Fakulta elektrotechniky a informatiky
Slovenská technická univerzita v Bratislave

Ing. Tomáš Janvars

Autoreferát dizertačnej práce

Príspevok k metódam dekódovania samoopravných kódov

na získanie akademického titulu: PhD.

v doktorandskom študijnom programe: 5.2.15 Telekomunikácie

v študijnom odbore: Telekomunikácie

Forma štúdia: externá forma

Miesto a dátum: 02.07.2018,

Ústav multimediálnych informačných a komunikačných technológií,
FEI STU, Ilkovičova 3, 812 19, Bratislava

STU

Dizertačná práca bola vypracovaná na: Ústav multimediálnych informačných a komunikačných technológií, FEI STU, Ilkovičova 3, 812 19, Bratislava

Predkladateľ: Ing. Tomáš Janvars
Ústav multimediálnych informačných a komunikačných technológií
Fakulta elektrotechniky a informatiky
Ilkovičova 3, 812 19, Bratislava

Školiteľ: prof. Ing. Peter Farkaš, DrSc.
Ústav multimediálnych informačných a komunikačných technológií
Fakulta elektrotechniky a informatiky STU
Ilkovičova 3, 812 19, Bratislava

Oponenti: prof. RNDr. Frank Schindler, PhD.
Ústav aplikovanej informatiky,
Fakulta Informatiky, Paneurópska vysoká škola

Ing. Stanislav Dlháň PhD.
Solution Architect,
Accenture Slovensko.

Autoreferát bol rozoslaný: 13.06.2018

Obhajoba dizertačnej práce sa bude konať dňa: 02.07.2018 **o** 10:00 **h na** Ústave multimediálnych informačných a komunikačných technológií, FEI STU, Ilkovičova 3, 812 09, Bratislava

prof. Dr. Ing. Miloš Oravec
dekan fakulty FEI STU,

STU

Obsah

1. Úvod

Vznik teórie informácií sa datuje ku koncu 40. rokov minulého storočia, kedy americký vedec Claude E. Shannon publikoval prácu, v ktorej sa zaoberal rôznymi aspektmi komunikačného kanála. Prakticky od čias vzniku teórie informácií sa začala skúmať a rozvíjať problematika okolo samoopravných kódov. Prvú triedu samoopravných kódov opísal a navrhol Hamming už v roku 1950. Boli to lineárne blokové kódy. V priebehu ďalších rokov sa ďalej rozvíjala teória okolo lineárnych blokových kódov. K výraznejšiemu posunu vpred a zároveň k zovšeobecneniu prišlo až koncom päťdesiatych rokov minulého storočia, kedy Bose, Ray-Chaudhuri a Hocquenghem objavili novú triedu lineárnych blokových kódov, opravujúcich viacnásobné chyby. V roku 1960 Reed a Solomon objavili podobnú triedu kódov pre nebinárne symbolové abecedy. Objav BCH a Reed-Solomonových kódov priniesol nové problémy a komplikácie pri realizácií kodérov a hlavne dekodérov, čo neskôr viedlo k objavom dôležitých algoritmov. Celkom inou cestou sa uberal už v 50. rokoch výskum neblokových kódov s nedefinovanou dĺžkou kódového slova, ktoré si vyžadovali sekvenčné dekódovanie. Keďže tieto kódy bolo možné reprezentovať pomocou stromu, boli nazvané stromové kódy. Ako najpožívanejšie stromové kódy sa ukázali konvolučné kódy, ktorých popularita ešte vzrástla po vynájdení Viterbiho algoritmu (1967).

2. Súčasný stav problematiky

Prelom v teórií kódovania nastal v roku 1993, keď *Berrou et al.* uviedli novú triedu kódov, ktorú nazvali turbo kódy [BER93][BER96]. Ich práca bola venovaná iteratívnemu dekódovaniu dvoch paralelne zreťazených rekurzívnych systematických konvolučných kódov. Na dekódovanie kódov komponentov bol použitý tzv. *Maximum a posteriori* (MAP), ktorý bol založený na algoritme BCJR. Algoritmus BCJR je optimálnym pre dekódovanie samoopravných kódov, avšak jeho komplexita zamedzuje jeho využitie v aplikáciách v reálnom čase. *Berrou*-ov algoritmus zachoval ideu ale zjednodušil komplexitu za len veľmi malé straty vo výkonnosti a aplikoval algoritmus na zreťazené konvolučné kódy. Tento algoritmus pre každý dekódovaný symbol dával ako výstup tzv. *Log-likelihood ratio* (LLR) hodnotu. LLR hodnota udávala iba akýsi pomer pravdepodobnosti vyslania jednotlivých symbolov. Algoritmus MAP potom umožňoval iteratívne dekódovanie kaskádovým zapojením viacerých dekodérov pre použité kódy komponenty. Výsledky dosiahnuté novou kódovacou schémou boli pozoruhodné. Zvyškovú chybovosť BER 10^{-5} pri použití kódu s rýchlosťou ½ bolo možné dosiahnuť už pri odstupe signálu-šum (E_B/N_0) na úrovni 0.7 dB, čo predstavuje odstup asi 0.5dB od Shannonovej hranice pre kódy s kódovou rýchlosťou ½.

Hlavným konkurentom Turbo kódov vo výkonnosti sú dnes LDPC kódy. LDPC kódy boli pôvodne vynájdené Galagerom [GAL62] ale viac ako 30 rokov ignorované. Znovu objavené boli v [MAC96] a dnes pre vyššie kódové rýchlosti dosahujú lepšie výsledky ako Turbo kódy.

Podobne ako je možné vytvárať konvolučné turbo kódy (CTC) spojením viacerých konvolučných kódov, je možné vytvárať aj blokové turbo kódy (BTC), resp. kódy násobky (Turbo Product Codes) a to viacnásobným použitím, resp. zreťazením lineárnych blokových kódov na tie isté informačné bity. Zreťazené kódy boli prvý krát predstavené v [ELI54] a neskôr analyzované v [FOR66]. Keďže sa nevyužívala technika iteratívneho dekódovania SISO dekodéra ale len tvrdé dekódovanie, zreťazené kódy vykazovali slabú výkonnosť. Zreťazené blokové kódy vzbudili opäť pozornosť po uvedení a aplikovaní SISO (soft-input soft-output) dekódovacích algoritmov [HAG96][PYN94]. Určitou variáciou a kompromisom je hybridné dekódovanie, kedy sa využívajú prednosti oboch metód dekódovania, výkonnosť mäkkého a rýchlosť a zjednodušená komplexnosť tvrdého dekódovania [ZHI12] [JAN15].

Iteratívne dekódovanie, či už turbo kódov, blokových kódov alebo LDPC kódov predstavuje najrozvojovejšiu oblasť teórií informácií, čo súvisí s dnešnými praktickými nárokmi na komunikáciu.

3. Ciele dizertačnej práce

Ako napovedá zadanie a samotný obsah tejto práce, v období doktorandského štúdia som sa venoval problematike iteratívneho dekódovania blokových kódov násobkov. Na základe štúdia literatúry použitej v tejto práci a doterajších skúseností s problematikou iteratívneho dekódovania blokových kódov násobkov som si vytýčil pár oblastí, úloh a cieľov, ktorých dosiahnutím, by som chcel prispieť k vedeckému pokroku v danej oblasti. Vytýčené ciele priamo nadväzujú na zadanie dizertačnej práce. Spomínané ciele je možné zhrnúť do nasledovných bodov:

- **zovšeobecniť a nájsť nové konštrukcie kódov**,

  Tento cieľ znamená vytvoriť nové typy kódov, resp. zgeneralizovať už existujúce triedy kódov, tak aby sa pre tieto triedy kódov dali použiť iteratívne dekódovacie algoritmy. Podarilo sa nájsť konštrukcie kódov násobkov do 5D a 6D rozmerov. Zovšeobecnené konštrukcie kódov sú popísané v kapitole.

- **nájsť modifikácie známych dekódovacích algoritmov**,

  Pre nájdenie modifikácií algoritmov bolo potrebné najskôr naštudovať existujúce „state of the art“ algoritmy. Tento cieľ bol prakticky najdôležitejší v oblasti môjho výskumu, sú mu venované kapitoly 5 a 6. Pre algoritmus mäkkého iteratívneho dekódovania kódov

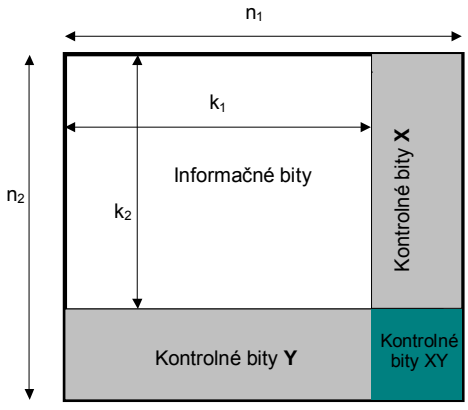
násobkov zložených z jednoduchých paritných kódov som našiel zovšeobecnenie a jeho aplikovanie pre viacrozmerné kódy [JAN03b]. Pre iteratívny algoritmus blokových kódov násobkov, navrhnutý v [PYN98], ktorý dnes som svojimi obmenami prakticky predstavuje najpoužívanejší algoritmus pre iteratívne dekódovanie blokových kódov násobkov som navrhol modifikáciu pre kódy násobky zložené z jednoduchých paritných kódov [JAN04] a optimalizáciu algoritmu pre akýkoľvek použitý kód komponent [JAN05]. V rámci tohto cieľa sa mi podarilo navrhnúť hybridný algoritmus iteratívneho dekódovania [JAN15], ktorý bol citovaný v článku IEEE COMMUNICATIONS SURVEYS & TUTORIALS zhrňujúcom najväčšie prínosy v oblasti algoritmov dekódovania blokových kódov násobkov [MUK16].

- **navrhnúť možné aplikácie viacrozmerných blokových kódov násobkov**.

Často býva najkomplikovanejšou úlohou navrhnúť reálnu aplikáciu pre uvedené teoretické postupy a výsledky. V rámci tejto práce sa nám podarilo navrhnúť uplatnenie samoopravných kódov násobkov ako kódov s obmedzenými dĺžkami behov – RLL kódy. Konštrukcia RLL kódov z kódov násobkov je popísaná v kapitole 8 dizertačnej práce.

4. Blokové kódy násobky

Blokové kódy násobky, ktoré vznikajú zreťazením blokových kódov predstavujú alternatívu ku klasickými turbo kódov, ktoré sú tvorené zreťazením konvolučných kódov. Blokové kódy násobky sú robustná trieda kódov s výbornými korekčnými vlastnosťami širokým praktickým využitím v oblasti mobilnej komunikácie. Najjednoduchší kód násobok vytvorený sériovým zreťazením je vyobrazený na obrázku 4.1.



Obrázok 4.1 Sériovo zreťazený 2D kód : $S_{2D}=C_1 \otimes C_2$

Na obr. 4.1 je zobrazené sériové zreťazenie dvoch systematických lineárnych blokových kódov C_1 a C_2 s parametrami (n_1, k_1, d_1) a $(n_2 k_2, d_2)$. Parametre výsledného kódu násobku S_{2D} sú nasledovné [PET61] :

$$n^{2D} = n_1 \times n_2 \qquad (4.1)$$

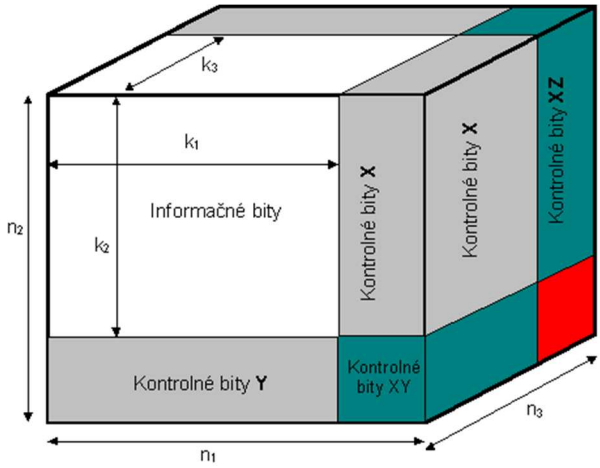
$$k^{2D} = k_1 \times k_2 \qquad (4.2)$$

$$d^{2D} = d_1 \times d_2 \qquad (4.3)$$

Výsledná kódová rýchlosť sériovo zreťazeného dvojrozmerného kódu je daná vzťahom:

$$R^{2D} = R_1 \times R_2 \qquad (4.4)$$

Pridaním ďalšieho rozmeru vytvoríme trojrozmerný kód, ktorý je pre názornosť uvedený na obr. 4.2.



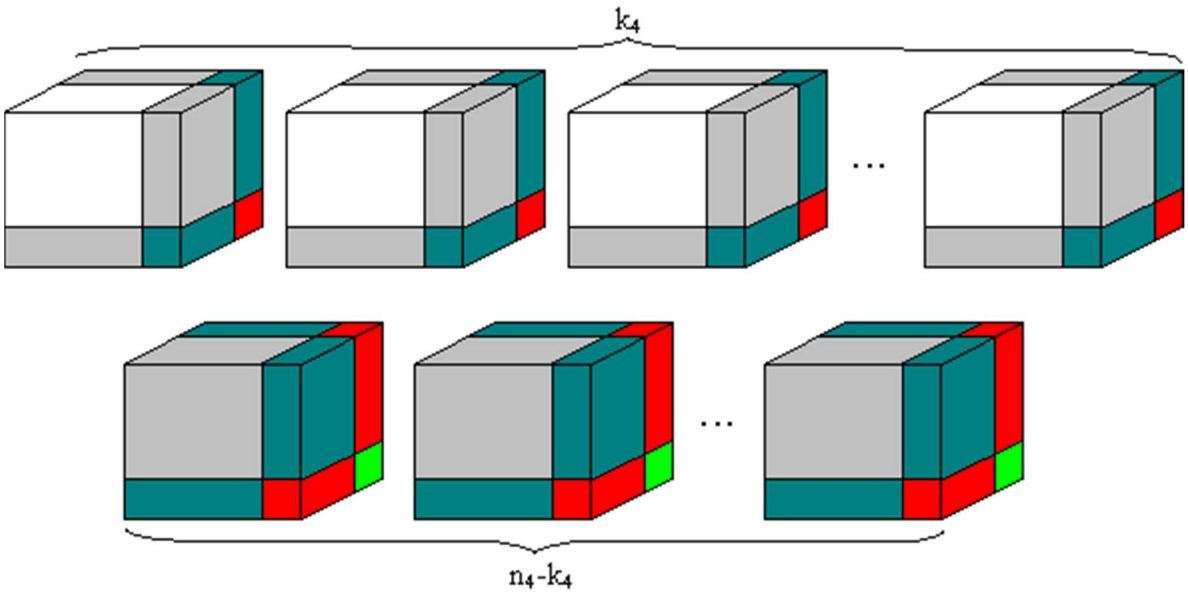
Obrázok 4.2 Sériovo zreťazený 3D kód : $S_{3D}=C_1 \otimes C_2 \otimes C_3$

