,9049	1	1
14	25,5075	0,8556	0,1444	0	0
15	18,4105	0,0396	0,9604	1	1

Jak bude dále ukázáno, algoritmus SD dosahuje lepších výsledků i v BER charakteristikách při srovnání s doposud teoreticky popsanými dekódovacími algoritmy HD a BF. Na druhou stranu musí být přihlédnuto k časové náročnosti SD algoritmu, která zvláště pro rozsáhlé paritní matice **H** je mnohonásobně vyšší než u ostatních, v práci prezentovaných, dekódovacích algoritmů, což je dokázáno později v kapitole 4.2.3 zabývající se analýzou časové náročnost dekódovacích algoritmů.

2.2.4 Log Likelihood algoritmus

Tento iterativní dekódovací algoritmus byl představen Gallagerem v jeho práci [3]. V mírně obměněné podobě je tento algoritmus popisován také v [10]. Algoritmus LD vychází z metody SD popsané v 2.2.3, ale je obměněn, kdy je využito logaritmické vyjadřování inicializačních pravděpodobností, a také jsou používaný logaritmy při výpočtu horizontálního i vertikálního kroku, popsaných níže. Tento algoritmus dekódování LDPC kódu je nejefektivnější ze všech v práci popisovaných algoritmů. Za uvedení stojí, že časová náročnost vykonání jedné iterace dekódování pomocí LD algoritmu v LDPC kódech s rozměrem paritní matice **H** v řádech stovek bitů (např. EG – LDPC kód nebo PG – LDPC kód se stupněm $s = 5$), je několika násobně nižší než podobně výkonný SD algoritmus. Údaje o výkonnosti a časové náročnosti

dekódovacího algoritmu LD lze nalézt v kapitole 4, která se zabývá porovnáváním výsledků získaných pomocí programu Simulace LDPC kódů. Popis tohoto programu lze nalézt v kapitole 3.2.2.

Při počáteční inicializaci algoritmu LD se vychází ze vztahu 2.13

$$\ln \frac{1 - P_d}{P_d} = z_d \beta_d, \tag{2.13}$$

kde z_d označuje znaménko vypočítané pravděpodobnosti na levé straně rovnice a β_d absolutní část této pravděpodobnosti. Pravděpodobnost P značí šanci, že na pozici d bude logická 1. Inicializace je posléze dokončena výpočtem logaritmického podílu nalezených koeficientů β_d dle vztahu 2.14 [3]

$$Q_{ij} = ln \frac{e^{\beta} + 1}{e^{\beta} - 1}. \tag{2.14}$$

Koeficientem Q_{ij} se naplní paritní matice, tj. koeficient je dosazen na pozice, kde je v matici log. 1.

Jak bylo zmíněno výše, algoritmus LD pracuje na shodném principu jako algoritmus SD, proto je cyklicky prováděn horizontální a vertikální krok. Horizontální krok je realizován sečtením koeficientů Q_{ij} v řádku, kdy je vynechávána aktuální koeficient, pro který je daný součet počítán. Zde je další rozdíl oproti SD algoritmu, který využíval součinu prvků v řádku.

$$r_{ij} = \sum_{j' \in N(i) \setminus j} Q_{ij'}. \tag{2.15}$$

$$R_{ij} = ln \frac{e^{r_{ij}} + 1}{e^{r_{ij}} - 1}. \tag{2.16}$$

Pomocí vztahů 2.15 a 2.16 je dokončen horizontální krok algoritmu. Zde se nesmí zapomínat udržovat aktuální údaj o znaménku, které je uschováno v proměnné z a je ke koeficientu R_{ij} přidáno. Znaménka jsou aktualizována podle 2.17

$$z_{ij} = \prod_{j' \in N(i) \setminus j} z_{ij'}. \tag{2.17}$$

Vertikální krok je stejně jako u SD algoritmu prováděn ve sloupcích paritní matice $\mathbf{H}$, která je aktuálně vyplněna prvky R_{ij}. Zde je opět proti SD dekódovacímu algoritmu rozdíl v tom, že se operuje se součtem prvků, nikoliv násobením prvků ve sloupcích. Vertikální krok lze popsat vztahem 2.18 pro nalezení nových koeficientů

$$Q_{ij} = Q_{ij} + \sum_{i' \in M(j) \setminus i} Q_{i'j} R_{ij}. \tag{2.18}$$

O hodnotě výstupního vektoru je pak rozhodnuto na základě výpočtu vztahu 2.19

$$Q_j = Q_{ij} + \sum_{i \in M(j)} R_{ij}. \tag{2.19}$$

Pak je již pouze odvozena hodnota vektoru $\mathbf{d}$ na základě dosazení do podmínek

$$Q_j > 0 \rightarrow \mathbf{d}(j) = 1$$

nebo

$$Q_j < 0 \rightarrow \mathbf{d}(j) = 0.$$

Postup při výpočtu algoritmu LD lze shrnout do několika kroků.

1. Inicializace je prováděna pomocí vztahů 2.13 a 2.14.
2. Horizontální krok je pak realizován sčítáním koeficientů pomocí vztahů 2.15 a 2.16. Je zde třeba také aktualizovat znaménku koeficientu R_{ij} podle 2.17.
3. Ve vertikálním kroku se pracuje se sloupci paritní matice, kdy jsou opět prováděny součty jednotlivých prvků. Tyto výpočty jsou vedeny podle 2.18, kdy jsou získány koeficienty pro novou iteraci. Tyto koeficienty jsou použity pouze za předpokladu, že není dekódování úspěšné již v první iteraci. Správnost dekódování je ověřována porovnáním dekódovaného vektoru $\mathbf{d}$ se zakódovaným vektorem $\mathbf{c}$.
4. Pokud nebylo dekódování úspěšné jsou opakovány kroky 2 a 3 dokud není splněna podmínka úspěšného dekódování nebo není překročena hodnota určující maximální počet iterací.

Hlavní výhodou LD algoritmu je odbourání časově náročného násobení v horizontálním kroku algoritmu. Tento fakt je maximálně zřetelný při simulaci rozsáhlých kódů, kde je zachován výkon pro opravu chyb způsobených přenosem přes zarušený informační kanál, ale mnohonásobně zkrácen čas vykonávání jedné iterace. Při předpokladu, že se jedná o iterační algoritmus dekódování a je tedy opakován cyklus dekódování, je tento fakt naprosto zásadní. Více k tomuto tématu pojednává kapitola 4.2.3, jenž se zabývá srovnáním časové náročnosti dekódovacích algoritmů.

3 LDPC KÓDY V MATLABU

Ze zadání práce vyplývá téma další kapitoly, která se zabývá softwarovou realizací LDPC kódu v prostření Matlab. Pro vlastní realizaci zadání bylo zvoleno prostředí *MATLAB R2010a*, v němž vznikal veškerý software zabývající se tématikou LDPC kódů. Cílem celé tvorby byl vznik programu, jenž je použitelný jako výuková pomůcka pro demonstraci funkčnosti LDPC kódů jako celku, tak jako simulační nástroj pro získání výsledků, na jejichž základě bude možno porovnat jednotlivé metody přístupu k LDPC kódům. Při tvorbě veškerých programů v prostředí Matlab bylo čerpáno z literatury [8], [15] a [16].

Tato kapitola je členěna do dvou oddílů. První oddíl, kapitola 3.1, se zaměřuje především na popsání algoritmů tvorby paritních matic **H**, kde bylo pro účel práce vybráno strukturované vytváření pomocí konečné geometrie. Dále jsou rozebírány všechny čtyři dekódovací algoritmy, jenž byly teoreticky přiblíženy v kapitole 2.2.

Druhý oddíl, kapitola 3.2, je prezentována vlastní grafická aplikace LDPC kódy. Tato aplikace funguje pouze jako úvodní okno pro výběr jednoho ze dvou vytvořených programů. Na výběr jsou Výuka LDPC kódů nebo Simulace LDPC kódů. Detailnější popis fungování a ovládání těchto programů je obsahem této kapitoly.

3.1 Algoritmizace LDPC kódů v Matlabu

Jak bylo zmíněno výše, obsahem kapitoly bude popis jednotlivých algoritmů pro vytváření a dekódování LDPC kódů. Všechny popisované algoritmy byly teoreticky přiblíženy v předchozím textu, především v kapitole 1.3.3 zabývající se tvorbou paritních matic pomocí konečné geometrie a v kapitole 2.2 věnované dekódovacím metodám pro LDPC kódy.

3.1.1 Algoritmizace vytváření LDPC kódů

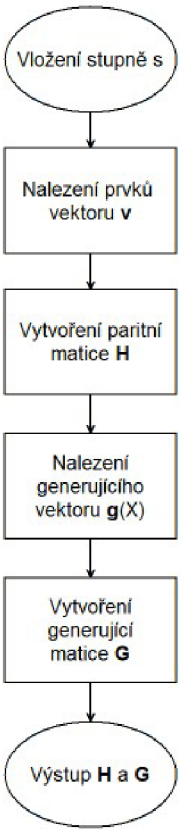
V následujícím textu jsou popsány algoritmy tvorby paritní matice **H** a generující matice **G** pomocí strukturovaných metod vytváření

- Euklidovské geometrie,
- projektivní geometrie.

EG v Matlabu

Při implementaci algoritmu Euklidovské geometrie do prostředí Matlab bylo vycházeno z popisu algoritmu v kapitole 1.3.3. Konkrétně je postupováno podle prezentovaného příkladu. Snahou bylo vytvořit funkci, jenž bude mít za vstupní parametr

stupeň s a jako výstup bude vracet $\mathbf{H}_{EG}(s)$ a $\mathbf{G}_{EG}(s)$ v systematickém tvaru. Tento postup lze zjednodušeně graficky znázornit pomocí vývojového diagramu zachyceného na obrázku 3.1.



Obr. 3.1: Vývojový diagram funkce EG.

Z vývojového diagramu je patrné, že vytvořená funkce má vstupní parametr s, s nímž operuje celý algoritmus. Volbou stupně s posléze získáváme EG – LDPC kódy zachycené v tabulce 1.1. Minimální velikost stupně paritní a generující matice EG je tedy $s = 2$.

Při implementaci EG algoritmu bylo nezbytné pracovat s GF, kde bylo čerpáno především z [9]. Byly využity funkce pro vytvoření GF nebo matice primitivních elementů, jenž byla stvořena k nalezení koeficientů jednotlivých bodů ve vektoru $\mathbf{v}$. Samotná paritní matice $\mathbf{H}_{EG}(s)$ poté vznikla cirkulací vektoru $\mathbf{v}$.

Před samotnou tvorbou generující matice muselo předcházet nalezení nulových bodů a vytvoření generujícího polynomu $\mathbf{g}_{EG}(X)$ dle vztahů 1.14 – 1.16. Generující matice pak vznikla posouváním nalezeného generujícího vektoru $\mathbf{g}_{EG}(X)$, které bylo realizováno přesně tolikrát, kolik bylo definováno informačních znaků k daného

kódu, sníženého o hodnotu jedna. V posledním kroku byla generující matice $\mathbf{G}_{EG}(s)$ převedena na systematický tvar pomocí dvojitého cyklu *for*.

PG v Matlabu

Algoritmus projektivní geometrie opět vychází z teoretického základu, jímž se zaobírala kapitola 1.3.3. Tento algoritmus je velmi podobný EG algoritmu, a proto ho lze popsat stejným vývojovým diagrame, tedy obrázkem 3.1. Tímto způsobem vznikají PG – LDPC kódy s klíčovým parametrem s, který je použit jako vstupní parametr vytvořené funkce PG. Přehled možných kódu lze nalézt v tabulce 1.4.

Vlastní programové řešení opět vychází z GF, kde je změna oproti EG algoritmu především ve velikosti matice primitivních elementů a v definování parametru β. Při hledání bodů vektoru $\mathbf{v}$ se využívá dvojitý cyklus *for* s testováním podmínky shody, což při velikosti paritní matice $\mathbf{H}_{PG}(5)$ vede k vnesení vyšší časové prodlevy při vykonávání funkce, než je tomu například u algoritmu EG s $\mathbf{H}_{EG}(5)$. Toto zpoždění kompenzují o něco lepší kódové parametry PG při srovnání s EG.

Paritní matice $\mathbf{H}_{PG}(s)$ je stejně jako u algoritmu EG vytvořena rotací vektoru $\mathbf{v}$. Generující matice $\mathbf{G}_{PG}(s)$ vzniká opět z generujícího polynomu $\mathbf{g}_{PG}(X)$, při jehož výpočtu se používá podobným matematický aparát jako v případě EG. Generující polynom je potřeba převézt z GF tvaru na binární tvar a posléze s každým řádkem realizovat bitový posun vpravo. V posledním kroku je opět generující matice $\mathbf{G}_{PG}(s)$ převedena do systematického tvaru, pro snadnější dekódování vstupní posloupnosti informačních znaků $\mathbf{k}$ z dekódovaného slova $\mathbf{d}$.

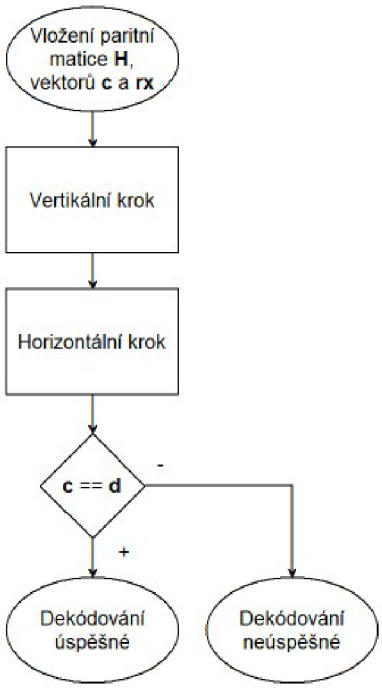
3.1.2 Algoritmizace dekódovacích metod LDPC kódů

Dalším důležitým krokem při tvorbě programu LDPC kódy v prostředí Matlab byla algoritmizace dekódovacích metod LDPC kódů. V následujícím textu je postupně rozebrána každá z metod popsaných výše.

HD v Matlabu

Nejjednodušší dekódovací metoda prezentovaná v práci je Hard Decision. Její jednoduchost vyplývá z faktu, že se nejedná o iterativní algoritmus, jak je naznačeno na obrázku 3.2.

Vytvořená funkce HD tedy předpokládá za vstupní proměnné paritní matici $\mathbf{H}$ použité geometrie, vektor zakódované posloupnosti informačních znaků $\mathbf{c}$, jímž se rozumí data vstupující na přenosový kanál a binární vektor $\mathbf{rx}$, o jehož podobě se rozhoduje pomocí tvrdého rozhodnutí z příchozích dat na vstupu dekodéru.



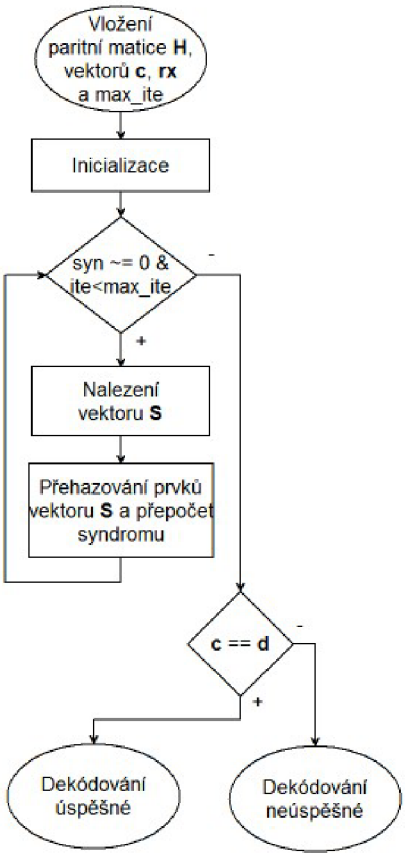
Obr. 3.2: Vývojový diagram funkce HD.

Další proces dekódování vychází z principu výměny informací mezi bitovými a součtovými uzly Tannerova grafu, formulovaného v 1.4. Tento princip je ve vývojovém diagramu 3.2 reprezentován blokem vertikální a horizontální krok, kdy algoritmus provádí potřebné matematické operace pro získání dekódovaného slova **d**. Následně jsou porovnány vektory **c** a **d**, čímž se rozhodne o úspěšnosti dekódování. V praxi by toto možné nebylo, ale zde je k tomu přistoupeno z důvodu názornější prezentace výsledků, především v programu Výuka LDPC kódů.

BF v Matlabu

Prvním ze skupiny iterativních dekódovacích metod pro LDPC kódy, jimiž se zabývá tato práce, je Bit-Flipping algoritmus. Na základě tohoto algoritmu vznikla v prostředí Matlab funkce BF, jejíž vývojový diagram je znázorněn na obrázku 3.3.

Podobně jako funkce HD i tato funkce využívá jako vstupní parametry paritní matici **H** dané geometrie, vektor zakódované zprávy **c**, binární vektor **rx**, který vstupuje do BF dekodéru a maximální hodnotu iterací, definovanou proměnnou *max_ite*.



Obr. 3.3: Vývojový diagram funkce BF.

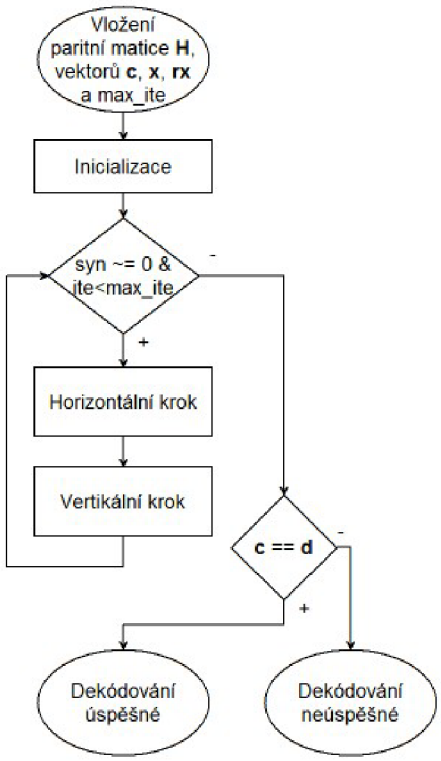
Algoritmus BF je charakteristický svou práci se syndromem matice, jak bylo popsáno v 2.2.2. Právě vektor syndromu paritní matice **H** je počítán během inicializace metody a uložen do proměnné *syn*. Následuje vlastní tělo algoritmu, jenž je tvořeno cyklem *while*, který je opakován dokud je splněna první podmínka z vývojového diagramu 3.3. V každém cyklu je zvýšena hodnota proměnné *ite* o hodnotu jedna a znovu přepočítán syndrom matice **H**.

V těle cyklu *while* se provede vynásobení syndromů se sloupci matice H, kdy je výsledek uložen do proměnné S. Následně je porovnávána hodnota proměnné S na jednotlivých pozicích a ukládají se pozice, kde je následující hodnota vyšší než předchozí. V posledním kroku cyklu jsou již pouze přehozeny bity ve výstupním vektoru **d**.

O výstupu dekódování se rozhoduje obdobně jako v případě metody HD, tedy porovnáním vektoru **c** s vektorem **d**. Algoritmus BF je časově nenáročným iterativním dekódovacím algoritmem, kdy jeho výkon v charakteristikách BER roste se zvyšujícím se počtem iteracemi pozvolně, jak bude ukázáno později.

SD v Matlabu

Dalším iterativním algoritmem, jenž již nepracuje s binárním vektorem, ale přímo s výstupem z informačního kanálu zarušeného AWGN šumem, je The Sum-Product algoritmus. Při implementaci do prostředí Matlab byla zrealizována snaha o algoritmizaci metody SD. Tato snaha vyústila do funkce SD, jenž pomocí vstupních parametrů (paritní matice **H**, výstup z informačního kanálu **x**, tvrdé rozhodnutí z vektoru **x** reprezentované vektorem **rx**, binární výstup z kodéru **c** a hodnota max_ite) vrací hodnotu dekódovaného slova **d** a informaci o úspěšnosti dekódování. Takto vytvořenou funkci lze popsat pomocí vývojového diagramu zachyceného na obrázku 3.4.



Obr. 3.4: Vývojový diagram funkce SD.

Z vývojového diagramu metody SD je patrné, že je založena na obdobném principu jako metoda HD na což již bylo upozorňováno v předchozí kapitole 2.2. Rozdíl je ovšem v implementaci horizontálního a vertikálního kroku, jenž jsou u této metody popsány vztahy 2.6 – 2.12 z kapitoly 2.2.3. Předpokladem pro korektní funkci algoritmu je výpočet vstupních inicializačních pravděpodobností výskytu logické

hodnoty 1 a inverzní pravděpodobnosti výskytu logické hodnoty 0. Tyto pravděpodobnosti se získávají při inicializaci algoritmu a posléze jsou na závěr každé iterace algoritmu přepočítávány.

Funkce SD provádí opakovaně kroky horizontálního a vertikálního kroku, pro zadané množství iterací. Po dokončení posloupnosti uvedených kroků, je vypočítán výstupní vektor **d** a s jeho pomocí přepočítán vektor syndromů, jenž pro úspěšné dekódování musí být nulový. Pokud není splněna některá z podmínek v prvním větvení vývojového diagramu 3.4, postupuje program ve vykonávání druhé větve, ve které je pouze porovnán dekódovaný vektor **d** s vektorem **c**. Podle výsledku tohoto porovnání je vypsána informace o úspěšnosti dekódování.

Algoritmus SD se řadí do skupiny výkonnější dekódovacích algoritmů, jimiž se práce zabývá. Výkonem v opravování chyb s přehledem překonává algoritmy HD a BF. Nevýhoda dekódovacího algoritmu SD je jeho časová náročnost provádění jednoho cyklu dekódování což se maximálně projevuje pro paritní matice od stupně $s = 5$. Vzniklé zpoždění je způsobeno častým použitím operace násobení při výpočtu horizontálního a vertikálního kroku. Konkrétní výsledky lze vyčíst z kapitoly 4.2 zabývající se zpracováním výsledků simulací dekódovacích algoritmů LDCP kódů.

LD v Matlabu

Posledním popisovaným dekódovacím algoritmem LDPC kódů v této práci je Log Likelihood algoritmus. Vývojový diagram této metody je principem shodný s vývojovým diagramem metody SD, uvedeným na obrázku 3.4. Ovšem obsah funkce LD v prostředí Matlab je odlišný.

Funkce LD využívá shodné vstupní parametry jako funkce SD a požadovaná funkce je také identická. Rozdíl je v cestě za dosažením daného výsledku, tj. dekódovaného slova **d**. Algoritmus LD se vyhýbá použití operace násobení při realizaci horizontálního a vertikálního kroku. Vychází ze vztahů 2.13 – 2.19 popsaných v kapitole 2.2.4 zabývající se teoretickým popisem metody LD. Samotná implementace do prostředí Matlab vyžadovala pouze převedení popsaných vztahů do adekvátního tvaru definovaného tímto prostředím.

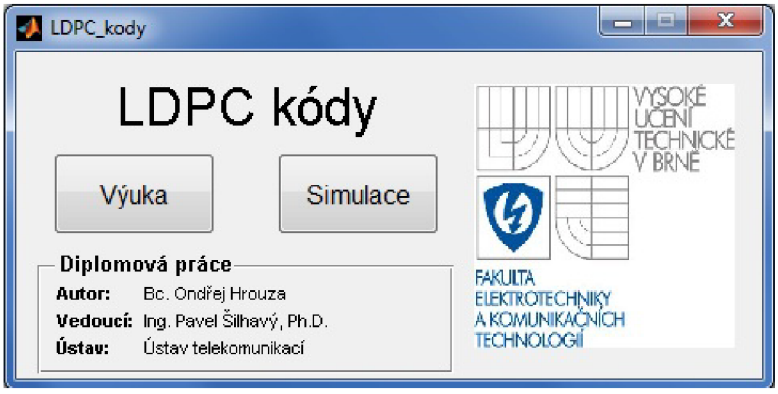
Při průchodu vývojovým diagramem 3.4 je testována pomocí cyklu *while* stejná podmínka jako u předešlých metod a po jejím nesplnění se program přesune na druhou větev, ve které se určí výsledek dekódování. Počet opakování hlavní kostry algoritmu je opět závislý na počtu zadaných iterací a na rychlosti konvergence k pozitivnímu výsledku dekódování.

Algoritmus LD má nejlepší poměr výkon/časová náročnost ze všech v práci probíraných dekódovacích algoritmů, což bude dokázáno v kapitole 4. Tohoto výsledku dosahuje především z důvodu nahrazení operace násobení součtem logaritmů.

3.2 Program LDPC kódy

V této podkapitole jsou blíže popsány vlastní grafické aplikace vzniklé v prostředí Matlab z algoritmů popsaných v kapitole 3.1. Toto grafické rozhraní, pojmenované souhrnně LDPC kódy, přináší koncovému uživateli program, pomocí něhož je schopen otestovat vytvářecí a dekódovací algoritmy LDPC kódů teoreticky popsané v této práci.

Při spuštění programu LDPC kódy se jako úvodní obrazovka objeví okno zachycené na obrázku 3.5. V tomto okně jsou zaznamenány obecné údaje o programu



Obr. 3.5: Úvodní menu programu LDPC kódy.

LDPC kódy, díky nímž lze program blíže identifikovat. Dále jsou pro uživatele přichystány dvě možnosti jak pokračovat, reprezentované tlačítky (Výuka a Simulace). Při akci stisknutí libovolného z tlačítek se program přepne do jednoho z podprogramů

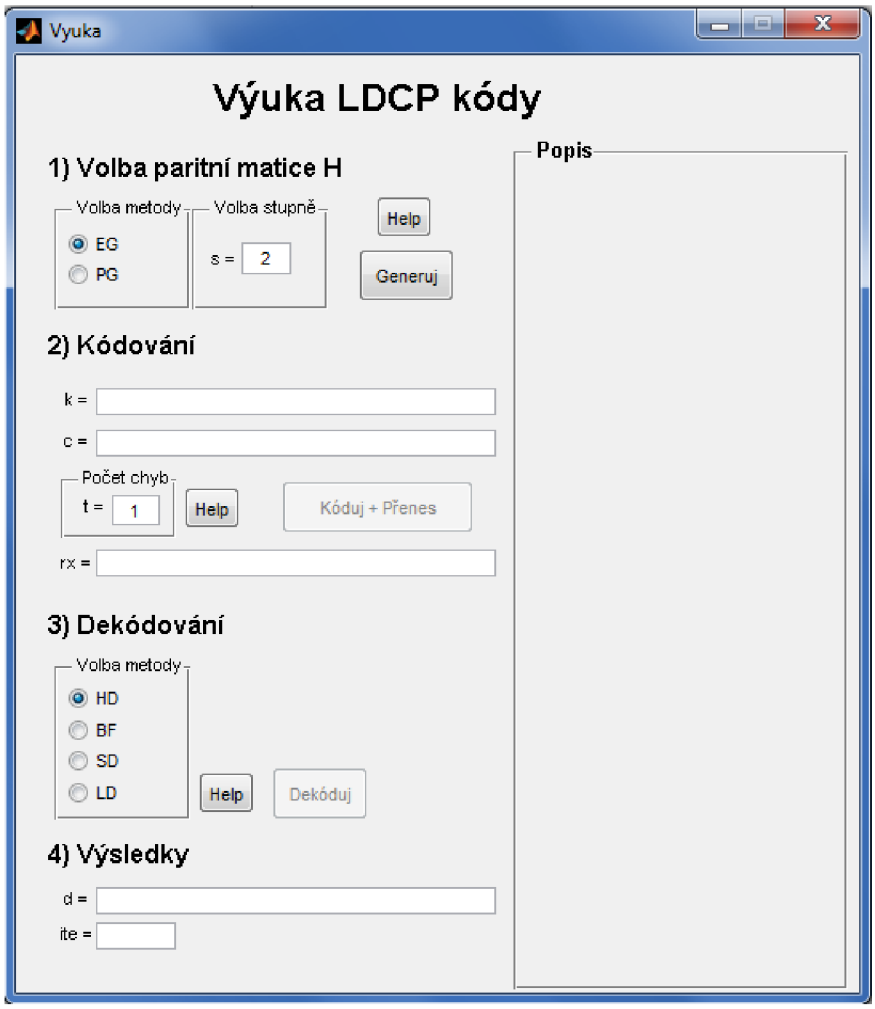
- Výuka LDPC kódy,
- Simulace LDPC kódy,

které již podle názvu napovídají svou funkci. Nepatrným omezením pro uživatele je, že nelze spustit oba podprogramy najednou, protože aktivní může být vždy pouze jeden. Popis grafického rozhraní těchto programů je obsahem následujících podkapitol práce.