Pridávaním ďalších rozmerov vieme navyšovať dimenziu kódov násobkov. Pre výskum v rámci tejto práce som navyšoval dimenziu až do rozmeru 6. Z simulácií uvedených neskôr je vidieť, že korekčné vlastnosti blokových kódov násobkov s dimenziou kódu narastajú. Pre lepšiu názornosť uvádzam postup vytvorenia štvorrozmerného sériového kódu. Ten je uvedený na obrázku č. 4.3. Zakódovanie 4D TPC vyžaduje, ako sa dá predpokladať, štyri takty. V krokoch 1,2,3 sa vytvorí paralelne k4 „kociek“ aplikovaním kódov C_1, C_2, C_3 a v 4. kroku aplikovaním kódu C4 vytvoríme zvyšných n4 - k4 kociek. Parametre, t.j. dĺžka slova, počet informačných symbolov a kódová rýchlosť, N rozmerného kódu sú dané nasledovne:

$$n^{ND} = \prod_{i=1}^{N} n_i \qquad (4.5)$$

$$d^{ND} = \prod_{i=1}^{N} d_i \qquad (4.6)$$

$$R^{ND} = \prod_{i=1}^{N} R_i = \prod_{i=1}^{N} \frac{k_i}{n_i} \qquad (4.7)$$



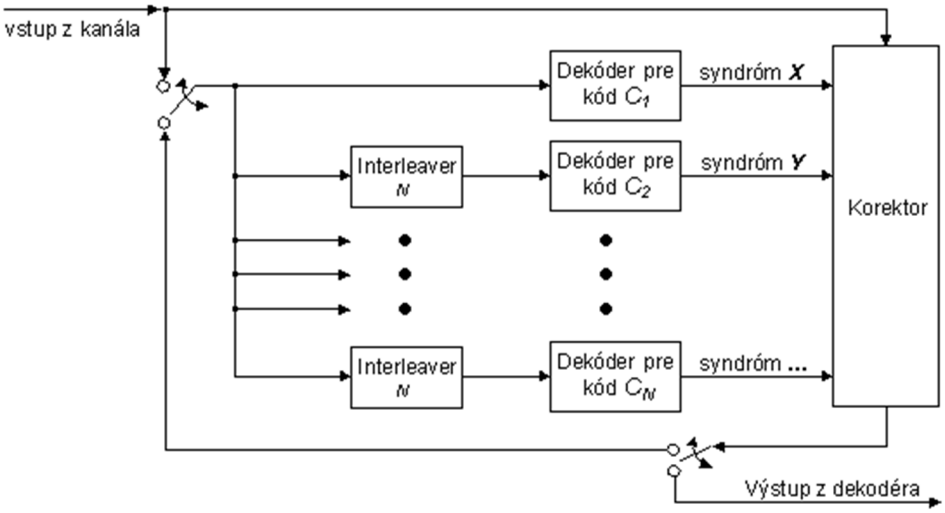
Obrázok 4.3 Sériovo zreťazený 4D kód : $S_{4D}=C_1 \otimes C_2 \otimes C_3 \otimes C_4$

5. Iteratívne dekódovanie kódov zložených z paritných kódov

5.1 Tvrdé dekódovanie kódov násobkov zložených z paritných kódov

Logická schéma dekodéra je na obrázku č. 5.1. Dekodér pracuje s tvrdými hodnotami, na výstupe rozhodovacieho bloku detektora. Ak bol v kodéri použitý interleaver pre kanál, treba na vstup dekodéra zaradiť deinterleaver, ktorý bude vykonávať reverzný proces. Bity na vstupe samotného dekodéra sú usporiadané v smere *X N*-rozmernej kocky. Tieto bity vstupujú do interleaverov, ktoré sú predradené jednotlivým dekodérom a taktiež do korektora. Dekodéry pre jednotlivé kódy komponenty vykonávajú kontrolu parity. Na vstupe dekodérov je vždy prítomných n_i bitov, nad ktorými je vykonaná operácia XOR. Operácia XOR pre jednoduchý paritný kód, nahrádza násobenie kódového slova maticou H^T. Výsledkom operácie XOR je syndróm dĺžky 1, ktorý budem nazývať elementárny syndróm. Ak má elementárny syndróm hodnotu 0, prijaté slovo je kódové slovo, ak je jeho hodnota 1, znamená to narušenie parity v kódovom slove. Elementárny syndróm je následne vyslaný do korektora, aby bola kontrola parity v každom dekodéri aplikovaná na správnu kombináciu

bitov, musia mať dekodéry C_2,... C_N predradené interleavery, ktoré vysielajú uložené bity v správnom poradí.



Obrázok 5.1 Schéma dekodéra pre paralelne zreťazený SP TPC

Množinu všetkých elementárnych syndrómov na výstupe dekodéra C_1 označíme syndróm *X*, na výstupe dekodéra C_2 syndróm *Y*, atď. Čo však predstavuje syndróm *X* ? Keď si uvedomíme, že každý riadok *N*-rozmerného „sériového“ kódu je reprezentovaný jedným elementárnym syndrómom, tak syndróm *X* ako celok sa dá logicky usporiadať do tvaru *(N-1)* -rozmernej kocky so stranami n_2 ,n_3 ,...n_N . Syndróm *Y* je taktiež *(N-1)* -rozmerná kocka, so stranami n_1,n_3,...n_N, ktorá z pôvodného kódového slova vznikla nahradením stĺpcov, atď. Pre „paralelný“ kód násobok majú syndrómy pre jednotlivé smery iné rozmery. To je spôsobené tým, že pri paralelnom zreťazení kódové slovo neobsahuje kontrolné symboly všetkých úrovní, ale len prvej úrovne. Preto pri výpočte syndrómu *X*, nemôžeme počítať paritu v posledných riadkoch jednotlivých stien a takisto v poslednej stene, resp. v poslednej kocke kódu, podľa rozmeru kódu. Syndróm *X* má v tom prípade rozmery n_2-1,n_3 -1,...n_N -1. Podobne možno ukázať, že syndróm *Y* má rozmery n_1-1,n_3 -1,...n_N -1. Pochopiteľne, interleavery v jednotlivých vetvách nebudú vysielať tie riadky, stĺpce atď, ktoré nie sú kompletné.

Korektor v každej iterácií algoritmu zhromaždí všetky syndrómy a na ich základe opravuje všetky bity nasledovným spôsobom. Ak súčet elementárnych syndrómov pre opravovaný bit je rovný, alebo presiahne tzv. prahovú hodnotu iterácie, korektor daný bit invertuje, inak bitu ponechá jeho pôvodnú hodnotu. Prahová hodnota teda určuje, v minimálne koľkých osiach kódu násobku musí byť porušená parita, aby došlo ku invertovaniu symbolu. Po preskúmaní všetkých bitov sú tieto bity opäť rozposlané do dekodérov a celý úkon sa v ďalšej iterácii opakuje. Stanovenie prahovej hodnoty je veľmi citlivá otázka. Po krátkom zamyslení

čitateľ určite zistí, že vysoké prahové hodnoty dovoľujú invertovať menej symbolov, a tým aj odhaliť menej chýb. Menšie prahové hodnoty dovolia invertovať viac symbolov, avšak sú náchylnejšie na vnášanie ďalších chýb, ktoré vzniknú invertovaním správneho bitu na nesprávny. Preto je počas dekódovania je nevyhnutné meniť prahovú hodnotu [HAT01].

Pri väčšej chybovosti kanála treba použiť vyššie prahové hodnoty, aby sa chybné symboly „nerozmnožili“ v prijatom slove. Naopak pri nižšej chybovosti kanálu, kedy v prijatom slove nie je toľko chybných symbolov a viac parít je správnych, treba použiť nižšie prahové hodnoty aby sa umocnilo opravovanie symbolov. Funkčnosť algoritmu je popísaná na nasledujúcom príklade.

5.2 Mäkké iteratívne dekódovanie kódov násobkov zložených z paritných kódov

V tejto podkapitole stručne opíšem *MAP* algoritmus navrhnutý pre TPC opísané v kapitole 4. Opäť budem uvažovať s prenosom binárnych symbolov cez AWGN kanál s použitím BPSK modulácie. Algoritmus vychádza zo uvedeného v [HAG96] Pre jednoduchosť budeme uvažovať, že energia signálov *E* bude jednotková. Funkcie hustoty podmienených pravdepodobností prijatých signálov za predpokladu vyslaných bitov pre AWGN kanál poznáme, sú zobrazené na obr. 2.3. My by sme však pre dekódovanie potrebovali poznať podmienené pravdepodobnosti bitov za predpokladu prijatých signálov $P(d = i / z)$. Tie vyjadríme použitím Bayesovho vzorca.

$$P(d = i / z) = \frac{p(z / d = i) \times P(d = i)}{\sum_{i=1}^{M} p(z / d = i) \times P(d = i)} \qquad i = 1,\ldots\ldots,M \qquad (5.1)$$

Počet symbolov v našom prípade : M=2. MAP kritérium dekóduje ten vyslaný symbol, ktorý má hodnotu $P(d = i / z)$ najväčšiu. Pre 2 symboly je rozhodovanie dané vzťahmi 5.2, 5.3.

$$P(d = 1 / z) \underset{H_2}{\overset{H_1}{\gtrless}} P(d = 0 / z) \qquad (5.2)$$

$$\frac{p(z / d = 1) \times P(d = 1)}{p(z / d = 0) \times P(d = 0)} \underset{H_2}{\overset{H_1}{\gtrless}} 1 \qquad (5.3)$$

pričom platí $p(z / d = 1) = p(z / s_1)$ a $p(z / d = 0) = p(z / s_2)$, čo je dôsledok zvoleného mapovania bitov na signály.

Log-likelihood ratio

Logaritmovaním vzťahu 5.3 dostávame užitočnú metriku tzv. Log-likelihood ratio – LLR [SKL01]. LLR je reálne číslo reprezentujúce mäkkú hodnotu na výstupe detektora, vyjadrenú vzťahmi 5.4 až 5.6:

$$L(d/z) = \log_e\left[\frac{P(d=1/z)}{P(d=0/z)}\right] = \log_e\left[\frac{p(z/d=1).P(d=1)}{p(z/d=0).P(d=0)}\right] \quad (5.4)$$

$$L(d/z) = \log_e\left[\frac{p(z/d=1)}{p(z/d=0)}\right] + \log_e\left[\frac{P(d=1)}{P(d=0)}\right] \quad (5.5)$$

$$L(d/z) = L(z/d) + L(d) \quad (5.6)$$

pričom *L(d)* tzv. je *apriórna* LLR, ktorá zohľadňuje pravdepodobnosti vyslania jednotlivých symbolov. Ak je vyslanie bitov 0 a 1 rovnako pravdepodobné, tak tento člen je rovný nule. *L(z/d)* je LLR získaná na základe meraní z kanála, ďalej označovaná ako $L_C(z)$ (channel). Za predpokladu prenosu cez AWGN kanál možno pre prijatý signál z_k, $L_C(z)$ vyjadriť nasledovne:

$$L_C(z_k) = \log_e\left[\frac{p(z_k/d_k=1)}{p(z_k/d_k=0)}\right] = \log_e\left(\frac{\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{z_k-\sqrt{E}}{\sigma}\right)^2\right]}{\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{z_k+\sqrt{E}}{\sigma}\right)^2\right]}\right) = \frac{2\sqrt{E}}{\sigma^2}z_k \quad (5.7)$$

Pre potreby dekódovacieho algoritmu potrebujem vyjadriť pravdepodobnosti jednotlivých signálov ako funkcie apriórnej *L(d)*. Zo vzťahov 5.5 a 5.6 platí :

$$L(d) = \log_e\left[\frac{P(d=1)}{P(d=0)}\right] = \log_e\left[\frac{P(d=1)}{1-P(d=1)}\right] \quad (5.8)$$

Po umocnení rovnice 5.8 a osamostatnení *P(d=1)* dostaneme :

$$P(d=1) = \frac{e^{L(d)}}{1+e^{L(d)}} \quad (5.9)$$

$$P(d=0) = 1 - P(d=0) = \frac{1}{1+e^{L(d)}} \quad (5.10)$$

Výpočet extrinsickej hodnoty

Pri odvodzovaní doterajších vzťahov sme na strane prijímača uvažovali len s detektorom. Ak zahrnieme aj dekodér, vzťah 5.3 sa zmení nasledovne. Pre jednoduchosť budem ďalej namiesto $L(d/z)$ písať $L(\hat{d})$:

$$L(\hat{d}) = L_C(z) + L(d) + L_e(\hat{d}) \qquad (5.11)$$

kde $L_e(d)$ je tzv. extrinsická LLR hodnota [HAG96][SKL01]. $L_e(d)$ je dodatočná informácia o bite *d*, ktorá je získaná z procesu dekódovania na základe použitého samoopravného kódu. Výpočet extrinsickej hodnoty je špecifický pre použitý kód a tak isto pre dekódovací algoritmus.

Bez ďalšieho dokazovania uvediem, že vo všeobecnosti pre ľubovoľnú dĺžku slova *m* platí vzťah :

$$L(d_j) = \log_e \frac{\prod_{i=1,i\neq j}^{m}\left(1+e^{L(d_i)}\right) - \prod_{i=1,i\neq j}^{m}\left(1-e^{L(d_i)}\right)}{\prod_{i=1,i\neq j}^{m}\left(1+e^{L(d_i)}\right) + \prod_{i=1,i\neq j}^{m}\left(1-e^{L(d_i)}\right)} = L_e(d_j) \qquad (5.12)$$

LLR hodnotu, ktorú sme vypočítali v (5.21) je zároveň extrinsická LLR hodnota pre bit d_j, získaná aplikovaním paritného kódu [HAG96].

Iteratívne dekódovanie

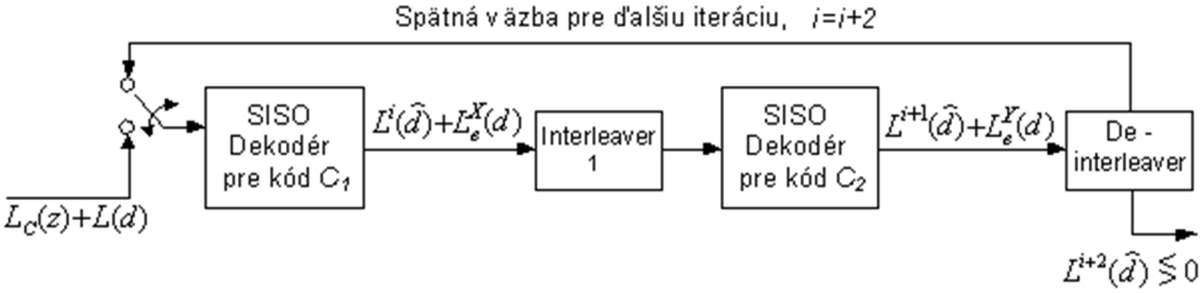
Keď vieme ako vypočítať extrinsickú hodnotu pre použitý paritný kód, môžeme začať dekódovanie. Vychádzame zo vzťahu 5.11. Ak sú pravdepodobnosti vysielania bitov 0 a 1 rovnaké, tak *apriórna LLR L(d)* je pred samotným dekódovaním nulová. Ak tomu tak nie je *apriórna LLR* je vypočítaná pre informačné symboly pomocou vzťahu 5.8. Pre paritné bity je *apriórna LLR* nulová vždy. Pre názornosť uvažujme len s 2D paritným kódom. Algoritmus iteratívneho dekódovania je pre každý bit nasledovný:

1. nastav $L(\hat{d}) = L_C(z) + L(d)$ a *i=0*.
2. vypočítaj extrinsickú hodnotu $L_e{}^X(d)$ aplikovaním paritného kódu C_1 (v smere X), t.j. dosadením $L(\hat{d})$ ostatných bitov vo väzbe do vzťahu 5.12.
3. nastav novú apriórnu LLR hodnotu pre bit : $L^{i+1}(\hat{d}) = L^i(\hat{d}) + L_e{}^X(d)$
4. vypočítaj extrinsickú hodnotu $L_e{}^Y(d)$ aplikovaním paritného kódu C_2 (v smere Y), t.j. dosadením $L(d)$ ostatných bitov vo väzbe do vzťahu 5.12.

5. nastav novú apriórnu LLR hodnotu pre bit : $L^{i+2}(\hat{d}) = L^{i+1}(\hat{d}) + L_e{}^{Y}(d)$
6. ak je počet iterácií dostatočný, choď na bod 7, inak *i=i+2* a choď na bod 2.
7. dekóduj bit *d* ako logická nula ak $L^{i+2}(\hat{d}) < 0$, inak dekóduj bit ako logická 1.

V každej iterácií sa využívajú všetky väzby, v ktorých sa bit nachádza, t.j. počítame extrinsické hodnoty pre všetky smery. Dekódovanie 3D kódu by vyžadovalo počítať extrinsickú hodnotu aj pre kód C_3 (v smere Z), atď.

Na obr. 5.2 je načrtnutá schéma *Soft-In / Soft-Out* iteratívneho dekodéra, kopírujúca iteratívny algoritmus dekódovania pre 2D paritný kód.