3.2.1 Program Výuka LDPC kódy

Program Výuka LDPC kódy patří k základních pilířům této práce. Aplikace umožňuje rychlé ověření funkčnosti kodeku LDPC kódů. Libovolný uživatel je při práci s programem Výuka LDPC kódu provázen výstižnou nápovědou, kdy jsou popsány veškeré vstupy ze strany uživatele a postup při procházení programu. Celková koncepce výukového programu LDPC kódů je postavena co nejjednodušeji, aby byla obsluha celé aplikace uživatelsky co nejefektivnější.

Po spuštění programu Výuka LDPC kódy, pomocí tlačítka Výuka v úvodním menu aplikace LDPC kódy, se zobrazí okno znázorněné na obrázku 3.6. Grafické



Obr. 3.6: Program Výuka LDPC kódy po spuštění.

rozhraní aplikace Výuka LDPC kódy lze rozdělit na dvě poloviny. V levé polovině si koncový uživatel volí potřebná nastavení, kdy je optimální postupovat od shora dolů. Podle aktuálního nadpisu uživatel snadno rozpozná co právě nastavuje a pokud si není jistý, je výhodné využít tlačítka Help, jenž je obsaženo v každé sekci. Po stisknutí tlačítka Help se v popředí vytvoří okno nápovědy s veškerými potřebnými instrukcemi k úspěšnému použití programu Výuka LDPC kódů.

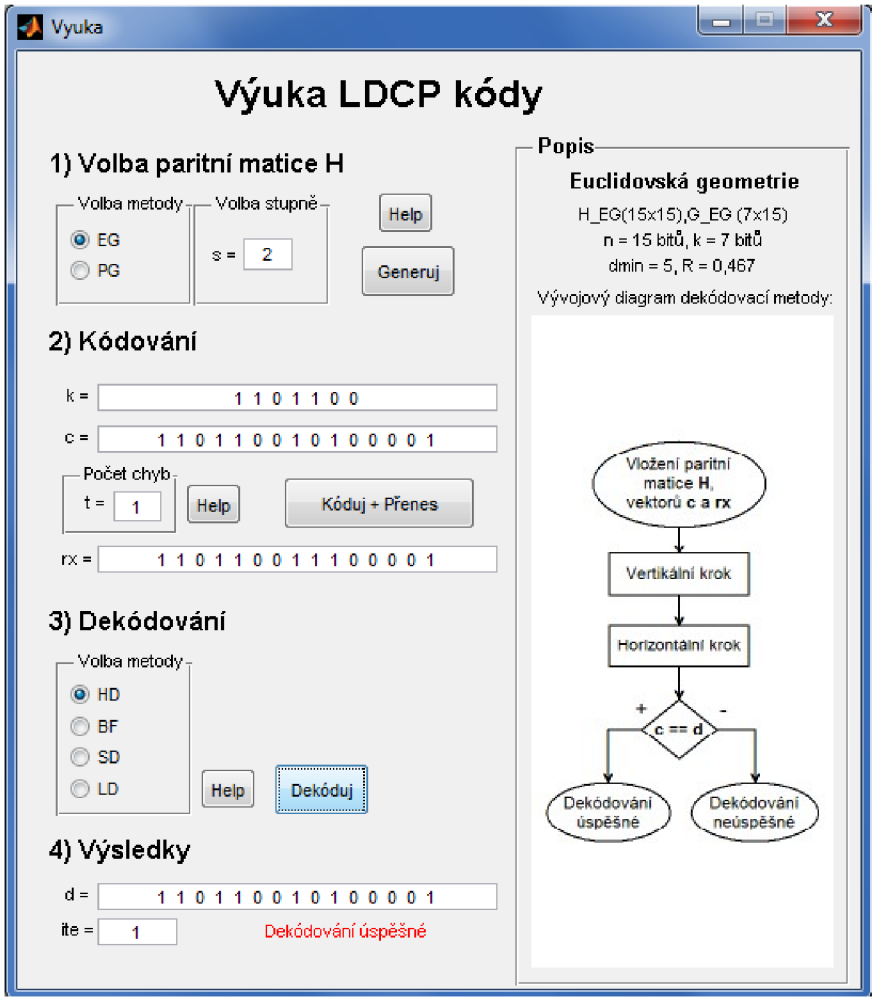
Protože je ve výukovém programu kladen velký důraz na jednoduchost, je uživateli umožněno volit pouze jednu vytvářecí metodu LDPC kódu (EG nebo PG) a maximální stupeň $s = 4$. Dále je umožněno zvolit počet chyb vnořených do zakódované zprávy $\mathbf{c}$, jejichž maximální počet se rovná délce vektoru $\mathbf{c}$. Nakonec si uživatel zvolí libovolnou ze čtyř předdefinovaných dekódovacích metod (HD, BF, SD a LD) a v případě potřeby také zadá maximální počet iterací daného algoritmu. Po

zadání veškerých potřebných parametrů již stačí pouze stisknout tlačítko Dekóduj. Po několika okamžicích jsou získány výsledky. Výsledky se zde rozumí dekódovaná posloupnost **d** a potřebný počet iterací k získání dekódovaného slova. Důležitým výstupním faktorem je také hláška o úspěchů/neúspěchu dekódovaní, jenž se zobrazí červeným písmem ve spodní části okna. Dle této hlášky zjistí uživatel, zda byl vybraný LDPC kód schopen opravit požadovaný počet chyb v zadaném počtu iterací.

Pokud jsou úspěšně provedeny veškeré kroky programu, jimiž jsou

1. Volba paritní matice **H**,
2. Kódování,
3. Dekódování,

dojde program do stavu zachyceného na obrázku 3.7. Odtud je patrný význam pravé



Obr. 3.7: Program Výuka LDPC kódů po zobrazení výsledků.

poloviny grafického rozhraní programu Výuka LDPC kódů. Tato plocha slouží k variabilnímu popisu aktuálně zvolených parametrů LDPC kódu. Uživatel zde má přehledně zachyceny základní parametry použitého kódu jako jsou např. bitová délka

kódu n nebo počet informačních znaků k. Je zde také prezentován vývojový diagram použitého dekódovacího algoritmu, jenž přibližuje cestu programu při získávání požadovaného výsledku.

Ideální volbou pro názornou ukázku funkce kodeku LDPC kódů je zadání stupně paritní matice $s = 2$. Při této volbě jsou zobrazovány jednotlivé bity, kdy je na první pohled patrná poloha vnořených chyb ve slovu **rx** nebo dekódovaná posloupnost informačních znaků **k**, jimiž se rozumí prvních k-bitů dekódovaného slova **d**. Při volbě vyššího stupně s by bylo zobrazení jednotlivých bitů nepřehledné, a proto jsou zobrazeny pouze velikosti příslušných vektorů.

3.2.2 Program Simulace LDPC kódů

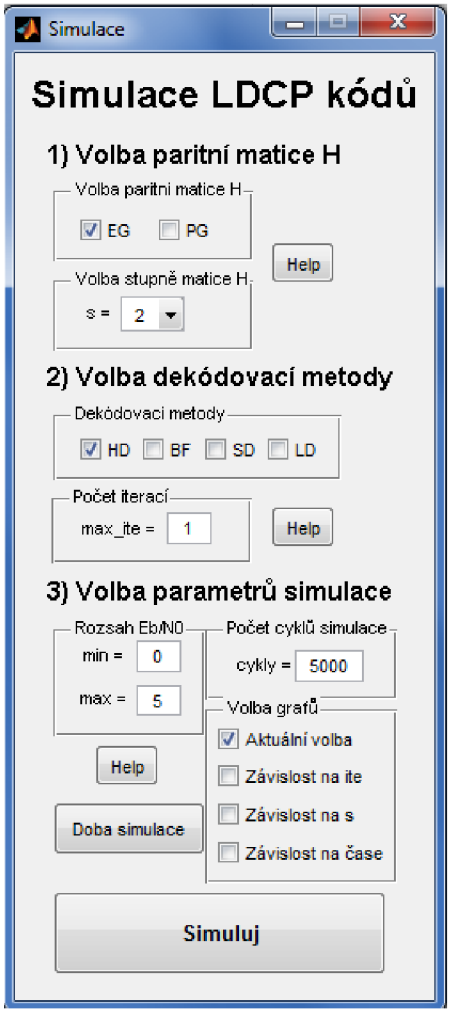
Další cestou z úvodního menu programu LDPC kódy je stisknutí tlačítka Simulace. Pří téhle volbě se spustí program Simulace LDPC kódy, jenž byl vytvořen za účelem výkonnostního porovnání dekódovacích algoritmů LDPC kódů popisovaných touhle prací. Simulační program nabízí uživateli mnoho možností, jak porovnávat jednotlivé algoritmy LDPC kódů. Grafická podoba programu Simulace LDPC kódů je zachycena na obrázku 3.8.

Z obrázku 3.8 je vidět podobná struktura grafického rozhraní jako v případě programu Výuka LDPC kódů, protože je v důsledku stále aplikována funkce kodeku LDPC kódů. Zde je ovšem kladen daleko vyšší důraz na uživatelskou variabilitu. Uživatel může libovolně kombinovat vytvářecí metody LDPC kódů spolu s dekódovacími algoritmy. Pro korektní funkci programu je opět vhodné postupovat od shora dolů po jednotlivých bodech a v případě potřeby využívat tlačítka nápovědy.

V prvním bodě je volen způsob vytváření kontrolní matice **H**, kdy mohou být klidně zvoleny obě vytvářecí metody současně. Vkládání stupně paritní matice je realizováno pomocí nabídky obsahující až stupeň $s = 5$, kdy vznikají matice o rozměrech vyšší než tisíc bitů, viz. popis v kapitole 1.3.3. Rozsáhlejší paritní matice jsou ideální pro simulaci BER charakteristik, protože se na ně díky jejich délce lépe aplikuje AWGN šum.

Bod dva programu Simulace LDPC kódy je určen pro volbu dekódovacích metod. Vybírá se stále ze stejné množiny algoritmů, jenž jsou probírány v práci. Rozdíl oproti výukovému programu je zde především v možnosti volby libovolného počtu dekódovacích algoritmů, což umožňuje snadné porovnávání zvolených dekódovacích metod v jednoho grafu.

Nejdůležitějším bodem z vytvořené aplikace Simulace LDPC kódy je závěrečný bod tři, ve kterém si uživatel volí parametry simulace. Může zde ovlivnit například rozsah E_b/N_0, který bude simulace akceptovat pouze pokud bude vyšší mez větší než



Obr. 3.8: Program Simulace LDPC kódů.

spodní mez. Rozsah E_b/N_0 je vhodné volit celočíselně, protože osa x je ve výstupním grafu dělena na dílky od spodní do horní meze s odstupem jednoho decibelu. Hodnota odstupu signálu od šumu nepřímo ovlivňuje počet chyb vnořených do zakódovaného slova $\mathbf{c}$ vlivem přenosu přes informační kanál. Dále si uživatel volí počet cyklů simulace, čímž se rozumí, kolikrát bude opakován cyklus kódování/dekódování pro zadaný LDPC kód. Počet cyklů ovlivňuje kvalitu výstupních dat, protože výstupní hodnoty BER charakteristik jsou průměrovány právě počtem zadaných cyklů. Zde ovšem musí být uživatel obezřetný, protože se zvyšujícím se počtem cyklů kodeku narůstá také čas výpočtu simulačního programu.

Velice užitečnou funkci v simulačním programu představuje tlačítko Doba simulace. Po stisknutí tohoto tlačítka je proveden odhad doby výpočtu výstupních dat ze zadaných parametrů. Tento odhad není přesný na sekundy, ale umožní uživateli přibližně určit dobu simulace. Takhle lze snadno předejít časově náročným simulacím trvajícím v řádech hodin, pokud je záměrem uživatel získat pouze zevrubné

srovnání určitého parametru LDPC kódů. Pro co nejpřesnější odhad času simulace je optimální stisk tlačítka Doba simulace opakovat, protože určení doby simulace je závislé na mnoha aspektech jako například aktuální vytížení procesoru, variabilitě trvání procesu dekódování v závislosti na aktuální hodnotě parametru E_b/N_0, rozptyl zpoždění při vytváření paritních matic nebo zkreslení doby výpočtu dekódovacího algoritmu vlivem vykonávání dalších instrukcí, které nejsou při odhadu uvažovány.