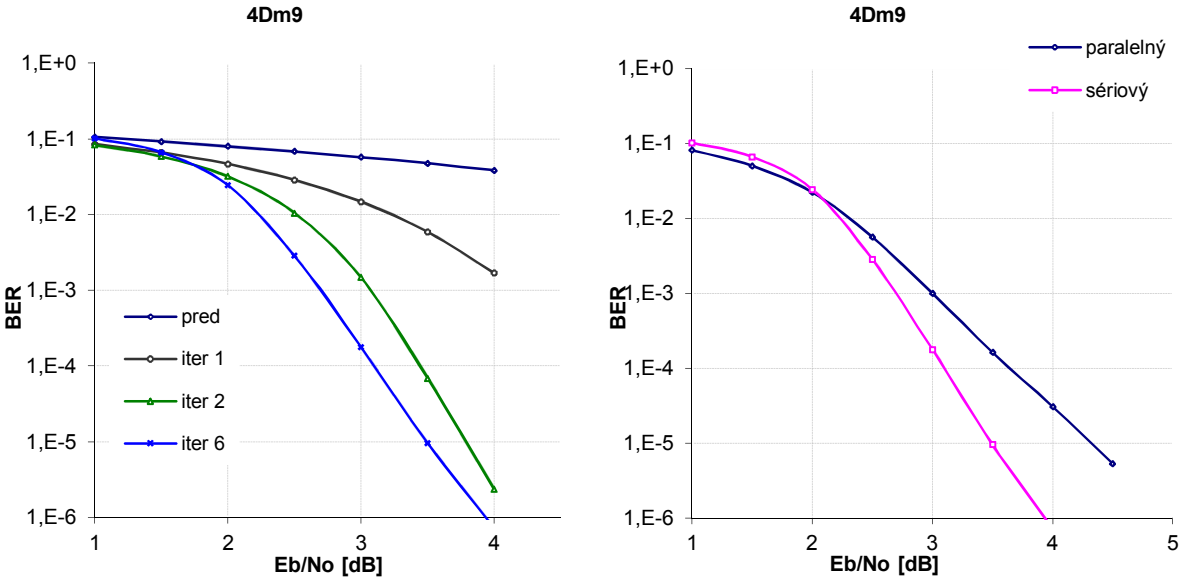


Obrázok 5.2 Schéma iteratívneho *SISO* dekodéra pre 2D SP TPC

Algoritmus dekódovania je, na rozdiel od algoritmu popísaného v časti 5.1.1, sekvenčný. Dekodéry v tomto prípade počítajú extrinsické hodnoty a upravujú mäkké hodnoty. Dekodér kódu C_1 počíta extrinsické hodnoty aplikovaní vzťahu 5.12 na kódové slová kódu C_1. Interleaver 1 zhromaždí upravené apriórne *LLR* všetkých bitov a vysiela ich pozdĺž osi *Y* prijatého slova tak, aby hodnoty vstupujúce do dekodéra kódu C_2 boli „kódové“ slová kódu C_2. Ten opäť upraví mäkké hodnoty (použitím 5.21). Deinterleaver vykoná reverzný proces k činnosti iterleavera 1, t.j. vysiela *LLR* hodnoty v poradí, ako ich „akceptuje“ dekodér C_1. Tým je zavŕšená jedna iterácia.

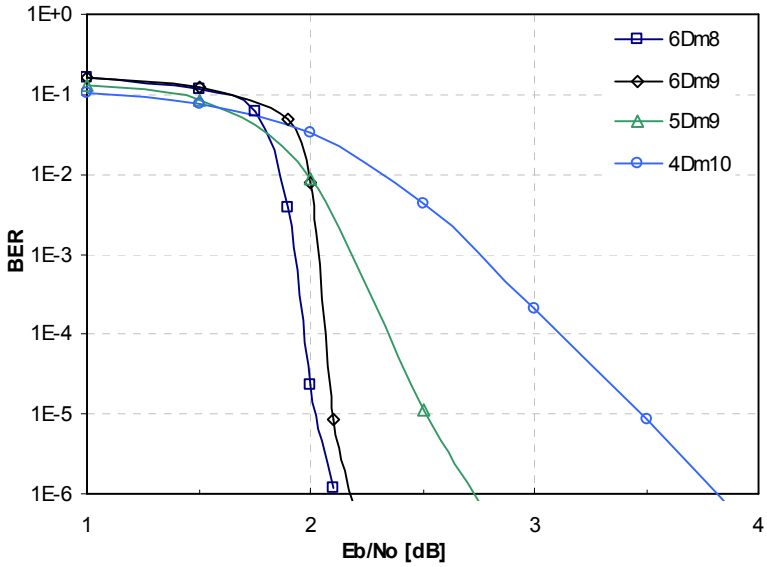
Výkonnosti

Korekčné schopnosti paritných kódov násobkov sú analyzované v [RAN01] a prezentované v [JAN03b]. Z výsledkov na obrázku 5.3 vidieť ako sa vyvíja zvyšková chybovosť počas jednotlivých iterácií (obrázok a) a porovnanie korekčných schopností sériovo a paralelne zreťazeného kódu. Na obrázku 5.4 je graf závislosti zvyškovej chybovosti v závislosti na rozmere kódu tým teda aj na veľkosti kódového slova.



a) b)

Obrázok 5.3 Graf závislosti BER od E_B/N_0 pre rôzny počet iterácií pre kód 4Dm9 a porovnanie sériovo a paralelne zreťazeného kódu (AWGN kanál, BPSK)



Obrázok 5.4 Graf závislosti BER od E_B/N_0 pre kódy rôznych rozmerov pre AWGN kanál s použitím BPSK modulácie

Z obr. 5.3a vidieť, že najväčší zisk kódovania prinášajú prvé dve iterácie. Je to dôsledok malej štatistickej závislosti medzi extrinsickými a apriórnymi hodnotami *LLR* [HAG96].Počas

dekódovania vzrastá štatistická závislosť nakoľko apriórne *LLR* upravujeme pomocou extrinsických *LLR*, ktoré následne opäť počítame pomocou apriórnych, preto zisk s pribúdajúcimi iteráciami klesá. Z výsledkov simulácií z obr. č. 5.4 vidieť ako narastajú korekčné vlastnosti s rastúcou dimenziou kódu.

6. Takmer optimálne iteratívne dekódovanie kódov násobkov

6.1 Algoritmus dekódovania

Mäkké dekódovanie lineárnych blokových kódov

V tejto časti opíšem algoritmus dekódovania *LBK* [PYN98], ktorý bude neskôr aplikovaný pri iteratívnom dekódovaní TPC opísaných v kapitole 4. Opäť budem uvažovať s prenosom binárnych symbolov cez AWGN kanál s použitím BPSK modulácie. Pre jednoduchosť budeme opäť uvažovať, že energia signálov *E* bude jednotková. Ak použijeme také mapovanie bitov, že ak na vstupe modulátora bol bit 0, na výstupe demodulátora v ideálnom prípade bude hodnota –1. Pre vstupný bit 1 by to mala byť hodnota 1. V ďalšom opise budeme toto mapovanie využívať.

Prijaté slovo na výstupe detektora označím $\boldsymbol{Z} = (z_1, ..., z_i, ..., z_n)$. Toto slovo sa dá vyjadriť nasledovným súčtom, čo je výstup demodulátora, obr. 2.2, resp. 2.3 :

$$Z = E + N \qquad (6.1)$$

kde $\boldsymbol{E}$ je vyslané kódové slovo pozostávajúce z hodnôt : $\boldsymbol{E} =(e_1, ..., e_i, ..., e_n)$, pričom $e_i=-1$ ak bol vyslaný bit 0 a $e_i=+1$, ak bol vyslaný bit 1 (dôsledok mapovania). Hodnoty n_i slova $\boldsymbol{N}=(n_1, ..., n_i, ..., n_n)$ sú komponenty aditívneho bieleho gaussovského šumu s disperziou σ^2. Pri použití optimálneho rozhodovania (*MLD*) by sme za vyslané kódové slovo prehlásili také kódové slovo $\boldsymbol{D}$, ktorého vzdialenosť od prijatého slova $\boldsymbol{Z}$ je minimálna. Algebraicky zapísané

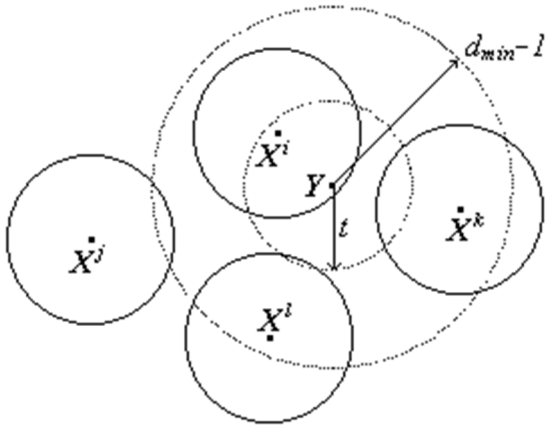
$$D = C^i \quad \text{ak} \quad \left|Z - C^i\right|^2 \le \left|Z - C^j\right|^2 \qquad \forall j \in [1,2^k], \ \ j \neq i \qquad (6.2)$$

kde $\boldsymbol{C^i} =(c_1^i, ..., c_i^i, ..., c_n^i)$ je *i*-té kódové slovo použitého kódu komponentu *C* s parametrami (n, k, d_{min}) pozostávajúce zo symbolov { -1, +1 } a člen

$$\left|Z - C^i\right|^2 = \sum_{i=1}^{n} \left(z_i - c_i\right)^2 \qquad (6.3)$$

vyjadruje druhú mocninu euklidovskej vzdialenosti medzi slovami ***Z*** a $\boldsymbol{C^i}$. Aby sme našli optimálne kódové slovo ***D***, potrebujeme „preskúšať " všetky kódové slová kódu komponentu. Keďže počet slov kódu rastie exponenciálne s počtom informačných bitov 2^k, algoritmus by bol nepoužiteľný už pre blokové kódy s dĺžkou slova $k>6$. Keďže na priblíženie sa ku kapacite kanála potrebujeme použiť kód s čo najväčšou dĺžkou slova, treba zvoliť iný spôsob ako nájsť optimálne slovo ***D***.

V roku 1972 Chase navrhol suboptimálny algoritmus [CHA72] s relatívne malou zložitosťou pre takmer optimálne dekódovanie *LBK*. Algoritmus vychádza z predpokladu, že pri vyššom odstupe signál-šum optimálne (s najväčšou pravdepodobnosťou vyslané) kódové slovo ***D*** sa nachádza vnútri *n*-rozmernej guli s polomerom $d_{min}-1$ okolo prijatého slova $\boldsymbol{Y}=(y_1, ..., y_i, ..., y_n)$, pričom symboly $y_i = 0{,}5 \times \left(1 + \mathrm{sgn}(z_i)\right)$. Viď obrázok č. 6.1.



Obrázok 6.1 Grafické znázornenie rozloženia kódových slov okolo prijatého slova ***Y***

Takže množinu kódových slov, pre ktoré budeme počítať euklidovskú vzdialenosť (6.2), obmedzíme len na najpravdepodobnejšie kódové slová vnútri *n* -rozmernej gule. Ako však určiť najpravdepodobnejšie kódové slová vnútri tejto gule? Na to sa využije prijaté slovo ***Z***. Postup, ako identifikovať množinu najpravdepodobnejších kódových slov je nasledovný :

1) Nájdeme *p* najmenej spoľahlivých symbolov v slove *Y*. Hodnota *p* je voliteľná. Pochopiteľne, čím bude *p* väčšie tým bude algoritmus zložitejší. Spoľahlivosť jednotlivých symbolov y_i určíme z prijatého slova ***Z***, ako to vysvetlím neskôr.
2) Vytvoríme postupne chybové slová $\boldsymbol{T^q}$. Počet všetkých slov bude *r,* každé dĺžky *n*.

$$r = 1 + p + \binom{p}{2} + ... + \binom{p}{p-1} + 1 = 2^p \qquad (6.4)$$

Slová budú vyzerať nasledovne: Prvé slovo bude nulové. Ďalších *p* slov bude obsahovať postupne jednu "1" na najmenej spoľahlivých miestach a "0" na ostatných miestach. Ďalšia skupina slov bude obsahovať po dve "1" na najmenej spoľahlivých miestach a na ostatných "0". Počet týchto slov je daný počtom možných kombinácií dvojíc, teda *p* nad 2. Ďalšia skupina testovacích slov bude obsahovať tri "1" na najmenej spoľahlivých miestach, zvyšok "0" atď. Až skupinu uzavrieme jedným slovom s *p* jednotkami.

3) Ku každému slovu $\boldsymbol{T^q}$ priradíme testovacie slovo $\boldsymbol{S^q}$ tak, že pre jednotlivé bity testovacieho slova platí $s_i^q = y_i \oplus t_i^q$. Následne algebricky dekódujeme slovo $\boldsymbol{S^q}$, čoho výsledkom bude kódové slovo $\boldsymbol{X^q}$, ktoré pridáme do množiny Φ.

Po dekódovaní všetkých testovacích slov množina Φ bude obsahovať kódové slová $\boldsymbol{X^q}$ (obr. č.6.1), ktoré sa nachádzajú v blízkosti nami prijatého a sú vhodní „kandidáti" na optimálne kódové slovo ***D***. Optimálneho slovo ***D*** je získané pomocou vzťahu 6.2, aplikovaného len na množinu Φ. Pred počítaním euklidovských vzdialeností je potrebné previesť kódové slova $\boldsymbol{X^q}$ na kódové slová $\boldsymbol{C^q}$ mapovaním bitov { 0,1 } na symboly { -1,+1 }.

Teraz sa vrátim k tomu, ako vypočítať spoľahlivosť bitov y_i. Spoľahlivosti jednotlivých bitov y_i určíme na základe už spomínanej metriky *LLR*. Ak sú vysielané bity 0, 1 rovnako pravdepodobné, pre *LLR* na výstupe detektora platí

$$L(y_i / z_i) = \log_e \left[\frac{P(y_i = 1/z_i)}{P(y_i = 0/z_i)} \right] = \frac{2}{\sigma^2} z_i \qquad (6.5)$$

V prípade, že σ je konštanta, tak *LLR* môžeme predeliť členom 2 / σ^2 a spoľahlivosť bitu y_i bude daná veľkosťou $|z_i|$.

*Chase*ov algoritmus nám pre každý riadok (resp. stĺpec) kódu násobku dáva optimálne slovo ***D***. Iteratívne dekódovať kód násobok bude možné iba vtedy, ak budeme schopní určiť spoľahlivosť slova ***D***. Určiť spoľahlivosť optimálneho slova bude znamenať vypočítať, s akou pravdepodobnosťami sú jednotlivé symboly d_i slova ***D*** totožné s vyslanými bitmi. Na základe vypočítaných pravdepodobností upravíme mäkké hodnoty z_i. Po upravení všetkých hodnôt môžeme pokračovať v dekódovaní stĺpcov (resp. riadkov) atď.