Volba výstupních charakteristik

Ve spodní části programu Simulace LDPC kódů, v části tři, dle obrázku 3.8 je *Volba grafů*, kde si uživatel volí mezi čtyřmi různými výstupy simulace. Jednotlivé výstupy jsou závislé na individuálním parametru simulace. Je samozřejmostí, že lze zvolit všechny možnosti najednou, ale potom lze očekávat výrazně zvýšenou dobu výpočtu simulace. Jednotlivé možnosti lze dělit podle závislosti na parametru takto:

- **Aktuální volba**
  Nejzákladnější simulace, jenž je závislá pouze na aktuálně zadaných vstupních parametrech (metoda vytváření LDPC kódu, dekódovací metody, počet cyklů simulace, atd.). Veškeré výstupní křivky jsou vyneseny do jednoho grafu. Proto lze snadno porovnávat vliv volby vytvářecí geometrie, nebo srovnat dekódovací algoritmy v závislosti na fixním zvoleném počtu iterací. Pouze v této volbě je také do výstupního grafu vynesena křivka nedekódovaného modulovaného signálu BPSK.
- **Závislost na iteracích**
  Ideální použití pro jednu dekódovací metodu, kdy se do grafu vykreslí křivky pro jednotlivé iterace. Aby v grafu nebylo přehnané množství křivek, jenž by způsobily nepřehlednost grafu, je volen rozsah iterací z množiny

  $$ite \in \{1, 2, 3, 5, max\},$$

  kdy musí být maximální hodnota $max > 5$. V opačném případě se vykreslí křivky od hodnoty $ite = 1$ až po hodnotu $ite = max$.
  Dále je možno v jednou grafu porovnávat buď obě vytvářecí metody EG s PG při jediné dekódovací metodě, nebo jednu vytvářecí metodu pro libovolný počet dekódovacích algoritmů. Při volbě obou geometrií a vyššího počtu dekódovacích metod se vykreslí grafy dva, kdy každý z grafů reprezentuje výstupní data pro danou geometrii.
- **Závislost na stupni** s
  Při téhle volbě je nejdůležitějším vstupním parametrem stupeň paritní matice s. Uživatel si zvolí maximální hodnotu tohoto parametru v bodě 1 z obrázku 3.8. Program poté provede simulaci pro každý stupeň od hodnoty $s = 2$

do $s = max$, kde hodnota max definuje zvolenou hodnotu. Takto lze lehce z jednoho grafu porovnat závislost velikosti paritní matice **H** na výkonnosti zvolené dekódovací metody.

Vykreslování do grafů je řešeno podobně jako v případě závislosti na iteracích, kdy je kladen důraz na to, aby nebyly grafy přehuštěny. Uživatel je ovšem schopen optimální volbou vstupních parametrů získat výsledky srovnání výkonnosti dekódovací metody LDPC kódu na typu a velikosti paritní matice.

- **Závislost na čase**

  Poslední možnou volbou a tedy zároveň typem výstupní charakteristiky je závislost dekódovacích algoritmů na čase. Zde je na osu y vynesen čas místo hodnoty BER jak tomu bylo v případě ostatních možností. Vlastní funkce programu je založena na měření času vykonání procesu kódování a následně dekódování zvolenou metodou či metodami. Pro vyšší variabilitu vykreslovaní do grafů je zde zakomponován také vliv závislosti na iteracích, jak bylo popsáno výše. V praxi to znamená, že si uživatel zvolí počet iterací a výstupem bude graf, kde místo hodnot BER pro jednotlivé iterace daného dekódovacího algoritmu budou zobrazeny změřené časy vykonávání procesu kodeku pro zvolený algoritmus v příslušné iteraci.

  Vykreslování do grafů je řešeno identicky jako v případě s volbou závislou na počtu iterací. Proto lze porovnávat například jednu dekódovací metodu v závislosti na iteracích, kdy je ve výsledku zachycena doba vykonávání jednoho cyklu procesu kódování/dekódování v jednom grafu. Lze také vytvořit souhrnný přehled pro každou vytvářecí geometrii a různé dekódovací algoritmy v závislosti na čase.

Pomocí výše popsaných možností lze provézt komplexní analýzu LDPC kódů založených na popisovaných vytvářecí a dekódovacích metodách. Proto je program Simulace LDPC kódů velice užitečnou pomůckou pro komparaci jednotlivých přístupů k dané problematice a spolu s programem Výuka LDPC kódů tvoří komplet, jenž je praktickým výstupem této práce. V následující kapitole 4 jsou dále komentovány výstupní grafy získané z popisované simulační aplikace Simulace LDPC kódy.

4 ANALÝZA VÝSLEDKŮ

Tato kapitola se zabývá analýzou výsledků získaných ze simulace LDPC kódu za pomocí programu Simulace LDPC kódů. Popis tohoto programu lze najít v kapitole 3.2.2. Analyzovány jsou jak metody vytváření LDPC kódů, tak samotné dekódovací algoritmy. Kombinací těchto faktorů lze získat vysoké množství různorodých výsledků, a proto se tato kapitola zaobírá pouze těmi nejnázornějšími, ze kterých lze vyvodit jasné závěry.

V následujících podkapitolách jsou postupně prezentovány výsledky srovnání vytvářecích geometrií a následně dekódovacích algoritmů LDPC kódů. Hlavními kritérii jsou zde především výkonnost dekódovacích algoritmů v závislosti na BER a časová náročnost vytváření a dekódování LDPC kódů.

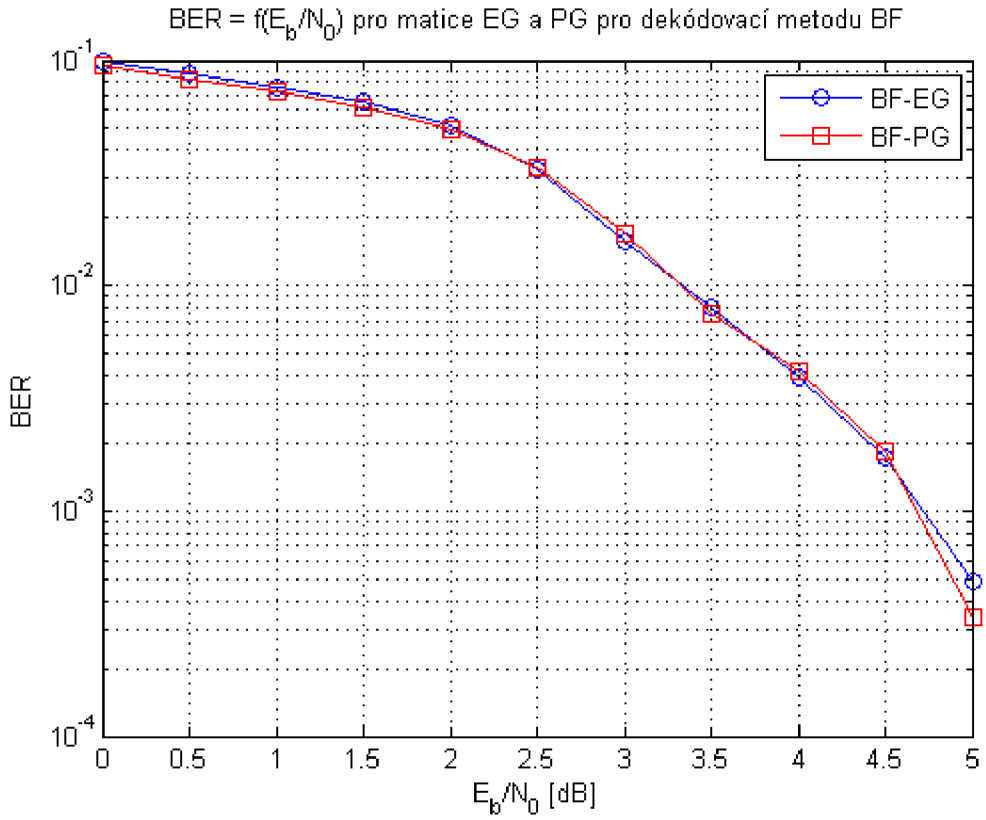
4.1 Analýza vytvářecích metod

Pro analýzu vytvářecích metod LDPC kódů jsou v práci vybrány pouze strukturované geometrické konstrukce. Mezi dva představitele této koncepce patři Euklidovská geometrie a projektivní geometrie. Tyto geometrie jsou po teoretické stránce popsány v kapitole 1.3.3 a jejich algoritmizace v prostředí Matlab je tématem kapitoly 3.1.1.

Vytvářecí metody se využívají pro tvorbu paritní matice $\mathbf{H}$ a generující matice $\mathbf{G}$, jenž jsou základem každého LDPC kódu. Jako vstupní parametr vytvářecích metod slouží stupeň s, s jehož pomocí vznikají LDPC kódy uvedené v tabulkách 1.1 a 1.4. Jako názorná ukázka porovnání analýzovaných vytvářecích metod LDPC kódů je zde použit obrázek 4.1.

Z grafu 4.1 je patrné, že obě vytvářecí geometrie mají podobné BER průběhy. Rozdíl v chybovosti BER v celém průběhu je velmi nízký, kdy pro geometrii PG vychází maximálně o dvě desetiny lépe, než pro geometrii EG, a to při hodnotě $E_b/N_0 = 5$ dB. V tomto konkrétním případě byl využit stupeň paritních matic $s = 5$, kdy vznikají matice o velikosti přes 1000 bitů a případný rozdíl by zde měl tedy být nejmarkantnější. Jako demonstrační dekódovací metoda je použit BF algoritmus při konstantním počtu iterací, jejichž počet zde není důležitý. Díky o něco lepší informační rychlosti R geometrie PG, oproti geometrii EG se stejným stupněm, se jeví využití projektivní geometrie jako schůdnější varianta.

U LDPC kódů založených na konečné geometrii s $m = 2$, kdy vnikají čtvercové paritní matice je možné dopředu dopočítat veškeré parametry vzniklého kódu. Jediným parametrem, jenž nelze předem dopočítat je čas. Proto se následující podkapitola 4.1.1 věnuje srovnání časů potřebných k vytvoření matic příslušné geometrie a velikosti. V tomto oddíle se také nachází další podkapitola 4.1.2, jejímž obsahem



Obr. 4.1: Graf srovnání vytvářecích metod LDPC kódů.

je porovnání paritních matic z hlediska chybovosti BER při změně jejich velikosti.

4.1.1 Časová náročnost vytvářecích metod

Protože výkonnost obou vytvářecích geometrií je podobná, jak bylo předvedeno v předchozím textu při analýze obrázku 4.1, je nutno přihlédnout k časové náročnosti daných vytvářecích algoritmů LDPC kódů. Toto zpoždění je vneseno při počáteční inicializaci LDPC kódů, kdy jsou generovány jak kontrolní matice **H**, tak generující matice **G**, jenž jsou nezbytné pro procesy kódování a dekódování.

Pro určení časů potřebných k vygenerování požadovaných matic bylo použito měření pomocí programu Matlab na počítači s procesorem Intel Core 2 Duo (frekvence každého jádra 2,2 GHz) a pamětí RAM o velikosti 4 GB. Tento procesor byl využit i při dalších měřeních času uvedených v této práci. Dané měření poté probíhalo tak, že byly střídavě spouštěny algoritmy vytváření LDPC kódů (EG a PG) a měřena doba výpočtu. Měřené časy byly posléze průměrovány k získání co nejpřesnějších hodnot. Přehled získaných hodnot zachycuje tabulka 4.1.

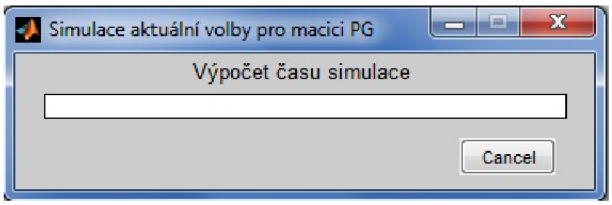
Z této tabulky je zřetelné, že doba vytváření projektivní geometrie je vyšší než doba potřebná pro vytvoření Euklidovské geometrie. Tento rozdíl není patrný pro

Tab. 4.1: Přehled časové náročnosti vytvářecích geometrií.

Stupeň s	EG [s]	PG [s]
2	0,0324	0,0387
3	0,084	0,1844
4	0,6231	2,5761
5	15,2263	60,1262

geometrie do stupně $s = 3$, ale pro vyšší hodnoty s je doba vytvoření paritní a generující matice geometrie PG přibližně čtyřnásobně vyšší něž v případě geometrie EG. V praxi tento nepoměr nehraje výraznou roli, protože potřebné matice LDPC kódů jsou vytvořeny předem a vloženy do paměťového prostoru kodéru respektive dekodéru. Zde je přihlíženo především na velikost rámce, jenž má být zakódován, případně na velikost jednotlivých fragmentů rámce. Z tohoto požadavku vyplývá volba vytvářecí matice LDPC kódů.

Hodnoty z tabulky 4.1 mají význam pro vzniklý program Simulace LDPC kódů, popsaným v kapitole 3.2.2, kdy uživatel musí předpokládat vstupní zpoždění především při volbě PG se stupněm $s = 5$. O toto zpoždění je prodloužena prodleva při prvním kroku na waitbaru, znázorněného na obrázku 4.2, jenž je spuštěn ihned po stisknutí tlačítka Simuluj.

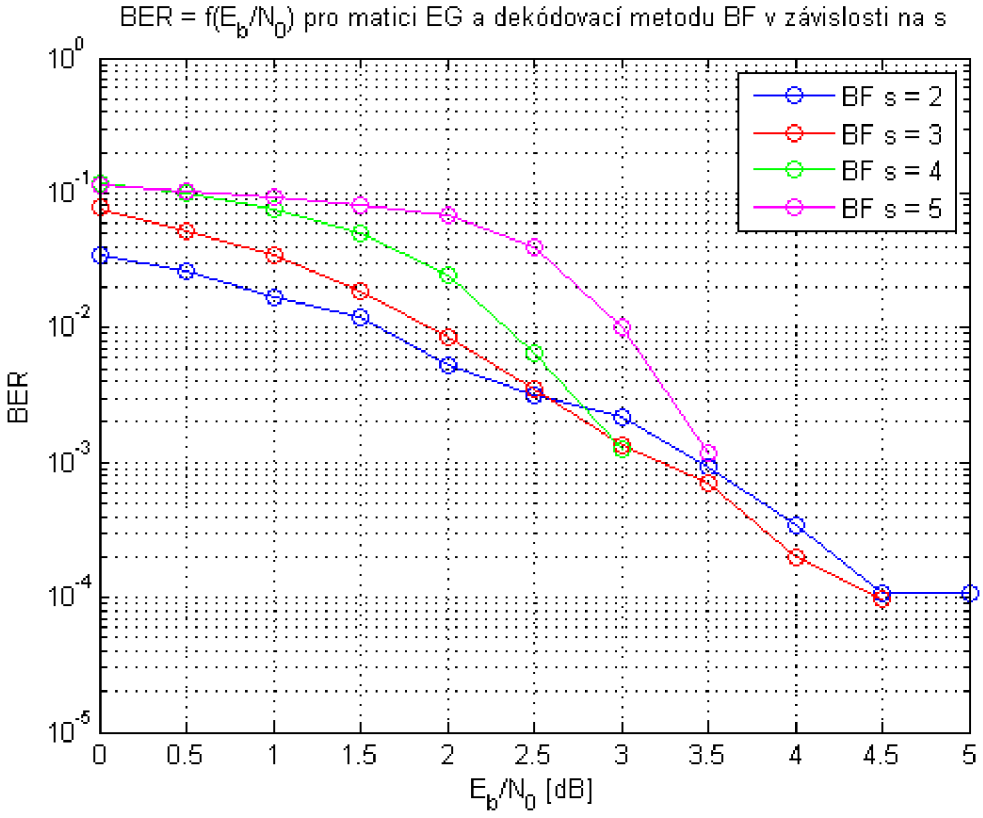


Obr. 4.2: Waitbar programu Simulace LDPC kódů.

4.1.2 Výkonnost LDPC kódů v závislosti na velikosti paritní matice

Po srovnání vytvářecích metod LDPC kódů z hlediska časové náročnosti pro generování potřebných matic, následuje výkonnostní porovnání v závislosti na velikosti paritní matice. Protože obě vytvářecí geometrie mají minimální rozdíl ve výkonnosti BER, je zde pro srovnání využita pouze vytvářecí geometrie EG. Cílem je určit optimální velikost paritní matice (stupeň s) daného LDPC kódu pro dosažení co nejlepšího poměru výkon/čas.

Pro simulaci výkonnosti EG LDPC kódů v závislosti na velikosti stupně s lze využít program Simulace LDPC kódů, kde se nachází možnost zvolit typ simulace v závislosti na stupni s. Tímto způsobem byl získán graf zachycený na obrázku 4.3. Tento graf ukazuje vliv velikosti paritní matice EG na funkci $BER = \mathrm{f}(E_b/N_0)$ pro



Obr. 4.3: Graf závislosti EG na stupni s.

dekódovací metodu BF při maximálním počtu iterací rovných $ite = 100$. Při malé hodnotě stupně $s = 2$ a $s = 3$ je vidět, že i při relativně vysoké hodnotě odstupu signálu od šumu ($E_b/N_0 = 4{,}5$ dB) nevede proces dekódování k příznivému výsledku, protože výstupní chybovost se stále pohybuje kolem $BER = 10^{-4}$. Naopak při vysokém zarušení kanálu je schopen algoritmus BF dosáhnout lepších výsledků, než pro vyšší stupně $s = 4$ a $s = 5$. To je způsobeno délkou takto vzniklého LDPC kódu (viz. tabulka 1.1), kdy vlivem zarušení informačního kanálu vzniká u delších slov mnohem více chyb než u kratších slov. Dekódovací algoritmus BF, stejně jako ostatní, v práci popisované dekódovací algoritmy, poté dosahuje lepšího výkonu v korekční schopnosti pro kódy s nižším obsahem chyb. To ovšem neznamená, že LDPC kódy s vytvářecími geometriemi se stupněm $s = 2$ nebo $s = 3$ jsou lepší než kódy se stupněm $s = 4$ či $s = 5$.

Se vzrůstající délkou kódu n roste také minimální vzdálenost d_{min} a je tedy

možné opravovat vyšší počet nezávislých chyb t ve vektoru zakódovaného slova $\mathbf{c}$. Zlom na obrázku 4.3 nastává pro stupeň $s = 4$ na hodnotě odstupu signálu od šumu $E_b/N_0 = 3$ dB, kdy dekódovací metoda BF vykazuje přibližně $BER = 10^{-3}$. Při dalším zvyšování hodnoty E_b/N_0 je již dekódování vždy úspěšné na zvolené množině dat. Stejně to platí také pro $s = 5$, kdy na hodnotě odstupu signálu od šumu $E_b/N_0 = 3{,}5$ dB je změřena poslední chybovost. Dále již při zvyšování odstupu signálu od šumu je metoda BF také stoprocentně úspěšná, což se nedá tvrdit pro kódy s vytvářecími maticemi se stupni $s = 2$ a $s = 3$, které vykazují určitou chybovost i nadále.

Při volbě vytvářecích matic je tedy nutno uvažovat nejen délku vstupního bloku dat, ale také předpokládanou zarušenost informačního kanálu. Pro nízké hodnoty odstup signálu od šumu SNR je zbytečné využívat velké vytvářecí matice LDPC kódu, protože daná chybovost nebude opravena. Pro tyto účely je vhodnější vstupní data dále dělit na menší bloky, kdy budou výsledky dekódování pozitivnější. Naopak pro méně zarušené informační kanály je vhodné využít rozsáhlejší LDPC kódy, v práci představované především stupněm $s = 4$ a $s = 5$, protože kapacita takto vzniklého kódu umožňuje dekódovacím metodám získat vyšší výkon něž při krátkých kódech.

4.2 Analýza dekódovacích metod

Následující text se věnuje porovnávání jednotlivých dekódovacích metod LDPC kódů, které byly teoreticky popsány v kapitole 2.2 a jejich algoritmizace do programového prostředí Matlab je obsahem kapitoly 3.1.2. Dekódovací algoritmy jsou opět porovnávány především na základě BER charakteristik, ale není zapomenuto také na časovou náročnost jednotlivých dekódovacích metod.

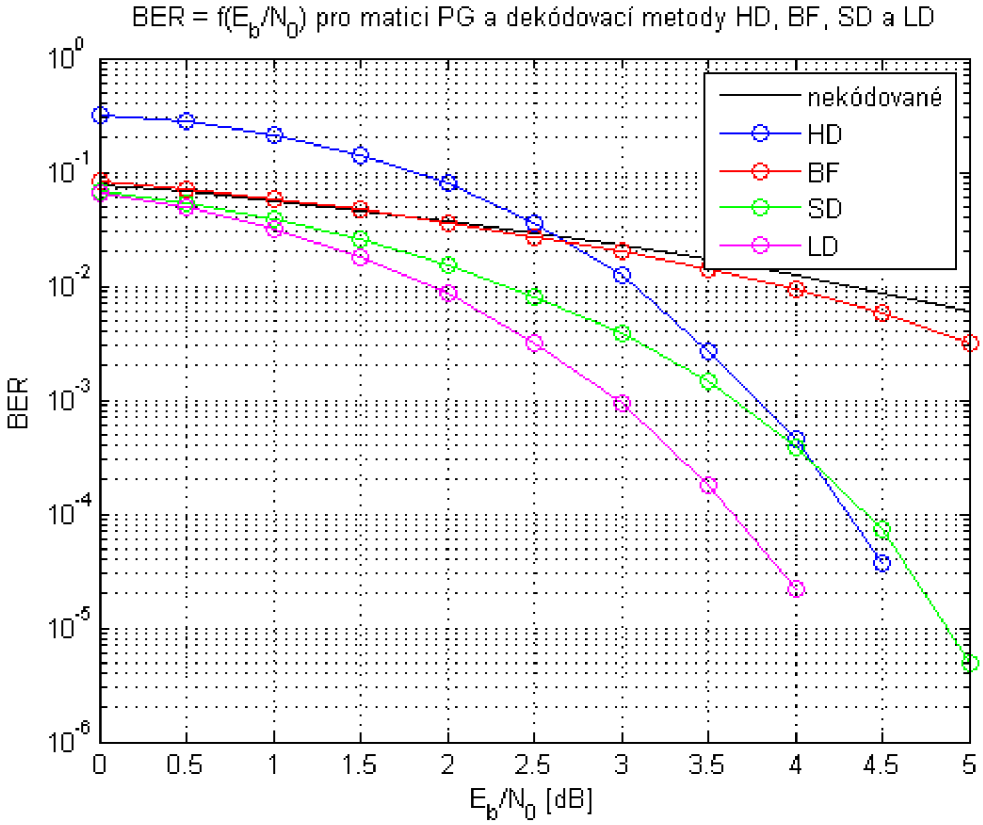
Jak bylo popsáno výše, pomocí programu Simulace LDPC kódů, lze vytvořit různorodé kombinace vytvářecích metod a dekódovacích algoritmů. Proto lze srovnávat LDPC kódy z různých pohledů jako jsou například

- typ vytvářecí geometrie (EG nebo PG),
- velikost vytvářecí matic – stupeň (s),
- počet iterací dekódovacích algoritmů,
- časová náročnost dekódovacích algoritmů.

První dva body již byly popisovány v předchozí kapitole 4.1, kde byl brán zřetel především na stranu vytvářecích geometrií. V této kapitole jsou již analyzovány dekódovací algoritmy, proto jsou následující podkapitoly děleny tak, aby co nejlépe vystihly obsáhlou oblast srovnání dekódovacích algoritmů LDPC kódů.

4.2.1 Výkonnostní porovnání dekódovacích algoritmů

Při porovnávání dekódovacích algoritmů LDPC kódů, popisovaných touto prací, musí být brán ohled na algoritmus HD, který byl do práce přidán spíše pro výukové účely. Tento algoritmus totiž není iterativním algoritmem, jako všechny ostatní dekódovací algoritmy. Jak bude ukázáno později v kapitole 4.2.2, výkon iterativních algoritmů v BER charakteristikách prudce roste s počtem iterací. Proto srovnání dekódovacích metod na obrázku 4.4 nemá plnohodnotný vyjadřovací význam, protože prezentuje srovnání první iterace dekódovacích algoritmů na projektivní geometrii stupně $s = 4$.



Obr. 4.4: Graf srovnání dekódovacích algoritmů v 1. iteraci.

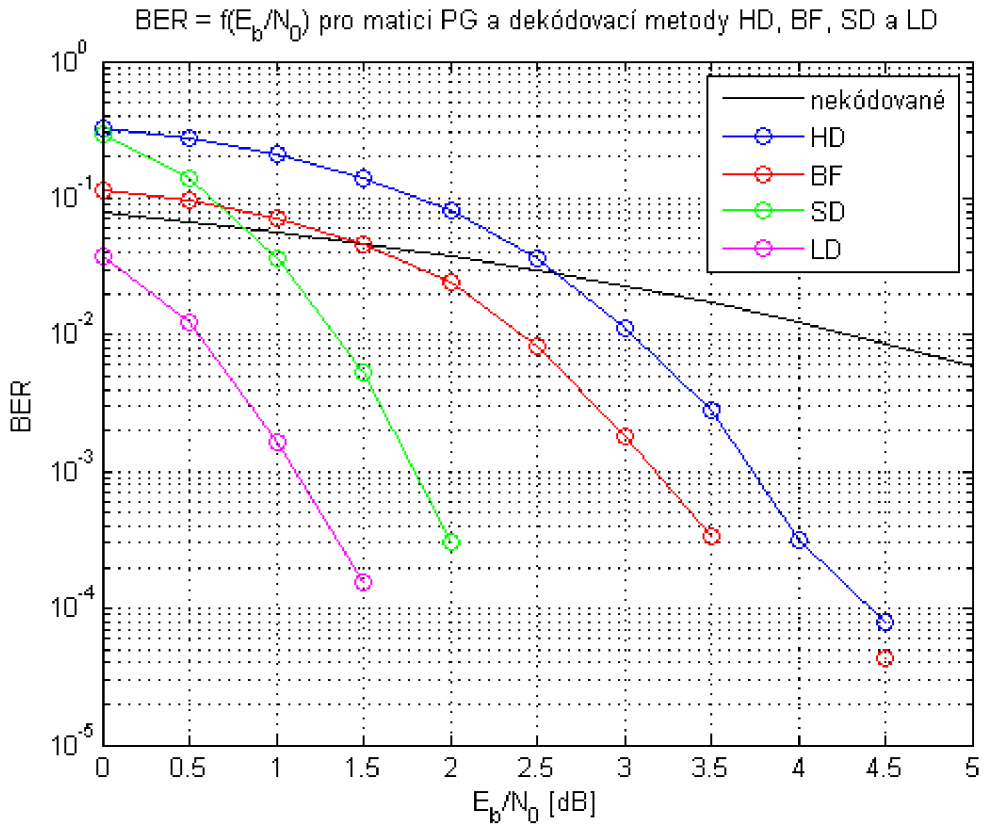
Z grafu 4.4 vyplývá úvodní pohled na dekódovací metody, kdy lze tvrdit, že za daných podmínek dosáhl nejvyššího výkonu algoritmus LD. Konfrontace algoritmu HD s ostatními algoritmy dopadla pro neiterativní algoritmus nadmíru dobře. Při nízkých hodnotách E_b/N_0 sice nedosahoval výkonu iterativních metod, ale při hodnotě $E_b/N_0 = 4$ dB již překonal jak algoritmus BF, tak i algoritmus SD. Zde se ovšem nesmí zapomínat, že se jedná stále o první iteraci dekódovacího procesu. Zatím co algoritmus HD již nemůže dosáhnout lepšího výsledku, iterativní algoritmy budou zlepšovat svůj výkon v BER každou iterací.