Výpočet spoľahlivosti bitov optimálneho slova

Ako náhle získame optimálne slovo ***D***, vypočítame spoľahlivosť jednotlivých symbolov slova na základe prijatého slova ***Z***. Spoľahlivosť symbolov d_i bude opäť daná metrikou *LLR* :

$$L(d_i) = \log_e \left[\frac{P(e_i = +1/\boldsymbol{Z})}{P(e_i = -1/\boldsymbol{Z})} \right] \qquad (6.5)$$

Výpočet *LLR* pre symboly d_i sa však bude odlišovať od výpočtu *LLR* pre bity y_i (6.4). Treba vziať do úvahy, že optimálne slovo ***D*** je jedno z 2^k kódových slov ***C***. Preto čitateľa vo vzťahu 6.5 možno vyjadriť nasledovne :

$$P(e_i = +1/Z) = \sum_{C^j \in \Omega^{+1}} P(E = C^j / Z) \qquad (6.6)$$

kde Ω^{+1} je množina všetkých kódových slov $\boldsymbol{C}^j$ pre ktoré platí, že symbol $c_i^j = +1$. Podobný vzťah možno napísať aj pre menovateľa (6.5)

$$P(e_i = -1/Z) = \sum_{C^j \in \Omega^{-1}} P(E = C^j / Z) \qquad (6.7)$$

Pričom Ω^{-1} je množina všetkých kódových slov $\boldsymbol{C}^j$ pre ktoré platí, že symbol $c_i^j = -1$. Po dosadení vzťahov 6.6 a 6.7 do 6.5 a použití Bayesovho vzorca za predpokladu, že všetky kódové slová sú vysielané s rovnakou pravdepodobnosťou, dostávame vzťah pre výpočet *LLR*:

$$L(d_i) = \log_e \left(\frac{\sum_{C^j \in \Omega^{+1}} p(\boldsymbol{Z}/\boldsymbol{E} = C^j)}{\sum_{C^j \in \Omega^{-1}} p(\boldsymbol{Z}/\boldsymbol{E} = C^j)} \right) \qquad (6.8)$$

kde $p(\boldsymbol{Z}/\boldsymbol{E} = \boldsymbol{C}^j)$ je funkcia podmienenej hustoty pravdepodobností prijatého slova ***Z*** za predpokladu vyslaného slova ***E***. Hustota je daná :

$$p(\boldsymbol{Z}/\boldsymbol{E} = \boldsymbol{C}^j) = \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \times \exp\left(-\frac{\left| \boldsymbol{Z} - \boldsymbol{C}^j \right|^2}{2\sigma^2} \right) \qquad (6.9)$$

Po substitúcii podmienenej hustoty danej (6.9) do čitateľa aj menovateľa (6.8) dostávame :

$$L(d_i) = \log_e \frac{\left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \times \left[\exp\left(-\frac{\left| \boldsymbol{Z} - \boldsymbol{C}^{+1(i)} \right|^2}{2\sigma^2} \right) + \exp\left(-\frac{\left| \boldsymbol{Z} - \boldsymbol{C}^{+1(j)} \right|^2}{2\sigma^2} \right) + \ldots \right]}{\left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \times \left[\exp\left(-\frac{\left| \boldsymbol{Z} - \boldsymbol{C}^{-1(i)} \right|^2}{2\sigma^2} \right) + \exp\left(-\frac{\left| \boldsymbol{Z} - \boldsymbol{C}^{-1(j)} \right|^2}{2\sigma^2} \right) + \ldots \right]} \qquad (6.10)$$

kde $\boldsymbol{C}^{+1(i)}$ je také kódové slovo z množiny Ω^{+1}, ktorého euklidovská vzdialenosť od prijatého slova ***Z*** je minimálna. $\boldsymbol{C}^{+1(j)}$ bude druhé najbližšie kódové slovo z množiny Ω^{+1}, $\boldsymbol{C}^{+1(k)}$ by bolo tretie najbližšie atď. Podobná situácia je v menovateli. $\boldsymbol{C}^{-1(i)}$ bude zase kódové slovo

z množiny Ω^{-1} najbližšie k prijatému slovu ***Z***, $\boldsymbol{C}^{-1(j)}$ bude druhé najbližšie z množiny Ω^{-1} atď. Ak uvážime, že hustoty podmienených pravdepodobností klesajú exponenciálne s rastúcou euklidovskou vzdialenosťou je a pri vyššom *SNR* takisto klesá výkon šumu : $\sigma^2 \rightarrow 0$, môžeme v čitateli aj menovateli zanedbať všetky členy sumy okrem prvého. Po úprave získame :

$$L(d_i) = \frac{1}{2\sigma^2}\left(\left|\boldsymbol{Z} - \boldsymbol{C}^{-1(i)}\right|^2 - \left|\boldsymbol{Z} - \boldsymbol{C}^{+1(i)}\right|^2\right) \qquad (6.11)$$

Ako vidíme zo vzťahu 6.11 na určenie spoľahlivosti bitu d_i nám postačia dve kódové slová $\boldsymbol{C}^{-1(i)}$ a $\boldsymbol{C}^{+1(i)}$ z množiny Φ, ktoré sú najbližšie k slovu ***Z***, s podmienkou $c_i^{+1(i)} = -c_i^{-1(i)}$. Pochopiteľne slovo, ktoré je bližšie k slovu ***Z*** bude optimálne slovo ***D*** a druhé jeho „konkurent“. Čím ďalej je konkurent od optimálneho slova, tým vyššia je spoľahlivosť nášho rozhodnutia v prospech slova ***D***.

Po dosadení 6.3 do 6.11 sa navzájom odčítajú tie členy $\left(z_j - c_j^{-1(i)}\right)^2$ a $\left(z_j - c_j^{+1(i)}\right)^2$, pre ktoré platí $c_j^{+1(i)} = c_j^{-1(i)}$. Tam, kde platí $c_j^{+1(i)} = -c_j^{-1(i)}$ bude výsledkom rozdielu $4.z_j.c_j^{+1(i)}$. Výsledná rovnica pre *LLR* bitu d_i, z využitím hodnoty z_i bude :

$$L(d_i) = \frac{2}{\sigma^2}\left(z_i + \sum_{l=1, l\neq i}^{n} z_l.c_l^{+1(i)}.p_l\right) \qquad (6.12)$$

pričom pre symboly p_l platí to, čo bolo písané v predchádzajúcom odseku,

$$p_l \begin{cases} 0, & ak \quad c_l^{+1(i)} = c_l^{-1(i)} \\ 1, & ak \quad c_l^{+1(i)} = c_l^{-1(i)} \end{cases} \qquad (6.13)$$

Za predpokladu, že σ je konštanta, tak vypočítanú *LLR* (6.12) možno znormovať členom $2 / \sigma^2$ a dostaneme nasledovný vzťah :

$$z'_i = z_i + w_i \qquad (6.14)$$

kde

$$w_i = \sum_{l=1, l\neq i}^{n} z_l.c_l^{+1(i)}.p_l \qquad (6.15)$$

Znormalizovaná hodnota z'_i bude reprezentovať mäkkú hodnotu na výstupe dekodéra, ktorá môže byť opätovne použitá ako vstupná mäkká hodnota pre nasledujúcu iteráciu. Znamienka symbolov d_i a z'_i sú rovnaké a absolútna veľkosť z'_i nám udáva jeho spoľahlivosť. Ak porovnáme vzťahy 6.14 a 5.11 vidíme, že člen hrá tú istú rolu w_i ako extrinsická hodnota. Platia pre ňu aj tie isté podmienky, t.j. je to mäkká hodnota získaná z procesu dekódovania a pri jej výpočte sa *priamo* neuvažuje so symbolom z_i resp. d_i. Preto hovorím priamo, lebo

ako vieme z predchádzajúcej kapitoly, po prvej iterácií vzniká medzi vstupnou a extrinsickou hodnotu istá štatistická závislosť (viď časť 5.2.2).

Zo vzťahu 6.11 vieme, že na korigovanie mäkkej hodnoty potrebujeme dve slová z množiny Φ (získané *Chase*ovým algoritmom), ktoré sú najbližšie k slovu ***Z*** a zároveň majú opačné hodnoty bitov na mieste korigovaného symbolu. Ak znormujeme vzťah 6.11 konštantou 2 / σ^2 a to kódové slovo, ktoré je bližšie ku ***Z*** označíme ***D*** a to čo je ďalej ***C***, tak získame rovnicu pre výpočet z'_i pomocou euklidovských vzdialeností týchto slov od ***Z*** :

$$z'_i = \left(\frac{|\boldsymbol{Z}-\boldsymbol{C}|^2 - |\boldsymbol{Z}-\boldsymbol{D}|^2}{4} \right) \times d_i \qquad (6.16)$$

Výpočet je veľmi jednoduchý, avšak je tu jeden problém. Je jasné, že slovo ***D*** sa nám podarí nájsť určiť vždy. To však neplatí o konkurenčnom slove ***C,*** pretože môže nastať situácia kedy *Chase*ov algoritmus vráti bohatú množinu okolitých kódových slov k slovu ***Z***. Všetky však budú mať na mieste korigovaného symbolu rovnaké hodnoty, či už –1, alebo +1. Jedna z možností ako vyriešiť tento problém, je zväčšiť hodnotu *p* v prvom kroku *Chase*ovho algoritmu. Tým zväčšíme množstvo testovacích slov, následkom čoho bude množina Φ bohatšia. Pochopiteľne zložitosť algoritmu bude s rastúcou hodnotou *p* narastať. Ba čo viac, aj vtedy môže prísť k situácií, že neexistuje konkurenčné slovo. Vtedy by sa dalo predpokladať, že symbol z_i je s veľkou pravdepodobnosťou totožný s d_i. Treba však nájsť spôsob, ako upraviť hodnotu pravdepodobnosť z_i. V [PYN98] je spôsob úpravy symbolu z_i navrhnutý nasledovne :

$$z'_i = \beta \times d_i, \qquad \beta > 0 \qquad (6.17)$$

Úprava pomocou 6.17 je veľmi jednoduchá a ako ďalej uvidíme aj efektívna. Je dôležité, že vzťah 6.17 má tú istú vlastnosť ako 6.16 – znamienko z_i je po úprave totožné s d_i. Ostáva už len stanoviť faktor β, pričom musíme vziať do úvahy nasledovné skutočnosti :

- $\beta > 0$, aby znamienka z_i a d_i boli rovnaké
- Ak sa nepodarilo nájsť slovo ***C***, je s najväčšou pravdepodobnosťou veľmi ďaleko od upravovaného slova ***Z***.
- Ak slovo ***C*** je ďaleko od ***Z***, spoľahlivosť d_i je väčšia, tým pádom musí byť aj β väčšia.
- Na začiatku dekódovania je chybovosť vyššia, a teda rozhodnutie sa pre optimálne slovo ***D*** nemá až takú váhu ako neskôr. Ako dekódovanie pokračuje, upravované slovo ***Z*** sa čím ďalej tím viac „podobá“ na nejaké kódové slovo. Potom je logické, že nemusí existovať konkurenčné slovo. Hodnota β teda s pribúdajúcimi iteráciami raste.

Hodnota β je teda úmerná :

$$\beta \approx \log_e \left[\frac{P(e_i = d_i)}{P(e_i \neq d_i)} \right] \qquad (6.18)$$

Ak $P(e_i = d_i) \to 1$ potom $\beta \to \infty$ a podobne ak $P(e_i = d_i) \to 0.5$ potom $\beta \to 0$.

Teraz, keď vieme ako vypočítať mäkké hodnoty na výstupe *SISO* dekodéra kódu komponentu, môžeme prejsť k iteratívnemu dekódovaniu.

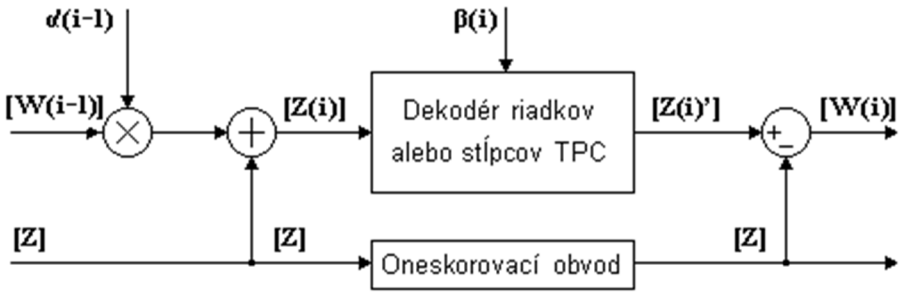
Iteratívne dekódovanie

Na rozdiel od algoritmu v časti 5.2 netreba prijaté hodnoty zj nijako upravovať na LLR hodnoty, maximálne len znormovať na jednotkovú energiu. Je to výhoda, lebo nemusíme dopredu poznať charakteristiku kanála. Ak ju poznáme, nič nebráni tomu, aby sme pracovali s LLR hodnotami. Algoritmus iteratívneho dekódovania pre prijaté slovo je nasledovný :

Nech [Z] predstavuje prijaté slovo kódu násobku na vstupe SISO dekodéra. Algoritmus iteratívneho dekódovania prijatého slova [Z] je nasledovný :

1. Nastav počítadlo dekódovaní i =1, α(0) = 0.
2. Použitím Chaseovho algoritmu na riadky (resp. stĺpce) slova [Z(i)] a vzťahov 6.16 a 6.17 vypočítame pre každý symbol zj novú mäkkú hodnotu z'j. Výstupom bude slovo [Z(i)'].
3. Odpočítaním slova [Z(i)'] od prijatého slova [Z] získame extrinsickú informáciu [W(i)] pre dekódovanie stĺpcov (resp. riadkov).
4. Vstupné slovo pre dekódovanie stĺpcov (resp. riadkov) získame nasledovnou úpravou : [Z(i+1)] = [Z] + α(i)[W(i)]
5. Ak bolo vykonaných dosť iterácií, aby sme mohli o každom bite spoľahlivo rozhodnúť, prejdi na bod 6. Inak inkrementuj i =i+1 a prejdi na bod 2.
6. Urob tvrdé rozhodovanie pre každý bit slova [Z(i+1)], ak je mäkká hodnota z'_j <0 bit dekóduj ako "0" inak ako "1".

Schéma SISO dekodéra komponentu TPC je na obr. 6.2. V súvislosti s dekódovacím algoritmom treba ešte objasniť určité veci.



Obrázok 6.2 Bloková schéma SISO dekodéra pre kódy komponenty

Extrinsickú hodnotu počítame podľa vzťahu 6.14, ako rozdiel novej mäkkej hodnoty z'_j a hodnoty z kanála z_j. Výhodu to má v tom, že nepotrebujeme upravovať hodnoty na výstupe detektora podľa charakteristiky kanála (napr. na LLR hodnoty).

V bode 3 algoritmu najskôr vypočítame extrinsickú hodnotu w_i, ktorú v kroku 4 vynásobíme faktorom α a až potom pripočítame k vstupnému symbolu z_j. Faktorom α zohľadňujeme to, že symboly v prijatom slove [Z] a vypočítané extrinsické hodnoty v slove [W(i)] majú rozdielnu varianciu [PYN98]. Rozptyl hodnôt vo [W(i)] je podstatne vyšší na začiatku dekódovania a s pribúdajúcim iteráciami klesá. Platí teda, že na začiatku dekódovania má faktor α malé hodnoty, aby sa vyvážil veľký rozptyl extrinsických hodnôt. Rozptyl extrinsických hodnôt v ďalších krokoch dekódovania klesá, tým pádom môže faktor α rásť. Hodnoty α by mohli byť nasledovné : α(i) = (0, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9, 1.0, 1.0 …), pričom i=0,1,2,…. Pri výsledkoch simulácií uvediem konkrétne použité hodnoty.

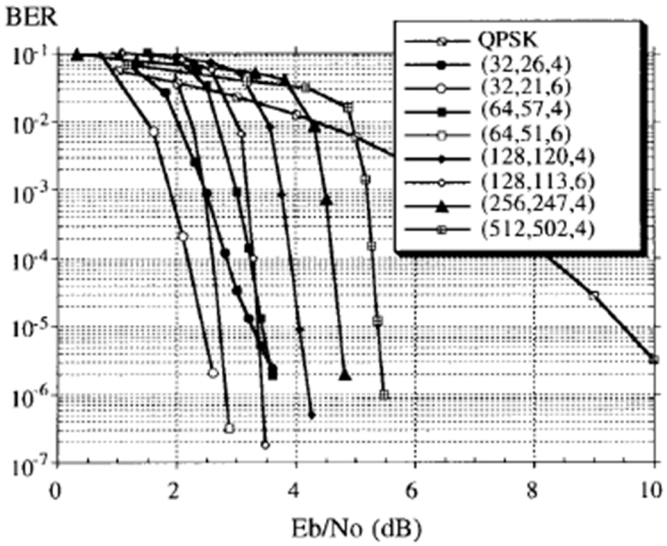
Na záver sa vrátim ešte k faktoru β. Vieme, že hodnoty β by počas dekódovania mali rásť, čo je dôsledok znižovania BER. Ak je počas dekódovania BER blízka nule, β by mala byť veľká a opačne. Z toho vyplýva, že β je funkciou BER. Ako teda stanoviť hodnoty β(i), aby boli čo najlepšie závislé od momentálnej chybovosti slova? Pri pohľade na (6.18) to nie je ľahké, keďže interval možných hodnôt β je $< 0, \infty)$. Tento zdanlivý problém sa dá vyriešiť nasledovným spôsobom [PYN98]. Extrinsickú hodnotu týchto bitov môžeme odhadnúť vzťahom :

$$w_j = \beta \times \overline{w} \times d_j \qquad (6.19)$$

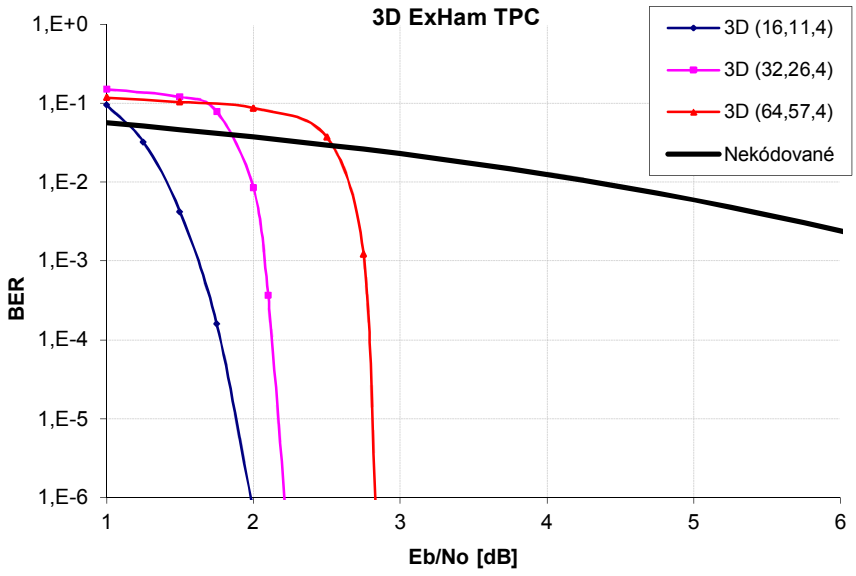
kde $\overline{w}$ bude priemerná hodnota z absolútnych extrinsických hodnôt pre tie symboly z_i , pre ktoré existuje aj konkurenčné slovo C. Teda ich nové mäkké hodnoty z'_i sa dajú vypočítať vzťahom 6.16. Hodnoty $\beta(i)$ budú potom z intervalu $< 0,1 >$, napr. : $\beta(i)$ =(0.2, 0.4, 0.5, 0.6, 0.7, 0.9, 1.0, 1.0,....), i=1,2,3... .