Pro co nejobjektivnější porovnání dekódovacích algoritmů v závislosti na první iteraci, by nyní měly následovat grafy pro porovnání obou geometrií pro zbývající stupně $s = 2, 3$ a 5. Toto by způsobilo velké množství grafů s podobnými výsledky. Proto je v práci prezentován pouze graf 4.4 jako dostačující úvodní komparace dekódovacích algoritmů LDPC kódů.

Parametry grafu 4.4 nebyly voleny zcela náhodně. Volba vytvářecí geometrie vychází ze srovnání z kapitoly 4.1, kde bylo ukázáno, že geometrie PG má o něco lepší kódové parametry něž geometrie EG a volba stupně s vychází z poznatků získaných grafem 4.3, kde byl porovnáván právě parametr s z hlediska výkonu BER pro dekódovací algoritmus BF. Volba stupně $s = 4$ se jeví jako optimální z hlediska poměru výkon/čas.

4.2.2 Vliv počtu iterací na dekódovací algoritmy

Vliv počtu iterací na dekódovací algoritmy má podstatný vliv na výkon dekódovacího algoritmu. Jako názorná ukázka je zde prezentován graf 4.5, jenž je přímým pokračováním grafu 4.4 z předcházející podkapitoly 4.2.1.

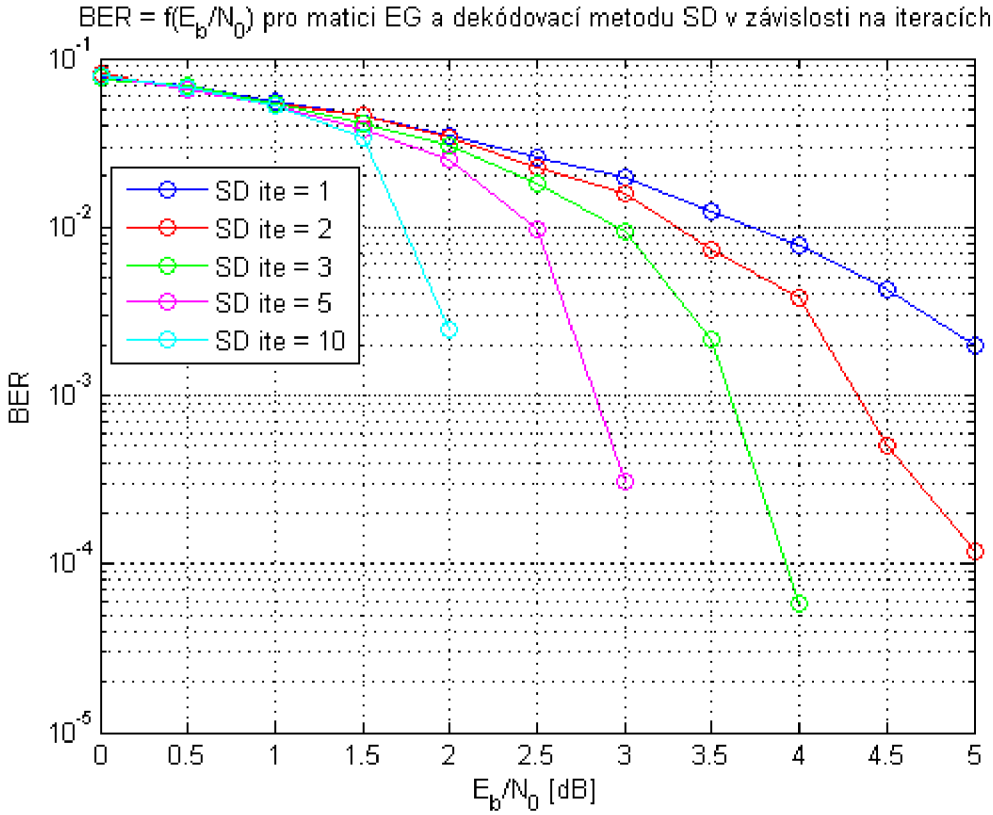


Obr. 4.5: Graf srovnání dekódovacích algoritmů v 50. iteraci.

Aby byl rozdíl mezi grafy 4.4 a 4.5 co nejmarkantnější, byl zvolen konstantní počet iterací $ite = 50$ pro všechny iterativní algoritmy. Zde byl opět nejvýkonnějším algoritmem LD, jenž dokázal na vysoce zarušeném kanále při hodnotě $E_b/N_0 = 1{,}5$ dB dosáhnout v dekódovaném slově **d** chybovost přibližně $BER = 10^{-4}$. Této hodnotě se ostatní algoritmy nedokázaly ani přiblížit. V pořadí druhý nejlepší dekódovací algoritmus SD dosáhl na dané hodnotě odstupu signálu od šumu chybovosti $BER = 10^{-26}$ a zbylé algoritmy vykazovaly ještě horší chybovost. Například chybovost ve slově **c** při zmíněném odstupu signálu od šumu je na hranici $BER = 10^{-17}$. Zisk algoritmu LD proti nedekódované posloupnosti je tedy při hodnotě $E_b/N_0 = 1{,}5$ dB přes dvě dekády v ose y (chybovosti BER).

Grafy 4.4 a 4.5 ukazují razantní zlepšení výkonu BER iterativních algoritmů zatím co algoritmus tvrdého rozhodnutí HD dostává stále podobné výsledky (ovlivněné průměrováním). Toto zlepšení je patrné především v možnosti dosáhnout bezchybného dekódování pro nižší hodnoty odstupu signálu od šumu E_b/N_0. Tento zisk se pohybuje kolem 2 dB pro algoritmy SD a LD. Pro algoritmus BF by bylo zapotřebí rozšířit osu x pro přesné odečtení hodnoty E_b/N_0, kdy je dekódování bezchybné.

Pro ilustraci funkce programu Simulace LDPC kódy, jenž umožňuje simulaci v závislosti na počtu iterací dekódovacího algoritmu, je přiložen obrázek 4.6, na



Obr. 4.6: Vliv počtu iterací na algoritmus SD.

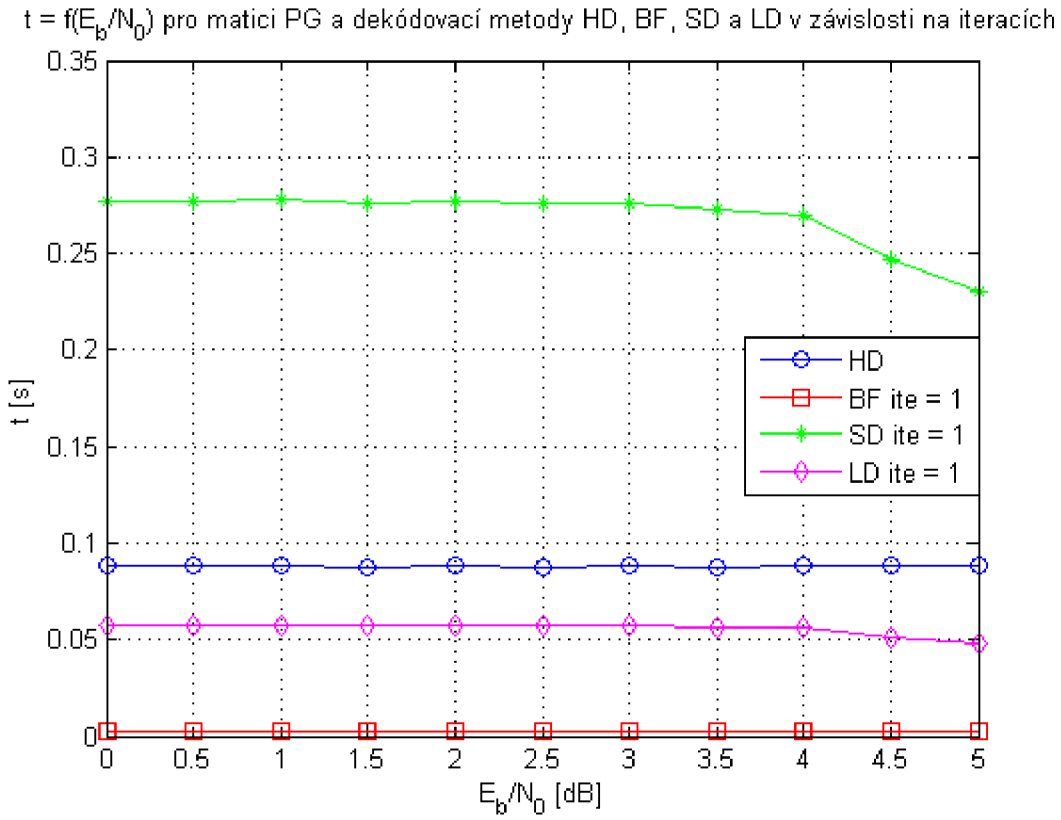
kterém je zachycen dekódovací algoritmus SD pro geometrii EG se stupněm $s = 5$. Z tohoto grafu je zřejmý výkonnostní zisk (v dB) pro každou zvyšující se iteraci. Tento zisk je ovšem vykoupen zvýšenou časovou náročností výpočtu procesu dekódování. Především dlouhé LDPC kódy, vznikající například při stupni $s = 5$, jsou po časové stránce velice citlivé na zvyšující se počet iteraci. Zde se vyskytuje problém volby optimálního počtu iterací pro daný dekódovací algoritmus při zvolené vytvářecí geometrii. Tomuto problému se věnuje následující podkapitola 4.2.3.

4.2.3 Časová náročnost dekódovacích algoritmů

Tato kapitola se věnuje komplexnímu srovnání časové náročnosti prezentovaných dekódovacích algoritmů v závislosti na typu a velikosti vytvářecí geometrie. Časová náročnost může být jeden z klíčových bodů při volbě optimálního dekódovacího algoritmu daného LDPC kódu. Zde je nutno podotknout, že dosažené časové údaje jsou závislé na kvalitě výpočetní techniky, ale jejich vypovídající hodnota je nezanedbatelná, protože umožňují jednotlivé dekódovací algoritmy porovnat.

Pro ukázku funkce programu Simulace LDPC kódy, v jehož koncepci je zakomponována funkce simulace dekódovacích algoritmů v závislosti na čase, je zde prezentován graf 4.7, pro který byla využita stejná vytvářecí geometrie jako pro grafy 4.4 a 4.5. Z tohoto grafu jsou zřetelné časy potřebné pro vykonání procesu dekódování pomocí daného dekódovacího algoritmu při konstantním počtu iterací $ite = 1$. Nejlepšího výsledku dosáhl dekódovací algoritmus BF, jemuž trval proces dekódování v řádech jednotek milisekund. Vynikajícího výsledku dosáhly také algoritmy LD a HD, u kterých bylo dekódování provedeno do $t = 0{,}1$ s. Nejhorší výsledek vykazoval algoritmus SD, jenž je ze všech v práci popisovaných dekódovacích algoritmů časově nejnáročnější.

Mírný pokles křivek SD a LD v grafu 4.7 při hodnotách odstupu signálu od šumu $E_b/N_0 = 4$ dB a více je způsoben tím, že algoritmus již nemusí opravovat příliš vysoké množství chyb v přijatém slovu. Tímto se zredukuje počet operací ve vykonávání dekódovacího procesu a konvergence k výsledku se o něco urychlí. Toto neplatí pro algoritmus HD, který neustále provádí jednokrokové dekódování, ani pro algoritmus BF, u něhož příliš neklesá bitová chybovost po vykonání první iterace, jak je patrné z grafu 4.4. Tento jev je zřetelnější při měření časové náročnosti při vyšším počtu iterací. Jako názorná ukázka poslouží obrázek 4.8, na kterém je zachycen algoritmus SD aplikovaný na vytvářecí metodu EG stupně $s = 5$. Zde lze odečíst zvyšující se časovou náročnost dekódovacího algoritmu SD s narůstajícím počtem iterací. Jednotlivé křivky v grafu mají opět klesající tendenci, kdy pro vyšší počet iterací je toto klesání strmější. Důvod je prostý a při jeho objasnění je ideální přihlédnout ke grafu 4.6, který zachycuje funkci $BER = \mathrm{f}(E_b/N_0)$ pro stejnou vy-

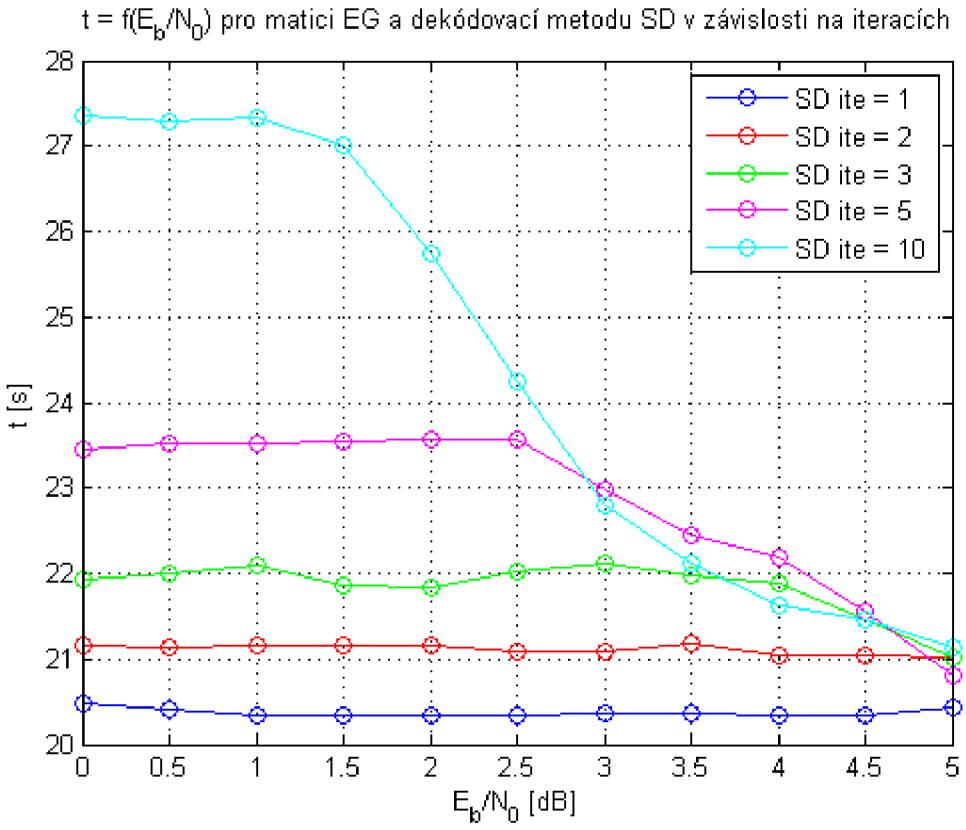


Obr. 4.7: Časová náročnost 1. iterace dekódovacích algoritmů.

tvářecí geometrii i dekódovací metodu jako graf 4.8. Výsledkem je poté zjištění, že se zvyšujícím se výkonem SD algoritmu (zvyšující se počet iterací) v závislosti chybovosti BER bude doba trvání dekódovacího procesu klesat. Při vyšších hodnotách E_b/N_0 je toto klesání zapříčiněno opravením všech chyb vzniklých přenosem přes zarušený informační kanál.

Na závěr této podkapitoly je uvedena tabulka 4.2. Tato tabulka přehledně zobrazuje časy potřebné pro vykonání procesu dekódování v příslušné vytvářecí geometrii při velikosti definované stupněm s. Tabulka je dále dělena do tří sekcí pro 1. iteraci, 10. iteraci a 50. iteraci dekódovacího procesu. Toto dělení přispívá k získání přehledu o nárůstu času dekódování se zvyšujícím se počtem iterací. Tento nárůst je viditelný především na vytvářecích geometriích od stupně $s = 4$, a proto zde opět nastává problém volby optimálního poměru výkon/čas. V tabulce jsou také zachyceny rozdíly časů pro obě vytvářecí geometrie LDPC kódů.

Z výsledků měření času jednotlivých dekódovacích algoritmů pomocí programu Matlab, vynesených do tabulky 4.2, je zřejmé, že stupeň vytvářecích matic $s = 5$ je z časového hlediska špatně aplikovatelný do praxe. Takto vytvořené LDPC kódy sice dosahují nejlepších výsledků ve funkci $BER = \mathrm{f}(E_b/N_0)$, ale čas potřebný k získání



Obr. 4.8: Časová náročnost algoritmu SD v závislosti na iteracích.

těchto výsledků je problematický. Nejhoršího výsledku zde dosáhl algoritmus SD, který při zvyšování počtu iterací snadno přesáhne hranice 30ti sekund pro vykonání dekódování. Jediným vhodným dekódovacím algoritmem pro LDPC kódy s vytvářecí geometrií stupně $s = 5$ je BF algoritmus. Tento algoritmus sice nedosahuje na výkon algoritmů SD a LD, ale jeho časová náročnost je velice pozitivní vlastností algoritmu. Při zvyšování iterací BF algoritmu se čas potřebný k provedení dekódování zvyšuje, ale i přes to stále zůstává na velice dobrých hodnotách při srovnání s ostatními prezentovanými dekódovacími algoritmy LDPC kódu.

Z pohledu praktického využití LDPC kódů se jeví nejoptimálněji stupeň $s = 4$, pro který dosahují dekódovací algoritmy solidních výsledků v BER charakteristikách a časová náročnost se pohybuje v řádech desetin sekund pro všechny dekódovací algoritmy. Proto byla většina grafů v práci zaměřena především na tento stupeň, pro získání dostatečného množství informací k získání výsledného porovnání dekódovacích algoritmů. Tuto skutečnost potvrzuje také třetí sekce tabulky 4.2, která potvrzuje, že časová náročnost pro nejsilnější dekódovací algoritmy (SD a LD) již narůstá velice pozvolna a naopak klesá se zvyšující se hodnotou odstupu signálu od šumu.

Tab. 4.2: Přehled časové náročnosti dekódovacích algoritmů.

1. iterace		EG			PG		
E_b/N_0 [db]		0	3	5	0	3	5
$s = 2$	HD [s]	0,0029	0,0018	0,0018	0,0023	0,0022	0,0022
	BF [s]	0,0008	0,0009	0,0009	0,0012	0,0012	0,0009
	SD [s]	0,0021	0,001	0,001	0,0037	0,0038	0,001
	LD [s]	0,0037	0,002	0,001	0,0032	0,0031	0,001
$s = 3$	HD [s]	0,0064	0,0064	0,0062	0,0078	0,0075	0,0077
	BF [s]	0,0015	0,0015	0,0009	0,0029	0,0018	0,0038
	SD [s]	0,0116	0,0117	0,0012	0,0162	0,0195	0,0008
	LD [s]	0,0094	0,0094	0,0012	0,0111	0,0162	0,0007
$s = 4$	HD [s]	0,0709	0,0707	0,0708	0,0899	0,0898	0,0901
	BF [s]	0,0033	0,0041	0,0064	0,0031	0,0039	0,0031
	SD [s]	0,2059	0,2072	0,2064	0,2829	0,2835	0,2850
	LD [s]	0,0566	0,0542	0,0564	0,0632	0,0613	0,0633
$s = 5$	HD [s]	6,0154	6,013	6,0157	6,6371	6,6438	6,6543
	BF [s]	0,0211	0,0177	0,0162	0,0177	0,0174	0,0213
	SD [s]	20,6045	20,6005	20,5959	22,7192	22,7483	22,7751
	LD [s]	0,4795	0,4798	0,4804	0,5163	0,5130	0,5136
10. iterace							
$s = 2$	BF [s]	0,0008	0,0012	0,0009	0,0013	0,001	0,0009
	SD [s]	0,0021	0,0027	0,001	0,0037	0,001	0,001
	LD [s]	0,0035	0,0025	0,001	0,0033	0,001	0,001
$s = 3$	BF [s]	0,0028	0,0015	0,0016	0,011	0,0037	0,0077
	SD [s]	0,0117	0,0116	0,0116	0,0251	0,0152	0,0087
	LD [s]	0,0096	0,0095	0,0096	0,0201	0,0128	0,0077
$s = 4$	BF [s]	0,0212	0,014	0,0104	0,0232	0,0158	0,0105
	SD [s]	0,4344	0,2715	0,2179	0,7225	0,2796	0,2814
	LD [s]	0,1573	0,0601	0,0595	0,568	0,066	0,0664
$s = 5$	BF [s]	0,1083	0,1088	0,1081	0,1153	0,1142	0,0488
	SD [s]	27,6719	22,9927	21,4207	30,4518	26,1428	23,6144
	LD [s]	4,289	0,9073	0,4856	4,606	0,9698	0,5135
50. iterace							
$s = 4$	BF [s]	0,0802	0,0218	0,0246	0,0834	0,0346	0,0125
	SD [s]	0,4008	0,295	0,2354	0,7046	0,3783	0,3097
	LD [s]	0,1651	0,114	0,0615	0,2984	0,1282	0,0682

5 ZÁVĚR

Obsahem této práce byla problematika LDPC kódů. V úvodní kapitole 1 se práce zabývala teoretickým základem pro popis LDPC kódů. V krátkosti byla nastíněna historie těchto kódů a poté v kapitole 1.2 jsou popsány základní vlastnosti LDPC kódů. Důležitou roli v práci dostala kapitola 1.3, jenž dopodrobna popisuje různé přístupy při tvorbě paritních matic. Největší důraz byl kladen na geometrickou konstrukci těchto matic, kdy byly popsány dvě metody – EG a PG.

Další náplní práce byla kapitola 2, která popisuje metody kódování a dekódování LDPC kódů. Za tímto účelem byly vybrány čtyři metody, které byly důkladně rozebrány v podkapitolách

- Hard-Decision algoritmus (HD) – 2.2.1,
- Bit-Flipping algoritmus (BF) – 2.2.2,
- The Sum-Product algoritmus (SD) – 2.2.3,
- Log Likelihood algoritmus (LD) – 2.2.4.

Některé algoritmy byly doplněny také příkladem výpočtu, jenž doplňuje teoretický obsah popisu dekódovacích metod. Z popisu jednotlivých algoritmů vyplývá, že nejefektivnějším je LD algoritmus, za kterým mírně zaostává SD algoritmus. Dle [11] se tyto algoritmy při vzrůstající délce kódu n dokáží přiblížit blízko k teoretické kapacitě kanálu.

Po teoretickém úvodu k problematice LDPC kódů následovala praktická část práce. Kapitola 3.1 se zabývá implementací algoritmů LDPC kódů do prostředí Matlab, ve kterém vznikaly veškeré kódy a grafické aplikace. V této části jsou popsány veškeré teoreticky popsané algoritmy ať vytváření, tak dekódování LDPC kódů. Text je doplněn vývojovými diagramy, jenž názorně doplňují textový popis algoritmů. V další části kapitoly 3 je popsán grafický výstup práce. Tím je rozuměn program LDPC kódy, jenž vznikl dle zadání diplomové práce. Program LDPC kódy se dále člení na dvě části

- Výuka LDPC kódy,
- Simulace LDPC kódy.

Dle názvů těchto dílčích grafických interface lze vydedukovat jejich význam. Jednotlivé programy jsou pak popsány z pohledu koncového uživatele, aby bylo ovládání co nejjasnější a nejjednodušší. Uživatel poté získá praktickou pomůcku pro porozumění a porovnávání LDPC kódů dle nejrůznějších nastavení jako jsou například volba vytvářecí geometrie, volba dekódovacích algoritmů nebo parametrů simulace.

V závěrečné kapitole práce, s pořadovým číslem 4, jsou analyzovány výsledky získané pomocí programu Simulace LDPC kódy. Analýza se zabývá především porovnáváním vytvářecích algoritmů LDPC kódů (EG a PG), kdy je zkoumán vliv velikosti paritní matice $\mathbf{H}$ na kódový zisk (graf 4.3). V tomto ohledu je důležité

přihlédnou také k volbě optimálního dekódovacího algoritmu a parametrů tohoto algoritmu. Pro ukázku byl zvolen algoritmus BF s parametrem $ite = 100$. Zde dosáhl nejlepšího výsledku stupeň $s = 4$, který byl dále použit pro porovnávání jednotlivých dekódovacích algoritmů. Volba vytvářecí geometrie padla na PG pro lepší kódové parametry nežli u jsou u geometrie EG.

Další obsah kapitoly 4 je zaměřen na porovnávání jednotlivých dekódovacích algoritmů, konkrétně kapitola 4.2. Zde je opět kladen důraz především na kódový zisk algoritmů, ale také na časové parametry procesu dekódování. Proto byla prezentována série grafů (4.4 – 4.8) pro co nejlepší podchycení veškerých volitelných parametrů kódu (typ vytvářecí geometrie, počet iterací, časová náročnost, apd.), které mohou ovlivnit výsledek simulace. Výsledkem komparace je zjištění, že velice vhodnou volbou je dekódovací algoritmus LD, který dosahoval nejlepšího kódového zisku a jeho časová náročnost byla také velice pozitivní.

Výsledná volba kombinace vytvářecí a dekódovací metody je přesto stále závislá na požadavcích dané realizace. Jedná se především o volbu délky vstupních dat a také na velikosti zarušení informačního kanálu. Dle těchto kritérií lze různorodě přizpůsobovat jak vytvářecí geometrii volbou stupně s, ale také dekódovací algoritmus především volbou počtu iterací. Zde je ovšem nutno přihlídnout k narůstajícímu času, který je nejrazantnější pro dlouhé LDPC kódy. Tuto skutečnost popisuje tabulka 4.2.

LDPC kódy poté přinášejí kvalitní kódovací systém jenž již našel uplatnění v některých standardech, jako jsou například IEEE 802.16e, IEEE 802.20, IEEE 802.3, IEEE 802.11n [2] nebo DBV-RS2 [10]. Nové trendy ukazují cesty využití této kódovací koncepce především pro vysokorychlostní spoje [13]. Hlavními oblastmi zájmu jsou jak vytvářecí metody, kde se za produktivní cestu předpokládá využití nerovnoměrných paritních matic, o kterých pojednává například [4], tak dekódovací algoritmy, jejichž návrh a realizace lze stále zdokonalovat a přizpůsobovat potřebné realizaci.

LITERATURA

[1] ADÁMEK, J. *Kódování*. Praha : SNTL – Nakladatelství technické literatury, 1989. 192 s.

[2] CAI, Z., TAN, P. H., SUN, S., CHIN P. S. *Efficient encoding of IEEE 802.11n LDPC codes* [online]. Electronics Letters, Prosinec 2006, Vol. 42 No. 25. Dostupný z URL: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4030685>.

[3] GALLAGER, R. G. *Low Density Parity Check Codes* [online]. 1963. 90 s. Dostupný z URL: <http://www.inference.phy.cam.ac.uk/mackay/gallager/papers/>.

[4] HE, Z., FORTIER, P., ROY, S. *A Class of Irregular LDPC Codes with Low Error Floor and Low Encoding Complexity* [online]. IEEE Communications Letters, Květen 2006, Vol. 10, No. 5. Dostupný z URL: <http://lrts.gel.ulaval.ca/publications/uploadPDF/publication_60.pdf>.

[5] LEINER, B. *LDPC Codes — a brief Tutorial* [online]. 2005. 9 s. Dostupný z URL: <http://www.ics.uci.edu/~welling/teaching/ICS279/LPCD.pdf>.

[6] LIN, S., COSTELLO, D. J. *Error Control Coding: Fundamentals and Applications, second edition*. Prentice Hall: Englewood Cliffs, NJ, 2005, 1271 s. ISBN: 0-13-042672-5.