6.2 Výkonnosti

Korekčné schopnosti modifikovaného *Chase*ovho algoritmu sú prezentované v [PYN98] ale boli odsimulované aj v rámci tejto práce. Na obrázku 6.3 sú znázornené výkonnosti rôznych 2D SPC zložených s rozšírených BCH kódov. Na obrázku 6.4 je znázornené ako sa zlepšuje výkonnosť po jednotlivých iteráciách (1 iterácia je dekódovanie vo všetkých rozmeroch).



Obrázok 6.3 Výkonnosti rôznych rozšírených 2D BCH kódov pre AWGN kanál s použitím BPSK modulácie, prevzaté z [PYN98]

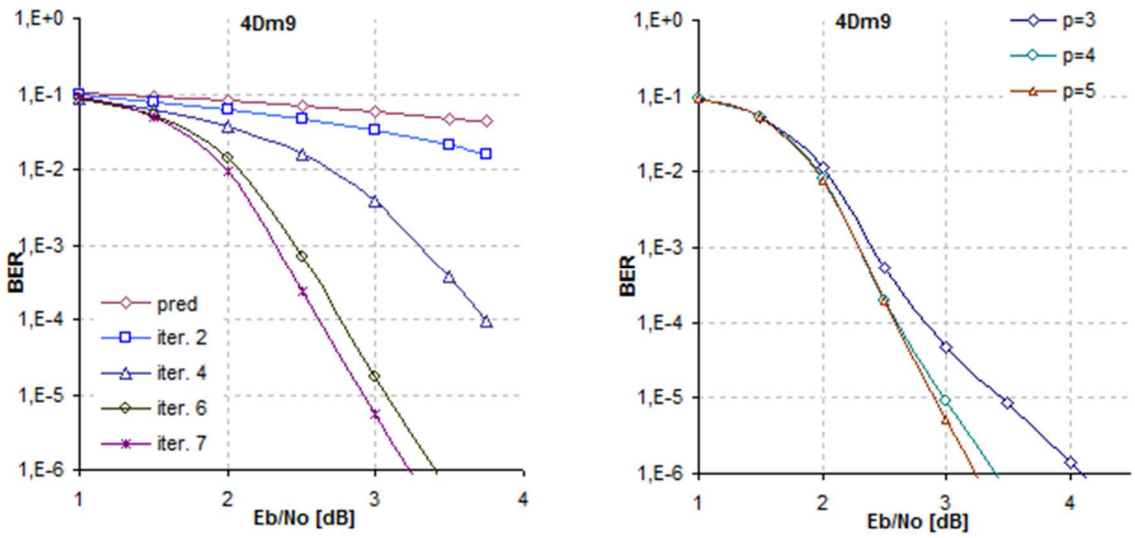


Obrázok 6.4 Graf závislosti BER od E_B/N_0 pre SC 3D rozšírené Hamming. kódy

V rámci zovšeobecnenia boli zrealizované konštrukcie 3D Hammingových a rozšírených Hammingových kódov a následne na nich aplikované uvedené dekódovacie algoritmy, výkonnosti vybraných 3D kódov sú na obr. 6.4.

Trojrozmerné kódy násobky zložené z rozšírených Hammingových kódov vykazujú vlastnosti kódov použiteľných v systémoch, kde je dôležitá zabezpečená komunikácia, resistentná voči odpočúvaniu. Samoopravné kódy a dekódovanie vhodné pre tieto systémy sa vyznačujú malým tzv. bezpečnostným rozstupom (Security Gap) kanalových E_b/N_0. Bezpečnostný rozstup kanálových E_b/N_0 je definovaný rozdielom kanálových E_b/N_0, pre ktoré kód a dekódovací algoritmus vykazuje vysokú chybovosť (blížiacou sa k 1/2) versus chybovosť na úrovni 10^{-5} [BAL15].

Výkonnosť *Pyndiah - Chase*ovho algorimu aplikovaného na kódy násobky zložené z jednoduchých paritných kódov boli prezentované v [JAN04].

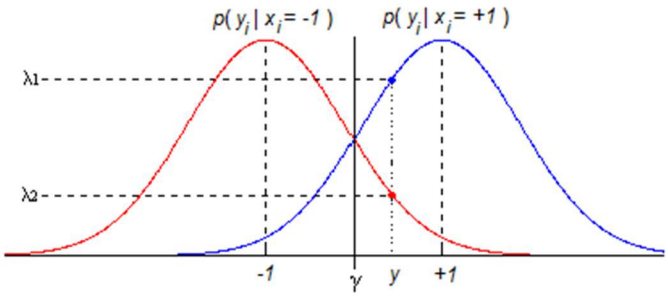


a) b)

Obrázok 6.7 Výkonnosť 4D kódu (9,8,2) v AWGN s mod. BPSK, a) vývoj BER počas dekódovania, b) graf výkonnosti vzhľadom na veľkosť parametra *p*, publikované z [JAN04]

7. Tvrdé iteratívne dekódovanie s N úrovňami kvantizácie

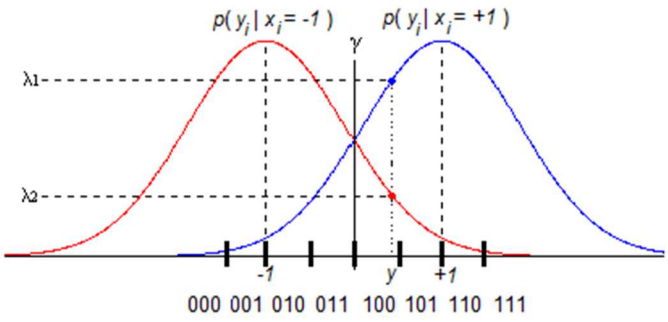
V [JAN15] sme navrhli tvrdé dekódovanie kódov násobkov zložených s jednoduchých paritných kódov s N-úrovňami kvantizácie. Algoritmus spája v sebe výhody tvrdého aj mäkkého dekódovania. Tvrdé rozhodovanie na výstupe rozhodovacieho bloku je dané podmienenými hustotami pravdepodobností prijatého signálu y za predpokladu vyslania symbolu x.



Obrázok 7.1 Pravdepodobnosté funkcie ML tvrdého rozhodovania.

Toto rozhodovacie kritérium sa nazýva ML, bolo už popísané v kapitole 5.2. Algoritmus mäkkého dekódovania dosahuje výborné výsledky [ARG02], [JAN03b] avšak vyžaduje relatívne vysokú komplexnosť dekodéra.

Naopak, ak by sa uvažovalo s čistým HIHO algoritmom, veľa informácie je tak povediac zahodenej. Preto sme v [JAN15] navrhli kvantizovať reálnu mäkkú hodnotu na výstupe do 8 resp. 16tich úrovní. Príklad kvantizácie výstupnej mäkkej hodnoty do 8 úrovní je na obrázku 7.1. Počet úrovní budeme označovať N_q, počet prahových hodnôt je potom rovný N_q-1.



Obrázok 7.2 Úrovne kvantizácie pre pravdepodobnostné funkcie AWGN kanála.

Aj na základe obrázku 7.2 vidieť, že nepracujeme s jedným bitom, t.j. s tvrdou hodnotou pre prijatý ale použijeme 3 bity pre 8 úrovní kvantizácie alebo 4 bity pre 16 úrovní kvantizácie. Prvý bit predstavuje tvrdú hodnotu, ktorú používa dekodér pre výpočet syndrómov (kapitola 5.1). Vnútorné kvantizačné úrovne predstavujú úrovne medzi strednými hodnotami mäkkých hodnôt. Vonkajšie kvantizačné úrovne predstavujú úrovne mimo stredných hodnôt. Pre 8 úrovní, podľa obrázku 7.6 existujú 4 vnútorné kvantizačné úrovne (010, 011, 100, 101) a 4 vonkajšie úroovne (000, 001, 110, 111. Pre moduláciu M-QAM by sme navrhli kvantizáciu i-zložky a q-zložky signálových komponentov a porovnávali ich s LUT (Look-up tabuľka) [BAL09]. LUT tabuľla by uchovávala kvantizačné úrovne prislúchajúce binárne reprezentácie každého z Log_2N_q bitov v prijatom symbole.

Dekódovací algoritmus pre tvrdé iteratívne dekódovanie s N úrovňami kvantizácie je podobné ako tvrdé dekódovanie popísane v kapitole 5.1 s jedným podstatným rozdielom. Pri dosiahnutí alebo prekročení prahovej hodnoty nesúhlasných paritných rovníc pre danú iteráciu nie je symbol invertovaný ako je tomu pri klasickom tvrdom dekódovaní, ale je len „posunutý“ o jednu kvantizačnú úroveň smerom k druhému symbolu. Úprava symbolu je vyjadrená vzťahom 7.3.

$$y_{i+1} = \begin{cases} y_i + 1, & \text{ak} \quad y_i < 0 \\ y_i - 1, & \text{ak} \quad y_i \geq 0 \end{cases} \tag{7.1}$$

V prípade ak súčet hodnôt syndrómov menší ako prahová hodnota, hodnota kvantizačnej úrovne sa upraví smerom k „potvrdeniu“ symbolu.

Výkonnosti uvedeného spôsobu dekódovania pre rôzne dimenzie paritných kódov násobkov boli merané pre AWGN kanál s použitím BPSK modulácie, obr. 7.3. Parametre kódov pre jednotlivé iterácie sú uvedené v tabuľke 7.1

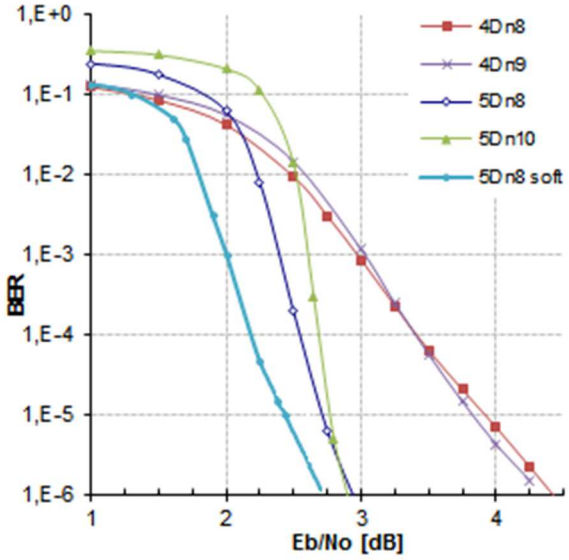
Kód	**Rýchlosť**	**Veľkosť slova**	**d_{min}**
4Dn8	0.5862	4096	16
4Dn9	0.6243	6561	16
5Dn8	0.5129	32768	32
5Dn10	0.5905	100000	32

Tabuľka 7.1 Parametre skúmaných kódov

Prahové hodnoty pre 4D a 5D kódy:

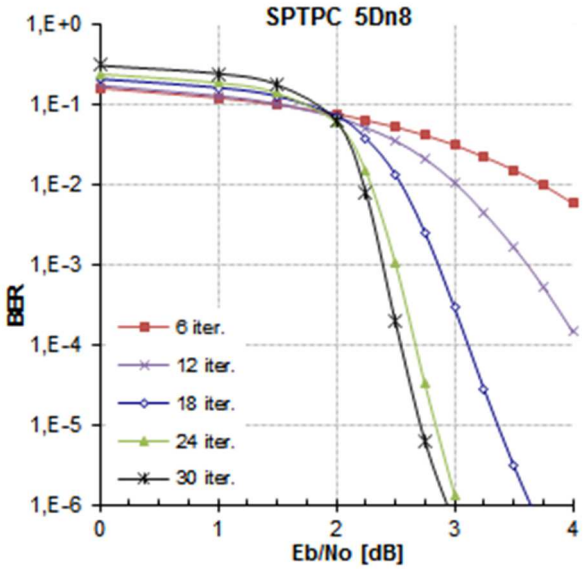
T_{4D} {4,3,4,2,3,4,2,3,2,4,2,3,3,2,3,4,2,3,3,3}

T_{5D} {5,4,3,5,3,4,5,3,4,2,5,4,2,3,5,4,2,3,4,3,2,4,3,3,4,2,3,3,2,3}



Obrázok 7.3 Výkonnosti rôznych 4D a 5D SPTPC kódov pri dekódovaní hybridným algoritmom, prevzaté z [JAN15].

Kapacita AWGN kanála pre kódy s rýchlosťou 0.5129 je na úrovni 0.17dB pre dosiahnutie chybovosti na symbol 10^{-5} [RAN01]. Navrhnutý spôsob dekódovanie je tak 2.53 dB od kapacity kanála pre dosiahnutej danej chybovosti, obrázok 7.4. V porovnaní s referenčným mäkkým dekódovaním má navrhnuté hybridné dekódovanie stratu len 0.23dB pri BER 10^{-5}, čo predstavuje vynikajúce výsledky, ak zohľadníme rýchlosť a zjednodušenú komplexitu dekódovania.



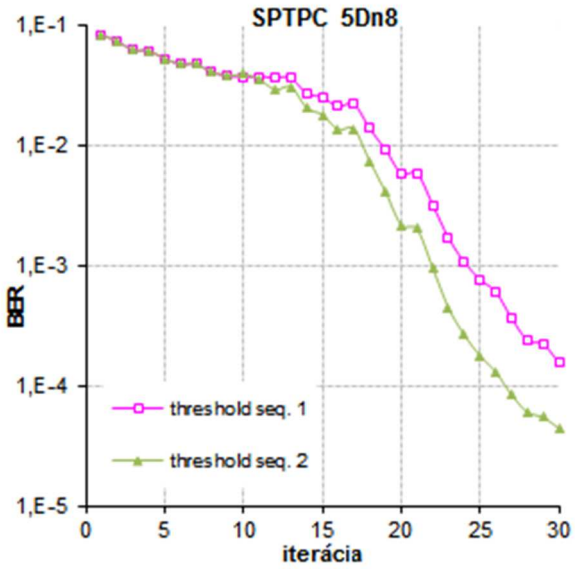
Obrázok 7.4 Výkonnosť 5D kódu (8,7,2) v kanály AWGN s mod. BPSK, vývoj BER v závislosti od iterácie dekódovania, prevzaté z [JAN15].