[7] MACKAY, D. *David MacKay's Gallager code resources* [online]. Dostupný z URL: <http://www.inference.phy.cam.ac.uk/mackay/CodesFiles.html>.

[8] MARCHAND, P, . *Graphics and GUIs with MATLAB*. Boca Raton, Florida : Chapman & Hall/CRC, 2003. 523 s. ISBN 1-58488-320-0.

[9] *MathWorks – Function Reference* [online]. Dostupný z URL: <http://www.mathworks.com/help/toolbox/comm/ref/a1037894415.html>.

[10] MOON, T. *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, New Jersey: Wiley – Interscience, 2005. 800 s. ISBN-13: 978-0070010697.

[11] MOREIRA, J. C., FARREL, P. G. *Essentials of Error-Control Coding*. Chichester: John Wiley & Sons, Ltd, 2006, 388 s. ISBN: 0-470-02920-X.

[12] NĚMEC, K. *Datová komunikace*. Brno: Skriptum VUT, 2007. 172 s.

[13] SHA, J., WANG, Z., GAO, M., LI, L. *Multi–Gb/s LDPC Code Design and Implementation* [online]. IEEE Transactions on VLSI Systems, Únor 2009, Vol. 17, No. 2. Dostupný z URL: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4689312>.

[14] SWEENEY, P. *Error control coding: from theory to practice*. Chichester: Wiley – Interscience, 2002. 240 s. ISBN: 0-470-84356-X.

[15] ZAPLATÍLEK, K. *MATLAB pro začátečníky*. Praha: BEN – technická literatura, 2005. 152 s. ISBN 80-7300-175-6.

[16] ZAPLATÍLEK, K. *MATLAB: tvorba uživatelských aplikací*. Praha: BEN – technická literatura, 2004. 216 s. ISBN 80-7300-133-0.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

A Velikost amplitudy signálu

AWGN Aditivní bílý gaussovský šum

BER Bitová chybovost – Bit Error Rate

BF Přehazování bitů (dekódovací algoritmus) – Bit-flipping algorithm

BPSK Binary-Phase Shift Keying – binární klíčování

$\mathbf{c}$ Zakódovaná posloupnost informačních bitů

c_i Bitové uzly Tannerova grafu

$\mathbf{d}$ Vektor dekódovaného slova

d_{min} Hammingova minimální vzdálenost

E_b/N_0 Energie na bit v poměru s výkonem spektrálního šumu

EG Euklidovská geometrie – Euclidean geometry

f Inicializační pravděpodobnost výskytu log. 1 nebo log. 0 u algoritmu SD

$\mathbf{G}$ Generující matice

$\mathbf{g}(X)$ Generující polynom pro matici $\mathbf{G}$

GF Galoisovo těleso

h Nezáporné celé číslo pro vytváření paritních matic

$\mathbf{H}$ Paritní matice

HD Tvrdé rozhodnutí (dekódovací algoritmus) – Hard decision algorithm

$\mathbf{I}$ Jednotková matice

ite Počet iterací dekódovacího algoritmu

$\mathbf{k}$ Posloupnost informačních bitů

LD Logaritmická pravděpodobnost (dekódovací algoritmus) – Log likelihood algorithm

LDPC Řídká paritní matice – Low Density Parity Check

m Dimenze EG nebo PG kódu

n Délka LDPC kódu

o Počet jedniček na sloupec Gallagerovi paritní matice

$p(X)$ Primitivní polynom

$\mathbf{P}$ Paritní submatice

P_d Inicializační pravděpodobnost algoritmu LD

PG Projektivní geometrie – projective geometry

$\mathbf{Q}$ Matice koeficientů pro výpočet dekódování pomocí algoritmů SD a LD

r Citlivost paritní matice $\mathbf{H}$

R Informační rychlost

RAM Random-access memory – paměť s přímým přístupem

$\mathbf{R}$ Matice koeficientů pro výpočet horizontálního kroku algoritmů SD a LD

$\mathbf{rx}$ Výstupní slovo z informačního kanálu

s Stupeň LDPC kódu

$\mathbf{s}$ Vektor syndromu paritní matice

s_i Součtové uzly Tannerova grafu

$\mathbf{S}$ Vektor hodnot syndromu paritní matice pro algoritmus BF

syn Syndrom v prostředí Matlab

t Počet chyb

SD Součet součinů (dekódovací algoritmus) – The sum-product algorithm

v Počet jedniček na řádek Gallagerovi paritní matice

$\mathbf{v}$ Vytvářecí vektor paritních matic pro EG a PG

w Váha pro určení generujícího polynomu

z Proměnná pro uložení znaménka u dekódovacího algoritmu LD

α Kořen polynomu

$\alpha_{i,j}$ Koeficient používaná pro dekódování pomocí algoritmu SD

α_d Znaménko inicializační pravděpodobnosti algoritmu LD

β Vytvářecí element

β_d Absolutní část inicializační pravděpodobnosti algoritmu LD

η Koeficient pro určení bodů ve vytvářecím vektoru

γ Váha sloupců paritní matice $\mathbf{H}$

ρ Váha řádků paritní matice $\mathbf{H}$

σ Rozptyl

Soubor matematického značení

a Proměnná - malé zkosené písmeno nebo velké zkosené písmeno

a_i Značení proměnných - dolní index

a^x Logická hodnota proměnné - horní index $x \in 0, 1$

$\mathbf{a}$ Vektor - malé tučné stojaté písmeno

$\mathbf{a}(i)$ Indexování ve vektoru

$\mathbf{A}$ Matice - velké tučné stojaté písmeno

$\mathbf{A}_x$ Popis matice - dolní index (číslo nebo znaky)

$\mathbf{A}^T$ Transponovaná matice - horní index T

$\mathbf{A}(i, j)$ Indexování v matici - pořadí řádek, sloupec

$\cdot$ Klasická operace násobení

: Určení všech prvků v řádku či sloupci

$\circ$ Násobení modulo 2

$\oplus$ Součet modulo 2

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

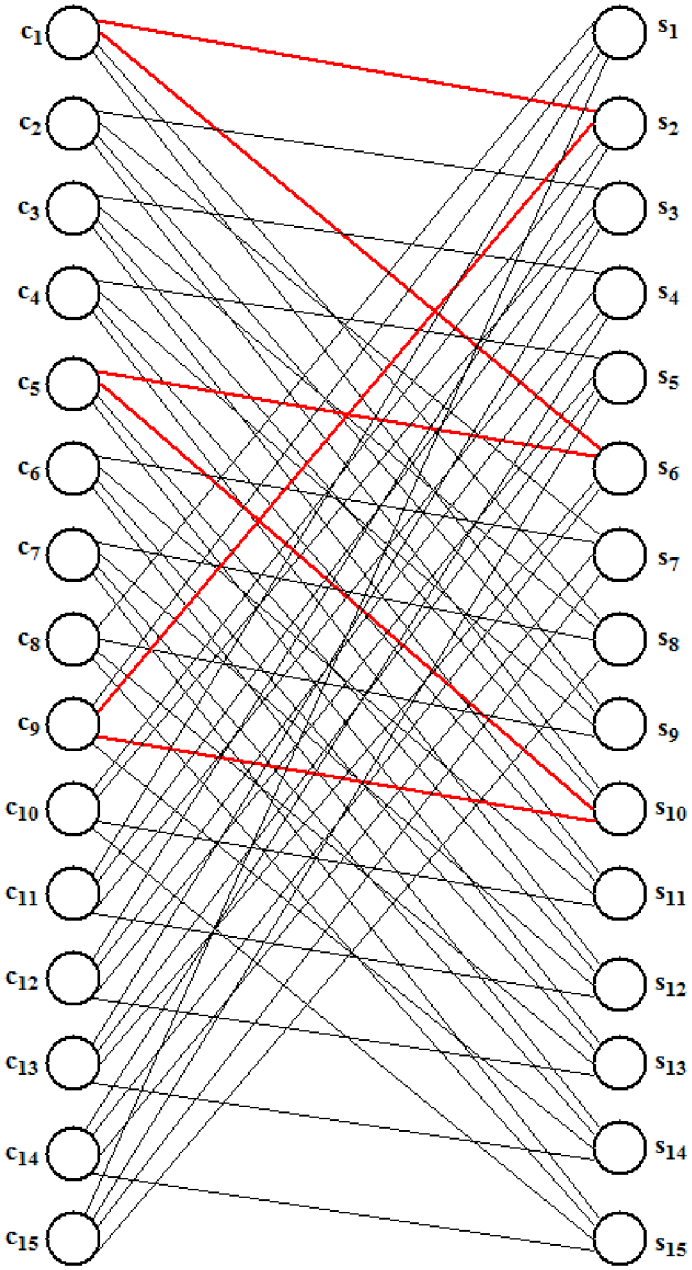
Na přiloženém CD se nachází výsledný text diplomové práce ve formátu PDF a také samostatná aplikace LDPC kódy. Pro spuštění aplikace LDPC kódy je vhodné využít prostředí MATLAB R2010a nebo samostatnou spustitelnou verzi, pro kterou je ovšem nezbytné mít nainstalovaný program MCRInstaller. Tento program je součástí CD. Dále je možno na přiloženém CD nalézt soubor REEDME.txt, jenž shromažďuje instrukce k užívání vytvořené aplikace LDPC kódy.

Adresářová struktura má následující tvar:

```
.
|- diplomka/                 %zde se nachází výsledný souboru DP.pdf
                             %s textem diplomové práce
|- latex/                    %veškeré materiály využité při tvorbě DP
                             %v LaTeXu (obrázky, pdf, a texty)
|   |- obr/
|   |- pdf/
|   |- text/
|- LDPC kódy - Matlab/       %vše potřebné pro spuštění aplikace
                             %Program LDCP kódy v Matlabu
|   |- GUI_final/
|      |- GUI_vyuka/
|      |- GUI_simulace/
|- LDPC kódy - Aplikace/     %exe verze aplikace Program LDCP kódy
|   |- src/
|   |- distrib/
'- REEDME
```

B TANNERŮV GRAF

V příloze je prezentován Tannerův graf, teoreticky popsaný v kapitole 1.4, paritní matice $\mathbf{H}_{EG}$ sestrojené v příkladu v kapitole 1.3.3. Tato matice je dále využita v příkladech numerických výpočtů dekódovacích metod HD a SD.



Obr. B.1: Tannerův graf matice $\mathbf{H}_{EG}$.

C NUMERICKÝ VÝPOČET ALGORITMU SD

Obsah této přílohy je tvořen tabulkami, jenž z důvodu přehlednosti nebyly zařazeny do kapitoly 2.2.3, která je věnována dekódovací metodě SD. Popis k těmto tabulkám je předmětem zmíněné kapitoly.

C.1 Inicializace SD algoritmu

Přiloženy tabulky C.1 a C.2 pro kontrolu výpočtů inicializačních parametrů algoritmu SD.

Tab. C.1: Výpočet parametru Q_{ij}^0 SD algoritmu.

	1	2	3	4	5	6	7	...	15
1								...	0,3109
2	0,0812							...	
3		0,9660						...	
4			0,2367					...	
5				0,0010				...	0,3109
6	0,0812				0,2753			...	
7		0,9660				0,0075		...	0,3109
8	0,0812		0,2367				0,9983	...	0,3109
9	0,0812	0,9660		0,0010				...	
10		0,9660	0,2367		0,2753			...	
11			0,2367	0,0010		0,0075		...	
12				0,0010	0,2753		0,9983	...	
13					0,2753	0,0075		...	
14						0,0075	0,9983	...	
15							0,9983	...	

C.2 Horizontální krok SD algoritmu

Tabulky C.3 a C.4 ukazující výpočtů koeficientů R_{ij}^0 a R_{ij}^1 slouží pro kontrolu numerického výpočtu při demonstraci horizontálního kroku z příkladu prezentovaného v kapitole 2.2.3.

Tab. C.2: Výpočet parametru Q_{ij}^1 SD algoritmu.

	1	2	3	4	5	6	7	...	15
1								...	0,6891
2	0,9188							...	
3		0,0340						...	
4			0,7633					...	
5				0,9990				...	0,6891
6	0,9188				0,7247			...	
7		0,0340				0,9925		...	0,6891
8	0,9188		0,7633				0,0017	...	0,6891
9	0,9188	0,0340		0,9990				...	
10		0,0340	0,7633		0,7247			...	
11			0,7633	0,9990		0,9925		...	
12				0,9990	0,7247		0,0017	...	
13					0,7247	0,9925		...	
14						0,9925	0,0017	...	
15							0,0017	...	

Tab. C.3: Výpočet parametru R_{ij}^0 SD algoritmu.

	1	2	3	4	5	6	7	...	15
1								...	0,6492
2	0,7171							...	
3		0,7647						...	
4			0,8909					...	
5				0,3523				...	0,1102
6	0,6416				0,7640			...	
7		0,6483				0,3597		...	0,1345
8	0,5992		0,6578				0,4166	...	0,7198
9	0,2865	0,6918		0,3209				...	
10		0,4226	0,6369		0,6605			...	
11			0,8306	0,6744		0,6768		...	
12				0,7227	0,9946		0,2770	...	
13					0,7231	0,6018		...	
14						0,7579	0,2451	...	
15							0,6229	...	

Tab. C.4: Výpočet parametru R_{ij}^1 SD algoritmu.

	1	2	3	4	5	6	7	...	15
1								...	0,3508
2	0,2829							...	
3		0,2353						...	
4			0,1091					...	
5				0,6477				...	0,8898
6	0,3584				0,2360			...	
7		0,3517				0,6403		...	0,8655
8	0,4008		0,3422				0,5834	...	0,2802
9	0,7135	0,3082		0,6791				...	
10		0,5774	0,3631		0,3395			...	
11			0,1694	0,3256		0,3232		...	
12				0,2773	0,0054		0,7230	...	
13					0,2769	0,3982		...	
14						0,2421	0,7549	...	
15							0,3771	...	