Dôležitým aspektom pri dekódovaní je správny výber sekvencie prahových hodnôt. Pre každý kód a sekvenciu prahových hodnôt môže byť zadefinovaná hraničná hodnota SNR kanála

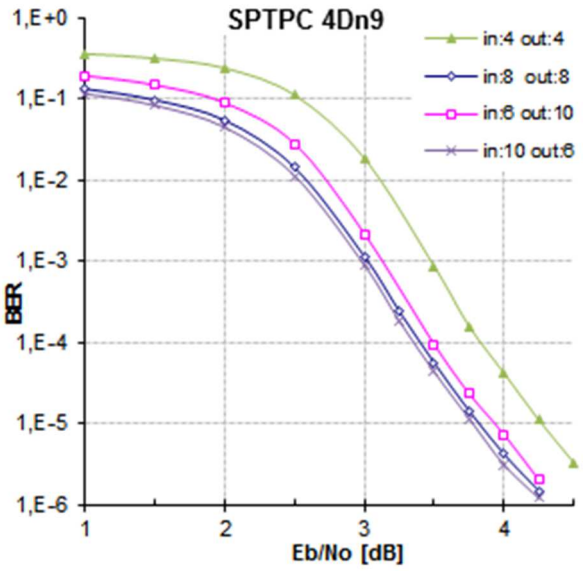
(Signal to Noise Ratio). Ak je SNR kanála nižšia ako hraničná SNR, dekódovanie pomocou danej sekvencie bude divergovať, t.j. počet chybných symbolov s s iteráciami dekódovania bude zvyšovať. Pri nižšej kanálovej SNR treba použiť sekvenciu prahových hodnôt s vyššími hodnotami. Nevýhodou „prísnejšej“ sekvencie môže byť, že sa pri dekódovaní falošne, mylne potvrdia určité symboly a tým sa výkonnosť dekódovania zhorší resp. spomalí. Dôkazom je obrázok 7.5 a výkonnosti dekódovania použitím dvoch sekvencii prahových hodnôt T^1_{5D} and T^2_{5D} pre kód 5Dn8 pri E_b/N_0=2.6dB. Sekvencie sa líšia len v jednej prahovej hodnote pri 10 iterácií.

$$T^1_{5D}\{5,4,3,5,3,4,5,3,4,\mathbf{4},5,4,2,3,5,4,2,3,4,3,2,4,3,3,4,2,3,3,2,3\}$$
$$T^2_{5D}\{5,4,3,5,3,4,5,3,4,\mathbf{2},5,4,2,3,5,4,2,3,4,3,2,4,3,3,4,2,3,3,2,3\}$$

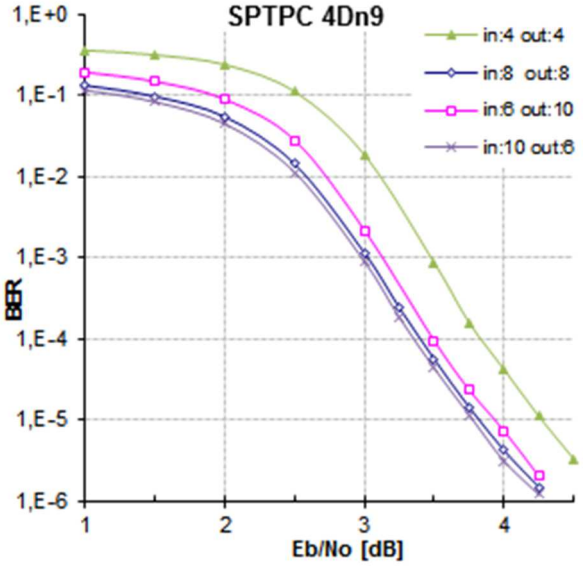
Výkonnosť dekódovacie algoritmu samozrejme závisí aj od počtu kvantizačných úrovní. Čím viac kvantizačných úrovní tým má hybridné dekódovanie bližšie k mäkkému dekódovaniu. Výkonnosti dekódovacieho algoritmu v závislosti od počtu kvantizačných úrovní pre kód 4Dn9 v AWGN kanály s použitím BPSK modulácie je na obr. 7.6. Ešte jedna myšlienka, ktorá bola prezentovaná v [JAN15] spočíva v zistení, že algoritmus dekódovania je citlivejší na zvýšený počet kvantizačných úrovní medzi strednými hodnotami. Ak sú kvantizačné úrovne rozdelené nerovnomerne, t.j. sú zhustené medzi strednými hodnotami prijatia symbolov a zjemnené smerom od stredných hodnôt výkonnosť algoritmu rastie. Napr. pri počte kvantizačných úrovní 16, najlepšie výsledky dosahuje algoritmus, ktorý uvažuje s 10 vnútornými a 6 vonkajšími úrovňami a naopak najhoršie výsledky vykazuje algoritmus so 6 vnútornými a 10 vonkajšími úrovňami kvantizácie.



Obrázok 7.5 Výkonnosť 2 rôznych sekvencií prahových hodôt pre 5D kód (8,7,2) v kanály AWGN s mod. BPSK pri E_b/N_0=2.6dB, prevzaté z [JAN15].



Obrázok 7.6 Výkonnosť 4D kódu (9,8,2) v kanály AWGN s mod. BPSK, pre rôzne počty a konštelácie kvantizačných úrovní [JAN15].



Obrázok 7.7 Výkonnosť 4D kódu (9,8,2) v kanály AWGN s mod. BPSK, pre rôzne počty a konštelácie kvantizačných úrovní [JAN15].

Publikácia [JAN15] bola citovaná, ako inovačný príspevok v publikácii [MUK16] zameranej na zhrnutie najvýznamnejších modifikácií dekódovacích algoritmov blokových kódov násobkov.

8. Aplikácie viacrozmerných kódov násobkov

Blokové kódy násobky majú dnes široké použitie v komunikačných systémoch. Aplikovanie samoopravných kódov zložených z kódov násobkov, prípadne ich modifikácii v súčasných komunikačných systémoch je prehľadne zhrnuté v [MUK16].

Ako jeden z cieľov dizertačnej práce boli nájsť ďalšie možné aplikovanie kódov násobkov. V rámci výskumu tejto dizertačnej práce sme sa sústredili na

8.1 Zostrojenie kontrolnej matice kódov násobkov

Kódy s obmedzenými dĺžkami behov (Run-Length Limited Codes, RLL) sú kódy, ktoré majú obmedzenia na dĺžky sekvencií zložených z rovnakých symbolov v kódových slovách. Tieto obmedzenia sú podnietené praktickým využívaním kódov. Kódy v komunikačných systémoch okrem samoopravných vlastností by mali vykazovať aj RLL vlastnosti. RLL kódy majú dnes široké využitie v komunikačných systémoch a v systémoch úložiska dát. Dôležitosť aplikovania RLL kódov bola podložená udelením „IEEE Medal of Honor v roku 2017“ K.A.S. Immink-ovi, práve za výskum v oblasti RLL. [IMM04].

V praktických aplikáciách sa využívajú kaskádne spojenie RLL kódov so samoopravnými kódmi. Dôvodom je, že RLL kódy samé o sebe nie sú schopné opravovať chybné symboly. Práve kaskádne spojenie RLL a samoopravných kódov (ECC) zabezpečí potrebné výsledné vlastnosti kódových slov. Krátky prehľad vývoja RLL-ECC kódov a súčasný stav výskumu je v [FAR17a] a [FAR17b]. Publikácie poskytujú aj potrebné podklady pre hlbšie porozumenie pozadia konštrukcie a tried RLL kódov.

V rámci výskumu sme našli spôsob ako aplikovať RLL vlastnosti na blokové kódy násobky [FAR18a] a [FAR18b]. Budeme uvažovať s 2D,3D a viacrozmernými kódmi násobkami, opísanými v kapitole 3.2, viď obrázky 3.2, 3.3, 3.4. Grafickú reprezentáciu kódových slov vieme vyjadriť kontrolnou maticou H nasledovne:

$$\mathbf{H} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,N} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ h_{(N-K),1} & h_{(N-K),2} & \cdots & h_{(N-K),N} \end{bmatrix} \qquad (8.1)$$

Konštrukciu H matice vysvetlíme pre 2D kód. Pre každé kódové slovo v 2D kóde existuje generujúca matica aj kontrolná matica $\mathbf{H}_c$, pre ktorú platí.

$$\mathbf{c}\mathbf{H}_c^T = \mathbf{0} \qquad (8.2)$$

Zostrojenie kontrolnej matice $\mathbf{H_C}$ prebieha vo viacerých krokoch. Prvým krokom je vybrať bijektívne mapovanie medzi každým kódovým slovom v 2D kóde a každým stĺpcom matice $\mathbf{H_C}$. Index $j = 1,\ 2,\ \cdots,\ N$, (pre prípad $N = n_1 \times n_2$) bude použitý indexovanie stĺpcov matice $\mathbf{H_C}$ a symbolov v kódovom slove.

V druhom kroku nasleduje bijektívne mapovanie medzi riadkami a stĺpcami všetkých kontrolných rovníc jednotlivých kódov komponentov.

V ďalšom kroku každý symbol 1 alebo 0 je priradený každému prvku matice $\mathbf{H_C}$ podľa nasledovného pravidla. Ak je symbol s indexom i v riadku j prítomný v kontrolnej rovnici kódu komponentu daného 2D kódu, je prislúchajúci element v matici $\mathbf{H_C}$ rovný 1 inak 0.

V prípade, ak potrebujeme aj kontrolné symboly kontrolných symbolov, kontrolné rovnice budú zahrnuté dvakrát, aj pre kódové slova v riadkoch aj stĺpcov, a teda následne musia byť v matici **H** identifikované a zmazané.

Pre názornosť ukážeme ako vytvoriť maticu $\mathbf{H_C}$ pre 2D Hammingov kód z obrázku 8.1.

c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}
c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}
c_{22}	c_{23}	c_{24}	c_{25}	c_{26}	c_{27}	c_{28}
c_{29}	c_{30}	c_{31}	c_{32}	c_{33}	c_{34}	c_{35}
c_{36}	c_{37}	c_{38}	c_{39}	c_{40}	c_{41}	c_{42}
c_{43}	c_{44}	c_{45}	c_{46}	c_{47}	c_{48}	c_{49}

Obrázok 8.1 2D kód násobok zložený s Hamm. kódu (7,4,3)

Uvažujeme teda s 2D kódom z obrázku 8.1. Kontrolná matica Hammingovho kódu z prvého riadku je daná nasledovne:

$$\mathbf{H}_1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \tag{8.3}$$

V kompaktnejšej podobne môže byť vyjadrená nasledovne:

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{h_1} & \mathbf{h}_2 & \mathbf{h}_3 & \mathbf{h}_4 & \mathbf{h}_5 & \mathbf{h}_6 & \mathbf{h}_7 \end{bmatrix}, \tag{8.4}$$

kde $\mathbf{h}_t$; $t = 1, \ldots, 7$ sú stĺpce matice $\mathbf{H}_1$ z (8.4).

Ak označíme kódové slovo 2D kódu nasledovne:

$$\mathbf{c} = (c_1, \ c_2, \ \ldots, \ c_{49}) \tag{8.5}$$

tak kontrolu matica 2D kódu vieme vyjadriť nasledovne:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 \\ \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_{13} & \mathbf{H}_{14} & \mathbf{H}_{15} & \mathbf{H}_{16} & \mathbf{H}_{17} \end{bmatrix}, \tag{8.6}$$

kde

$$\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8.7}$$

a

$$\mathbf{H}_{1t} = \begin{bmatrix} \mathbf{h}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t \\ \mathbf{0}_t & \mathbf{h}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t \\ \mathbf{0}_t & \mathbf{0}_t & \mathbf{h}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t \\ \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{h}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t \\ \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{h}_t & \mathbf{0}_t & \mathbf{0}_t \\ \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{h}_t & \mathbf{0}_t \\ \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{0}_t & \mathbf{h}_t \end{bmatrix} \tag{8.8}$$

$$\mathbf{0}_t = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{8.9}$$

V (8.7), $\mathbf{H_C}$ je matica o rozmere 42×49. Ďalším pozorovaním tejto matice si zistíme, že horných 21 (7x3) riadkov matice zodpovedá kontrolným rovniciam riadkov, kde každý riadok má 3 rovnice. Zvyšných 21 riadkov zodpovedná kontrolným rovniciam 7 stĺpcov 2D kódu, po

3 rovnice pre každý stĺpce. Zjednotenie týchto množín rovníc zahŕňa všetkých 49 symbolov v kódovom slove..

Pre viacrozmerný kód násobok, ktorý pozostáva z kódov s parametrami: $[n_1,\ k_1,\ d_{m1}]$, $[n_2,\ k_2,\ d_{m2}]$, … …,$[n_M,\ k_M,\ d_{mM}]$ postup zostrojenia **H** je obdobný. Uvažujme, jeden z kódov komponentov s parametrami $[n_i,\ k_i,\ d_{iM}]$. Konštrukciu RLL kódu začneme vpísaním všetkých kontrolných rovníc pre všetky riadky v jednom rozmere. (Pre 3D kód si to môžeme predstaviť pre všetky riadky v smere X). Počet rovníc by bol daný:

$$(n_i - k_i).(n_1 \times n_2 \times \ldots \times n_{(i-1)} \times n_{(i+1)} \times \ldots \times n_M).$$

V ďalšom kroku budeme pokračovať obdobne v druhom rozmere. Pre ďalšie rozmery už symboly nenasledujú v poradí ale sú "prekladané" a sú vyjadrené maticami $\mathbf{H}_1$, $\mathbf{H}_2$, …, $\mathbf{H}_{(i-1)}$, $\mathbf{H}_{(i+1)}$, …, $\mathbf{H}_M$, ktoré označíme $i\{\mathbf{H}_{1,j}\}$, $i\{\mathbf{H}_{2,j}\}$, …, $i\{\mathbf{H}_{(i-1),j}\}$, $i\{\mathbf{H}_{(i+1),j}\}$, …, $i\{\mathbf{H}_{M,j}\}$. (kvôli úspore mieste vyjadríme $(n_i - k_i) \times n_i$ matíc $\mathbf{0}_{i,j}$).

Výsledkom uvedeného postupu bude zostrojenie výslednej kontrolnej matice pre M rozmerov:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_i & \mathbf{0}_{i,j} & \ldots & \mathbf{0}_{i,j} \\ \mathbf{0}_{i,j} & \mathbf{H}_i & \ldots & \mathbf{0}_{i,j} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{i,j} & \mathbf{0}_{i,j} & \ldots & \mathbf{H}_i \\ i\{\mathbf{H}_{1,1}\} & i\{\mathbf{H}_{1,2}\} & \ldots & i\{\mathbf{H}_{1,(N/n_1)}\} \\ i\{\mathbf{H}_{2,1}\} & i\{\mathbf{H}_{2,2}\} & \ldots & i\{\mathbf{H}_{2,(N/n_2)}\} \\ \vdots & \vdots & \vdots & \vdots \\ i\{\mathbf{H}_{(i-1),1}\} & i\{\mathbf{H}_{(i-1),2}\} & \ldots & i\{\mathbf{H}_{(i-1),(N/n_{(i-1)})}\} \\ i\{\mathbf{H}_{(i+1),1}\} & i\{\mathbf{H}_{(i+1),2}\} & \ldots & i\{\mathbf{H}_{(i+1),(N/n_{(i+1)})}\} \\ i\{\mathbf{H}_{(i+2),1}\} & i\{\mathbf{H}_{(i+2),2}\} & \ldots & i\{\mathbf{H}_{(i+2),(N/n_{(i+2)})}\} \\ \vdots & \vdots & \vdots & \vdots \\ i\{\mathbf{H}_{(M-1),1}\} & i\{\mathbf{H}_{(M-1),2}\} & \ldots & i\{\mathbf{H}_{(i+2),(N/n_{(M-1)})}\} \\ i\{\mathbf{H}_{M,1}\} & i\{\mathbf{H}_{M,2}\} & \ldots & i\{\mathbf{H}_{M,(N/n_M)}\} \end{bmatrix} \quad (8.10)$$

Celkový počet stĺpcov v matici (8.10) je $N = (n_1 \times n_2 \times \ldots \times n_M)$ a riadkov

$$\kappa = (n_i - k_i).(n_1 \times n_2 \times \ldots \times n_{(i-1)} \times n_{(i+1)} \times \ldots \times n_M) +$$
$$+ (n_1 - k_1).(n_2 \times n_3 \ldots \times n_M) + (n_2 - k_2).(n_1 \times n_3 \ldots \times n_M) + \ldots,$$
$$+ (n_M - k_M).(n_1 \times n_2 \ldots \times n_{(M-1)})$$

Parametre kódu sú dané $[N,\ K,\ d_{mMD}]$, nasledovne:

$$N = n_1 \times n_2 \ldots \times n_M \tag{8.11}$$

$$K = k_1 \times k_2 \ldots \times k_M \tag{8.12}$$

$$d_{mMD} = d_1 \times d_2 \ldots \times d_M \tag{8.13}$$

Pozorovaním matice (8.10) sa dá zistiť, že obsahuje podmaticu, ktorú využijeme pri konštruovaní RLL vlastností:

$$\mathbf{H}_{cT} = \begin{bmatrix} \mathbf{H}_i & \mathbf{0}_{i,j} & \ldots & \mathbf{0}_{i,j} \\ \mathbf{0}_{i,j} & \mathbf{H}_i & \ldots & \mathbf{0}_{i,j} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{i,j} & \mathbf{0}_{i,j} & \ldots & \mathbf{H}_i \end{bmatrix} \tag{8.14}$$

8.2 Zostrojenie RLL kódov z kódov násobkov

V rámci [FAR18a] sme navrhli ako zostrojiť RLL - samoopravné kódy zo samoopravných kódov, ktorých kontrolne matice spĺňajú nasledovné vlastnosti.

Vlastnosť: Ak kontrólna rovnica lineárneho blokového kódu *C* obsahuje množinu *E* s párnym počtom symbolov, tak invertovanie nepárneho počtu symbolov v množine *E* vo všetkých kódových slovách kódu *C* má za následok vytvorenie nového kódu *C'*, ktorého žiadne kódové slovo neobsahuje len symboly 0 alebo symboly 1. *C'* je lineárnou obmenou (cosset) kódu *C*.

Túto vlastnosť vieme využiť, resp. zduplikovať pre neprekrývajúce sa množiny symbolov. Účelom operácie je nájdenie modifikátora ***m***, ktorý bude pripočítaný ku každému kódovému slovu pred poslaním do kanála alebo uloženia symbolu:

$$\mathbf{c}' = \mathbf{c} + \mathbf{m} \tag{8.15}$$

V (8.15) kódové slovo $\mathbf{c}'$ označuje vyslané alebo uložené kódové slovo výsledného RLL-ECC (samoopravného kódu s obmedzenou dĺžkou behu) kódu. Súčet predstavuje vektorové sčítanie symbolov nad polom $GF(2)$. (inými slovami symboly sú sčítane modulo 2).

Pred samotným dekódovaním musí byť modifikátor ***m*** odstránený. Odstránenie vplyvu modifikátora dosiahneme jeho opätovným pripočítaním k prijatému, resp. uloženému kódovému slovu:

$$\hat{\mathbf{c}} = \mathbf{c}' + \mathbf{m} \tag{8.16}$$

V (8.16) $\hat{\mathbf{c}}$ predstavuje odhadnuté kódové slovo pôvodného samoopravého kódu, ktoré môže obsahovať chyby. Preto na neho bude aplikovaný dekódovací algoritmus popísaný v kapitolách 5.2, resp. 6.1. Dekódovací algoritmus je aplikovaný podľa triedy použitých kódov [PYN94], [RAN01].

Pre názornosť použijeme príklad z [FAR17b]. Hammingov kód $[7,\ 4,\ 3]$ má kontrólnu maticu $\mathbf{H}$ danú podľa (8.3). (V tomto prípade je to aj kontrólna matica $\mathbf{H}_{1c}$.) Každé kódové slovo $\mathbf{c} = (c_6,\ c_5,\ c_4,\ c_3,\ c_2,\ c_1,\ c_0)$ musí spĺňať nasledovné kontrolne rovnice:

$$c_2 + c_3 + c_4 + c_5 = 0 \tag{8.17}$$

$$c_1 + c_3 + c_4 + c_6 = 0 \tag{8.18}$$

$$c_1 + c_2 + c_4 + c_7 = 0 \tag{8.19}$$

V tomto prípade tieto 3 rovnice nie sú disjunktné (neprekrývajúce sa) a preto vyberieme len jednu z nich pre získanie modifikátora kódového slova m. Ak vyberieme prvú rovnicu, t.j. (8.17) zahrňujúcu množinu symbolov $\{c_2, c_3, c_4, c_5\}$, tak modifikátor bude mať nepárny počet jednotkových symbolov v príslušnej množine symbolov, konkrétne na miestach $\{2,3,4,5\}$. Napríklad:

$$\mathbf{m} = (0\quad 0\quad 0\quad 1\quad 0\quad 0\quad 0) \tag{8.20}$$

Podľa [FAR17b] teda sekvencia zreťazených kódových slov výsledného RLL-ECC nemôže obsahovať viac ako 12 identických symbolov v behu.

Veta: Pre každý viacrozmerný kód násobok zložený z paritných kódov (SPC) s parametrami $[n_1 \times n_2 \times \ldots \times n_i \times \ldots \times n_M,\ \ k_1 \times k_2 \times \ldots \times k_i \times \ldots k_M,\ \ 2^M]$, v ktorom sú kódové slová párnej dĺžky (označenie n_i), vieme skonštruovať RLL-ECC kód využitím modifikátora m podľa definície v [FAR17b]. Výsledný RLL-ECC kód má rovnaké parametre ako pôvodný kód a súčasne platí, že neobsahuje viac ako $2(n_i - 1)$ identických symbolov v akýchkoľvek po sebe idúcich kódových slovách.

Dôkaz: Je zrejmé, že ak aspoň jeden kód komponent, ktorý má párnu dĺžku kódového slova (n_i), tak kontrólna matica pre viacrozmerný SPC kód bude mať $n_1 \times n_2 \times \ldots \times n_{i-1} \times n_{i+1} \times \ldots \times n_M$ riadkov, ktorý každý z nich obsahuje párny počet n_i po sebe nasledujúcich symbolov 1. (Napríklad použitím postupu v kapitole 8.1.1, mapovaním kódových slov s párnym počtom symbolov do prvých riadkov matice $\mathbf{H}_c$ bez prekrývania. [FAR18a]). Na základe dôkazu v [FAR17b], v každej s neprekrývajúcich množín môže byť znegovaných nepárny počet symbolov použitím logickej 1 v modifikátore m. Tým pádom najdlhší beh identických symbolov predstavuje dĺžku $2(n_i - 1)$ pre akékoľvek za sebou nasledujúce kódové slová.

8.3 Aplikovanie na ľubovoľné lineárne blokové kódy

Na základe konštrukcií RLL uvedených v [FAR17b] [FAR18a] vieme zovšeobecnením aplikovať tento algoritmus aj na kódy násobky pozostávajúcich z ľubovoľných lineárnych blokových kódov, a teda nie len z jednoduchých paritných kódov.

Veta: Pre ľubovoľný viacrozmerný kód násobok s parametrami $[n_1 \times n_2 \times \ldots \times n_M,\ \ k_1 \times k_2 \times \ldots k_M,\ \ d_1 \times d_2 \times \ldots \times d_M]$ môžeme vytvoriť RLL-ECC kód aplikovaním modifikátora podľa [FAR17b] ak aspoň jeden kód komponent vie byť pretransformovaný na RLL kód. RLL vlastnosti skonštruovaného RLL-ECC kódu budú zhodné s RLL vlastnosťami RLL-ECC kódu komponentu.

Dôkaz: Nech je jednorozmerný kód komponent viacrozmerného kódu násobku, ktorý vieme transformovať na RLL-ECC kód s dĺžkou kódového slova n_i a kontrolnou maticou $\mathbf{H}_i$. Je zrejmé, že kontrólnu maticu viacrozmerného kódu násobku vieme získať v tvare (8.10). V nej sa nachádza N / n_i podmatíc identických ako $\mathbf{H}_i$, ktoré tvoria diagonálnu maticu $\mathbf{H}_{cT}$ danú (8.14). V každom stĺpci sa nenulové symboly nachádzajú len na pozíciách prislúchajúcim jednotlivým podmaticiam $\mathbf{H}_i$. Preto symboly výsledného kódového slova kódu násobku nielen že vykazujú príslušnosť so stĺpcami podmatice $\mathbf{H}_i$, ale navyše môžu byť rozdelené do N / n_i disjunktných množín po n_i symboloch. Disjunktné množiny sú následne zreťazené do výsledného slova. Preto výsledný modifikátor predstavuje zreťazenie jednotlivých modifikátorov pre každú množinu, t.j. pre každú príslušnú časť kódu. RLL vlastnosti viacrozmerného kódu sú potom identické s RLL vlastnosťami jednorozmerného kódu komponentu.

Napríklad, pre vybraný jednorozmerný kód RLL-ECC založený na jednoduchom paritnom kóde dĺžky n_i, najdlhší beh identických symbolov nebude dlhší ako $2(n_i - 1)$ v ľubovoľnej sekvencií po sebe idúcich kódových slov tohto RLL-ECC kódu [FAR18b].

9. Záver a vedecké prínosy práce

V dizertačnej práci ako aj celom doktorandskom štúdiu som sa podrobne venoval algoritmom dekódovania blokových kódov násobkov zložených z lineárnych blokových kódov. Najskôr som podrobne objasnil konštrukciu a kódovanie blokových kódov násobkov. Pre konštrukcie kódov som našiel zovšeobecnenie pre N rozmerov. Pre N-rozmerné kódy zložené z jednoduchých paritných kódov je v kapitole 5.1 tvrdý algoritmus dekódovania, ktorého vlastnosti som následne odsimuloval, výsledky slúžili pre ďalšie porovnania a publikáciu [JAN15]. Pre paritné kódy násobky som zovšeobecnil mäkký algoritmus dekódovania uvedený v [HAG96] tak, aby bol použiteľný na paralelné aj sériové kódy násobky – kapitola 5.2. Zovšeobecnenie algoritmu pre N-rozmerné kódy bolo publikované v [JAN03b]. Ukázal a zdôvodnil som, prečo tento algoritmus nie je vhodný na dekódovanie iných, ako paritných kódov. Následne som naštudoval a zanalyzoval algoritmus, ktorý je takmer optimálny na dekódovanie ľubovoľných kódov násobkov. Jeho opravné schopnosti som odprezentoval na dvoj- a trojrozmerných kódoch. Modifikácie algoritmu pre kódy násobky zložené z jednoduchých paritných kódov boli publikované v [JAN04] a [JAN05].

Výsledky dosiahnutých poznatkov podloženými simuláciami a publikáciami zhrniem do niekoľkých základných bodov. Korekčné schopnosti paritných kódov, pri použití tvrdého dekódovacieho algoritmu sú, pri nízkych rozmeroch (3D,4D), slabé. Zvyšovaním počtu kódov komponentov dochádza k zlepšeniu, je to však na úkor väčšej zložitosti dekodérov. Možnosťou riešenia, ako dosiahnuť lepšie výkonnosti pri použití tých istých kódov je, použiť mäkké dekódovanie. Použitia mäkkého algoritmu prináša, v porovnaní s tvrdým dekódovaním, zisk kódovania približne 2dB pri výstupnej chybovosti 10^{-5}. Hybridné dekódovanie predstavuje kompromis aj v komplexnosti dekódovania aj v dosiahnutých výkonnostiach. Problematike hybridného dekódovania je venovaná kapitola 7. V tejto oblasti som publikoval vedecký príspevok [JAN15], ktorý bol citovaný v článku IEEE COMMUNICATIONS SURVEYS & TUTORIALS zhrňujúcom najväčšie prínosy v oblasti algoritmov dekódovania blokových kódov násobkov [MUK16].

Celkovo možno konštatovať, že dekódovanie kódov násobkov zložených z jednoduchých paritných kódov je výpočtovo aj algoritmicky pomerne jednoduché a hlavne rýchle. Nevýhoda týchto kódov spočíva v tom, že musia mať veľké rozmery (5D,6D) na to, aby vykazovali dobré výsledky. Ak porovnáme kódy 4Dm6 a 6Dm9, ktoré majú približne rovnaké kódové rýchlosti, zistíme, že pri zvyškovej chybovosti 10^{-5}, zisk kódovania kódu 6Dm9 predstavuje približne 1.7dB v porovnaní s kódom 4Dm6. Použitím Hammingových kódov ako kódov komponentov s modifikovaným mäkkým Chaseovým dekódovacím algoritmom sme schopní dosiahnuť dobré výsledky už pri dvojrozmerných kódoch. 2D kód zložený z kódov (32,26,4) dosahuje približne rovnaké výsledky ako 4Dm9, pričom má šesťnásobne kratšiu dĺžku kódového slova. Ešte lepšie výsledky vykazujú 3D rozšírené

Hammingove kódy, ktorých zisk kódovania v porovnaní s 2D kódmi je väčší ako 1.2 dB. Trojrozmerné kódy zložené z Hammingových kódov a ich iteratívne dekódovanie sú vhodnými adeptmi na aplikácie v oblastí zabezpečenej komunikácie [BAL15]. Reálne aplikácie, systémy resp. protokoly často okrem iného kladú nároky na dĺžky kódových slov. Pre dosiahnutie potrebných vlastností sa často pristupuje k skracovaniu, dierovaniu kódových slov. K téma skracovania kódov násobkov bol publikovaný vedecký článok [JAN13].

Ako jeden z cieľov dizertačnej práce bol nájsť možné aplikácie skúmaných kódov násobkov. V kapitole 8 sú popísané možnosti aplikovania blokových kódov násobkov ako kódov s obmedzenou dĺžkou behu identických symbolov (RLL kódy). V publikáciách [FAR18a] a [FAR18b] sme navrhli konštrukcie RLL kódov pre kódy násobky zložené z jednoduchých paritných kódov a tiež z ľubovoľných lineárnych blokových kódov. Získané poznatky majú vedecký a praktický význam v tom, že takto vytvorené samoopravné kódy majú okrem už overených korekčných vlastností aj RLL vlastnosti, vhodné pre praktické aplikácie.

Zosumarizovaním dosiahnutých výsledkov popísaných v tejto práci a zohľadnením publikovaných a citovaných vedeckých príspevkov mienim, že došlo k naplneniu úloh a cieľov, ktoré boli vytýčené v zadaní dizertačnej práci. Tento fakt je podložený citáciou v článku IEEE COMMUNICATIONS SURVEYS & TUTORIALS, ktorý uvádza najväčšie prínosy v oblasti algoritmov dekódovania blokových kódov násobkov [MUK16].

10. Zoznam publikačnej činnosti a ohlasov

AAB Vedecké monografie vydané v domácich vydavateľstvách

AAB01 FARKAŠ, Peter - JANVARS, Tomáš - FARKAŠOVÁ, Katarína - RUŽICKÝ, Eugen. On Run-Length Limited Error Control Codes constructed from Binary Product Codes. *Journal of Electrical Engineering, vol. 69 no.4 , June 2018,* accepted. (alebo po našom: akceptované, v tlači)

AFC Publikované príspevky na zahraničných vedeckých konferenciách

AFC01 FARKAŠ, Peter - TURCSÁNY, Matúš - JANVARS, Tomáš. A Hybrid DSSS/TDMA 2-Layer WSN Architecture with Implicit Addressing and Error Correction Control. In *11th International Symposium on Wireless Personal Multimedia Communications (WPMC'08) : Lapland, Finland, 8.-11.9.2008* : University of Oulu, 2008, s.CD-Rom.

AFC02 JANVARS, Tomáš - FARKAŠ, Peter. Punctured 2D Turbo Product Codes and Their Performance. In *TSP 2013 Telecommunications and Signal Processing : 36th International Conference on Telecommunications and Signal Processing; Rome, Italy, July 2-4, 2013*. Brno : VUT v Brně, 2013, s.259-262. ISBN 978-1-4799-0403-7. V databáze: SCOPUS: 2-s2.0-84885572999 ; IEEE: 13827441 ; WOS: 000333968000052.

AFC03 JANVARS, Tomáš - FARKAŠ, Peter. Hard decision decoding of single parity turbo product code with n-level quantization. In *TSP 2015 : 38th International conference on telecommunications and signal processing. Prague, Czech Republic. July 9-11, 2015*. Brno : Brno University of Technology, 2015, Art. No. 7296433. ISBN 978-1-4799-8498-5. V databáze: SCOPUS: 2-s2.0-84960192125 ; IEEE: 15566771 ; WOS: 000375231000059.

Ohlasy:
1. [1] MUKHTAR, H. - AL-DWEIK, A. - SHAMI, A. Turbo Product Codes: Applications, Challenges, and Future Directions. In IEEE Communications Surveys and Tutorials. Vol. 18, Iss. 4 (2016), s. 3052 - 3069. ISSN 1553-877X., Registrované v: WOS

AFD Publikované príspevky na domácich vedeckých konferenciách

AFD01 FARKAŠ, Peter - JANVARS, Tomáš - FARKAŠOVÁ, Katarína - RUŽICKÝ, Eugen. On run-length limited error control codes constructed from binary single parity check

product codes. In *2018 Cybernetics & Informatics (K&I) [elektronický zdroj] : 29th International Conference. Lazy pod Makytou, Slovakia. January 31-February 3, 2018*. 1. vyd. Bratislava : Slovak Chemical Library, 2018, USB, [4] s. ISBN 978-1-5386-4420-1. V databáze: IEEE: 17715449.

AFD02 JANVARS, Tomáš - FARKAŠ, Peter. Decoding of Single Parity Product Codes Using Improved Chase Algorithm. In *Winterschool on coding and information theory : Proceedings. Bratislava, Slovenská republika, 20.-25.2.2005*. Vienna : Telecomm. Research Institute, 2005, s.119-123.

AFD03 JANVARS, Tomáš - FARKAŠ, Peter. On Decoding of Single Parity Product Codes Using Modified Chase Algorithm. In *SYMPOTIC `04 : Joint IST workshop on mobile future&symposium on trends in communications SYMPOTIC `04. Bratislava, Slovenská republika, 24.-26.10.2004*. Piscataway : IEEE, 2004, s.62-65. V databáze: SCOPUS: 2-s2.0-16244422312 ; IEEE ; WOS: 000226222500019.

AFD04 JANVARS, Tomáš - FARKAŠ, Peter. On Decoding of Multidimensional Single Parity Turbo Product Codes. In *SYMPOTIC `03 : Joint IST workshop on mobile future&symposium on trends in communications SYMPOTIC `03. Bratislava, Slovenská republika, 26.-28.10.2003*. Piscataway : IEEE, 2003, s.25-28. V databáze: SCOPUS: 2-s2.0-84954129626 ; WOS: 000189349700008 ; IEEE: 7907146.

Ohlasy:

1. [1] HUANG, Ying - LEI, Jing. Performance Analysis of Multidimensional Parity Check Product Codes. In Journal of University of Electronic Science and Technology of China, 2010, vol. 39, iss. 2, s.214 - 218., Registrované v: SCOPUS
2. [3] HUANG, Ying - LIU, Taiwei - LEI, Jing. Turbo Product Codes for Holographic Storage. In Control and Automation, 2007, vol. 23, iss. 33, s.1754-1751.
3. [3] ZHANG, T. - MARELLI, A. - MICHELONI, R. Turbo Product Codes. In Codes and Turbo Codes, 2011, vol. 28, s.271-295.
4. [3] HUANG, Ying - LEI, Jing - ZHAO, Chao. Rate adaptive multi-dimensional Turbo Product Code System Design. In Journal Radio Communications Technology, 2007, vol. 33, iss. 1, s.19-22.

11. Zoznam použitej literatúry

[AHM12] Ahmadi, A., Bouanani, F., Ben-Azza, H., Benghabrit, Y., Reduced Complexity Iterative Decoding of 3D-Product Block Codes Based on Genetic Algorithms, Hindawi Journal of Electrical and Computer Engineerig, 2012, Article ID 609650

[ADW09] Al-Dweik, A. J., Goff, S.L., Sharif B. S., A Hybrid Decoder for Block Turbo Codes, *IEEE Transactions on Comm*, vol. 57, No. 5, pp. 1229-1232, May 2009

[ALD09] Al-Dweik, A. J., Sharif B. S., Non-Sequential Decoding Algorithm for Hard Iterative Turbo Product Codes, *IEEE Transactions on Comm*, vol. 57, No. 6, pp. 1545-1549, June 2009

[ALD11] Al-Dweik, A. J., Sharif B. S., Closed-chain error correction technique for turbo product codes, Communications, IEEE Transactions on Vol.: 59 , Issue: 3, pp. 632 – 638

[ARG02] Argon, C., and McLaughlin, S.W., A Parallel Decoder for Low Latency Decoding of Turbo Product Codes, *IEEE Trans. Commun.*, vol. 6, no. 2, pp. 70–72, Jun. 2002

[ARG04] Argon, C., McLaughlin, W., "An efficient Chase decoder for turbo product codes," *IEEE Trans. Commun.*, vol. 52, no. 6, pp. 896–898, Jun. 2004

[AZO12] Azouaoui, A. , Belkasmi,M., A new genetic decoding of Linear Block Codes, Multimedia Computing and Systems (ICMCS), 2012 International Conference on, pp. 1176 – 1182

[BAH74] Bahl, L. R., Cocke, J. , Jelinek, F., and Raviv, J. „Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. Inform. Theory, Vol. IT-20, pages 284-287, March 1974.

[BAL09] Baldi, M., Chiaraluce, F., Cancellieri, G.,, "Finite-Precision Analysis of Demappers and Decoders for LDPC-coded M-QAM Systems", IEEE Trans. On Broadcasting, Vol. 55, No. 2, pp. 239-250, June 2009

[BAL15] Baldi, M., Maturo, N., Chiaraluce, F., Cancellieri, G., Security gap analysis of some LDPC coded transmission schemes over the flat and fast fading Gaussian wire-tap channels, EURASIP Journal on Wireless Communications and Networking, Dec.2015

[BER93] Berrou, C., Glavieux, A., Thitimajshima, P., Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes, IEEE Proc. ICC `93, Geneva, Switzerland, May 1993, pp 1064-1070

[BER96] Berrou, C., Glavieux, A., "Near optimum error correcting coding and decoding: Turbo-Codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996

[BLA83] Blahut, R. E., *Theory and practice of error control codes*, Addison-Wesley, 1983, ISBN 0-201-10102-5

[BOS03] Bosco, G., Montorsi, G., Benedetto, S., "A new algorithm for "hard" iterative decoding of concatenated codes," *IEEE Trans. Commun*., vol. 51, no. 8, pp. 1229–1231, Aug. 2003.

[CHA72] Chase, D., A class of algorithms for decoding block codes with channel measurement information, *IEEE Trans. Info. Theory*, vol IT-18, Jan. 1972, pp. 170-182

[CHA06] Changlong X.,, Ying-Chang, L., and Wing Seng, L., The Design and decoding schemes for shortened turbo product codes, The 17th Annual IEEE International Symposium on

Personal, Indoor and Mobile Radio Communications (PIMRC'06)

[CHA07] Changlong X.,, Ying-Chang, L., and Wing Seng. L., Shortened Turbo Product Codes: Encoding Design and Decoding Algorithm, IEEE Trans. On Vehicular Technology, vol 56, No. 6, Nov. 2007

[CHE02] Chenn., Y., Parhi, K.K., Parallel decoding of interleaved of single parity check turbo product codes, Signal Processing Systems, 2002. (SIPS ,02). IEEE Workshop on 16-18 Oct. 2002 Page(s):27 – 32

[ELI54] Elias, P., "Error-free coding," *IRE Trans. Inf. Theory*, vol. IT-4, no. 4, pp. 29–37, Sep. 1954.

[FAR08] Farkaš, P., Turcsány, M., Janvars, T., A Hybrid DSSS/TDMA 2-Layer WSN Architecture with Implicit Addressing and Error Correction Control. In 11th International Symposium on Wireless Personal Multimedia Communications (WPMC'08): Lapland, Finland, 8.-11.9.2008. University of Oulu, 2008.

[FAR17a] Farkaš, P., Schindler, F., " Construction for obtaining trellis run length limited error control codes from convolutional codes", Journal of Electrical Engineering, vol. 68 no. 5, August 2017, pp. 401–404.

[FAR17b] Farkaš, P., Schindler, F., "Run Length Limited Error Control Codes Construction based on One Control Matrix Property", Journal of Electrical Engineering, vol. 68 no. 4, June 2017, pp. 322–324.

[FAR18a] Farkaš, P., Janvars, T., Farkašová, K., Ružický, E., On run-length limited error control codes constructed from binary single parity check product codes. In CIGÁNEK, J. -- KOZÁKOVÁ, A. 2018 Cybernetics & Informatics (K&I): elektronický zdroj 29th International Conference. Lazy pod Makytou, Slovakia. January 31-February 3, 2018. 1. vyd. Bratislava : Slovak Chemical Library, 2018, ISBN 978-1-5386-4420-1.

[FAR18b] Farkaš, P., Janvars, T., Farkašová, K., Ružický, E., On Run-Length Limited Error Control Codes constructed from Binary Product Codes. Journal of Electrical Engineering, vol. 69 no.4 , June 2018, accepted. (alebo po našom: akceptované, v tlači)

[FEN06] Feng, D., Yaun, J., Rate-Compatible Shortened Turbo Product Codes, Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd Volume 5, 7-10 May 2006 Page(s):2489 – 2493

[FOR66] Forney, G. D., *Concatenated Codes*. Cambridge, MA: MIT Press, 1966.

[GAL62] Gallager, R. G., "Low-density parity-check codes," *IRE Trans. On Information Theory*, pp.21-28, 1962.

[HAG96] Hagenauer, J., et al., Iterative Decoding of Binary Block and Convolutional Codes, IEEE Trans. Info. Theory, vol 42, no. 2, Mar. 1996, pp. 429-45

[HAT01] Hata., M., High-speed and robust error correcting code for future mobile communications of High-Dimensional Discrete Torus Knot, International symposium

WPMC'01, Aalborg, Denmark, September 2001,

[HEY07] He, Y., *and* Ching P.C., Performance Evaluation of Adaptive Two-Dimensional Turbo Product Codes Composed of Hamming Codes, Proceedings of the 2007 IEEE International Conference on Integration Technology March 20 - 24, 2007, Shenzhen, China

[HIR01] Hirst, S. A., B. Honary, B., and Markarian, G., "Fast Chase algorithm with an application in turbo decoding," *IEEE Trans. Commun.*, vol. 49, pp. 1693-1699, Oct. 2001

[IMM04] Immink, K. A. S. "Codes for Mass Data Storage Systems", Shannon Foundation Publisher, 2004.

[JAN03a] Janvars, T., Analýza nových iteratívnych metód dekódovania viacrozmerných kódov násobkov. Diplomová práca. Bratislava : FEI STU, 2003. 69 s.

[JAN03b] Janvars, T., Farkaš, P., On Decoding of Multidimensional Single Parity Turbo Product Codes, IEEE SympoTIC '03, Mobile Future and Symposium on Trends in Communications, Joint First Workshop on 26-28 Oct. 2003

[JAN04] Janvars, T., Farkaš, P., On Decoding of Single Parity Product Codes using modified Chase algorithm, IEEE SympoTIC '04, Mobile Future and Symposium on Trends in Communications, Joint Workshop on 26-28 Oct. 2004

[JAN05] Janvars, T., Farkaš, P., Decoding of Single Parity Product Codes using Chase Algorithm, Winterschool on Coding and Information Theory, Bratislava, Slovakia, February 20-25, 2005

[JAN13] Janvars, T., Farkaš, P., Punctured 2D Turbo Product Codes and Their Performance. In TSP 2013 Telecommunications and Signal Processing: 36th International Conference on Telecommunications and Signal Processing; Rome, Italy, July 2-4, 2013. Brno : VUT v Brně, 2013, s. 259--262. ISBN 978-1-4799-0403-7

[JAN15] Janvars, T., Farkaš, P., Hard decision decoding of single parity turbo product code with n-level quantization. In TSP 2015: 38th International conference on telecommunications and signal processing. Prague, Czech Republic. July 9-11, 2015.Brno : Brno University of Technology, 2015, ISBN 978-1-4799-8498-5.

[LEE12] Lee, I., Jung. J., Oh, D., Kim, M., Rate - compatible turbo product codes with non - symmetry block codes for satellite return link transmission technology ICT Convergence (ICTC), 2012 International Conference on Digital Object, pp. 415 - 419

[LER11] Leroux. C., Jego, C., Adde, P., Gupta, D., Jezequel, M., Turbo Product Code Decoder Without Interleaving Resource: From Parallelism Exploration to High Efficiency Architecture, Journal of Signal Processing Systems 64, 1. 2011 pp. 17-29

[MAC96] MacKay, D.J.C., and Neal, R. M., Near Shannon Limit Performance of Low Density Parity

Check Codes, Electronics Letters, July 1996

[MAC99] MacKay, D.J.C., "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, Vol. 45, No.2, March 1999, pp. 399-431.

[MAX06] Ma, X.R.; Xu, Y.Y.; Iterative decoding of parallel and serial concatenated single parity check product codes, Electronics Letters Volume 42, Issue 15, 20 July 2006 Page(s):869 – 870

[MOR02] Morelos-Zaragoza, R. H., The Art of Error Correcting Coding. Hoboken, NJ: Wiley, 2002.

[MUK16] Mukhtar, H., Al-Dweik, A., Shami, A., Turbo Product Codes: Applications, Challenges, and Future Directions, IEEE Communications Surveys & Tutorial, Vol. 18, No. 4, Fourth Quarter 2016, pp. 3052-3069

[PET61] Peterson, W.W., Error-correcting codes, M.I.T Press, 1961

[PRO95] Proakis, J. G., Digital Communicatin, 3rd edition, McGraw Hill, 1995, ISBN 0-07-051726-6

[PYN94] Pyndiah, R., Glavieux, A., Picart, A., and Jacq, S., "Near optimum decoding of product codes," in *Proc. GLOBECOM*, San Francisco, CA, Nov. 1994, vol. 1, pp. 339–343.

[PYN98] Pyndiah, R. M., Near-Optimum Decoding of Product Codes : Block Turbo Codes, IEEE Trans. Commun. , vol. 46, no. 8, Aug. 1998

[RAN01] Rankin, D. M., and Gulliver, T. A., "Single parity check product codes," *IEEE Trans. Commun.*, vol. 49, no. 8, pp. 1354–1362, Aug. 2001.

[RAN03] Rankin, D. M., Gulliver, T. A., and Taylor, D. P., "Asymptotic performance of single parity-check product codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 9, pp. 2230–2235, Sep. 2003.

[SKL88] Sklar, B., Digital Communication , Prentice Hall, 1988, ISBN 0-13211-939-0

[SKL97] Sklar, B., A primer on Turbo Code Concepts, IEEE Communication Magazine, December 1997

[YIN07] Ying-Chang, L., Changlong X., Wing Seng. L., A Low Complexity Decoding Algorithm for Turbo Product Codes, Radio and Wireless Symposium, 2007 IEEE 9-11 Jan. 2007 Page(s):209 - 212

[YIN08] Ying-Chang, L., Changlong X., Wing Seng. L., A Low Complexity Decoding Algorithm for Turbo Product Codes, IEEE Wireless. Commun., vol. 7, No.1, pp. 43-47, Jan. 2008

[ZHI12] Zhiping, S., Liang, Z., Xiang, F., Yang, Y., HIHO Decoding Algorithms of Turbo Product Codes for High Rate Optical Communications, China Communications 2012, Vol. 9, pp. 114-123

12. Zoznam použitých skratiek

AWGN	Additive White Gaussian Noise
BCH	Bose, Ray-Chaudhuri, Hocquenghem
BCJR	Bahl, Cock, Jelinek, Raviv
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BSC	Binary symmetric channel
BTC	Block Turbo Code
CCEP	Closed Chain Error Pattern
CTC	Convolutional Turbo Code
GSM	Global System for Mobile communications
IEEE	Institute of Electrical and Electronics Engineers
HIHO	Hard Input Hard Output
LBK	Lineárny blokový kód
LDPC	Low density parity check
LLR	Log – Likelihood Ratio
MAP	Maximum a posteriori
MLD	Maximum Likelihood Detection
PCC	Paralel concatenated code
RS	Reed Solomon
RLL	Run Length Limited
SCC	Serial concatenated code
SISO	Soft Input Soft Output
SNR	Signal to Noise Ratio
SPTPC	Single Parity Turbo Product Code
TPC	Turbo Product Code