VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

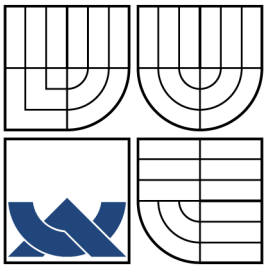
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

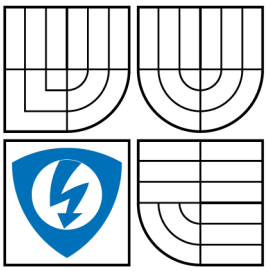
BCH KÓDY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JAKUB FROLKA
AUTHOR



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

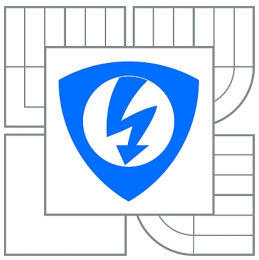
BCH KÓDY
TITLE OF STUDENT'S THESIS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAKUB FROLKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAKUB ŠEDÝ,
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. Jakub Frolka	***ID:***	110408
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

BCH kódy

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku protichybového zabezpečení pomocí BCH kódů a zaměřte se především na zabezpečení a dekódování zabezpečené zprávy zmíněnými kódy. V programovém prostředí Matlab vytvořte program, který bude demonstrovat zabezpečovací proces a výpočty v Galoisově tělese a dekódování zabezpečené zprávy jednotlivými algoritmy. Program bude koncipován jako výuková pomůcka a rovněž jako analyzační nástroj pro porovnání dosažených parametrů kódu a bude obsahovat grafické rozhraní doplněné nápovědou. S pomocí vytvořeného programu realizujte zevrubné srovnání dosažených parametrů (kódového zisku a dalších) pro různé parametry kódu.

DOPORUČENÁ LITERATURA:

[1] LIN, S., COSTELLO, D.J. Error Control Coding: Fundamentals and Applications, second edition. Prentice Hall: Englewood Cli_s, NJ, 2005, ISBN: 0130426725.

[2] MOON, T.K. Error Correction Coding: Mathematical Methods and Algorithms. Wiley-Interscience, 2005, ISBN: 0471648000.

[3] SWEENEY, P. Error Control Coding: From Theory to Practice. Wiley-Interscience, 2002, ISBN: 047084356X.

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

Vedoucí práce: Ing. Jakub Šedý
Konzultanti diplomové práce:

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

ABSTRAKT

Práce se zabývá problematikou zabezpečení dat pomocí BCH kódů. V práci jsou popsány BCH kódy v binární i nebinární podobě a jejich nejvýznamnější podskupina RS kódy. Dále jsou v práci popsány dekódovací metody Peterson-Gorenstein-Zierlův, Berlekamp-Masseyův a Euklidův algoritmus. Pro prezentaci postupu kódování a dekódování, byla vytvořena aplikace v prostředí Matlab, která má dvě části – Výuka BCH kódů a Simulace BCH kódů. Jako poslední část práce byla srovnána výkonnost BCH kódů pomocí vytvořené simulační aplikace.

KLÍČOVÁ SLOVA

BCH kód, Reed-Solomonův kód, Berlekamp-Masseyův algoritmus, Euklidův algorimus, Peterson-Gorenstein-Zierlův algoritmus, Matlab

ABSTRACT

The work deals with data security using BCH codes. In the work are described BCH codes in binary and non-binary form, and their most important subclass RS codes. Furthermore, this work describes the method of decoding Peterson-Gorenstein-Zierl, Berlekamp-Massey and Euclidean algorithm. For the presentation of encoding and decoding process, the application was created in Matlab, which has two parts – Learning BCH codes and Simulation of BCH codes. Using the generated application performance of BCH codes was compared at the last part of the work.

KEYWORDS

BCH code, Reed-Solomon code, Berlekamp-Massey algorithm, Euclidean algorithm, Peterson-Gorenstein-Zierl algorithm, Matlab

FROLKA, Jakub *BCH kódy*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 58 s. Vedoucí práce byl Ing. Jakub Šedý,

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „BCH 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. Jakubu Šedému za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Také bych chtěl poděkovat celé rodině, která mě celou dobu plně podporovala.

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

Tato diplomová práce se zabývá problematikou BCH kódů. Cílem práce je prostudovat problematiku zabezpečení a dekódování zabezpecené zprávy, následně vytvořit aplikaci v programu Matlab, pro demonstraci zabezpečovacího a dekódovácího procesu.

Práce je rozdělena do několika částí, v první části je uvedena teorie, která má za úkol seznámit s potřebnými znalostmi v oblasti protichybového zabezpečení. Po obecném popisu BCH kódů je uveden přehled nezbytných znalostí lineární algebry, popis Galoisova tělesa a operacemi s ním spojené. Dále jsou rozebrány vlastnosti binárních i nebinárních kódů, postup jejich zakódování a jejich dekódovací algoritmy. V této práci jsou popsány dekódovací algoritmy a to Berlekamp-massey, Peterson-Gorenstein-Zierler a Euklidův. Jsou zde také popsány Reed-Solomonovy kódy, které jsou speciální podskupinou BCH kódů. Na konkretním příkladě RS kódu je předveden proces zakódování i dekódování. Na konci práce je popsána podle požadavků zadání, vytvořená aplikace s grafickým rozhraním v programu Matlab, která je vhodná jako výuková pomůcka a také jako analyzační nástroj.

1 OBECNÝ POPIS BCH KÓDŮ

Tématem této kapitoly je popsat tzv. BCH kódy, rozvést jejich vlastností, uvést jejich použití v současných spojovacích zařízení. Bose–Chaudriho–Hocquenghem kódy jejich název vychází ze jmen jejich autorů, používáme pro ně zkratku BCH kódy. Tyto kódy jsou řazeny mezi lineární blokové cyklické kódy založeny na Hammingových kódech, oproti kterým BCH kódy opravují více chyb. BCH kódy byly objeveny A. Hocquenghemem v roce 1959 a nezávisle Bosem a Chaudhurim v roce 1960 [5]. Cyklická struktura těchto kódu byla dokázána Petersonem v roce 1960. Nejjednodušší variantou těchto kódů je binární, díky ní mohou být BCH kódy jednoduše implementovány v digitálních zařízeních. Jednou z jejich dobrých vlastností je velká volitelnost jejich parametrů, dobrý vztah mezi počtem informačních znaků, počtem opravovaných chyb a detailně vypracované dekódovací metody [1]. První dekódovací algoritmus byl navržen W. W. Petersonem v roce 1960. Následně byly vytvořeny další dekódovací algoritmy pány Berlekampem, Chienem, Forneyem, Masseyem a dalšími [5]. Speciálním případem BCH kódů jsou RS kódy (Reed–Solomonovy kódy). Jsou důležité např. proto, že pomocí nich lze konstruovat velmi účinné binární kódy [1].

1.1 Použití BCH kódů v současných spojovacích zařízeních

BCH kódy jsou korekční kódy, které jsou využívány pro zabezpečení dat při jejich přenosu. Jejich použití je jednoznačné, jsou využívány v systémech , které spolu komunikují a nebo také přenášejí mezi sebou data. Chyby vzniklé při přenosu dat přes přenosový kanál nebo ukládáním na zálohovací média mohou být opraveny pomocí BCH kódům. Zvláštní podskupinou BCH kódů jsou kódy nazývané Reed-Solomonovy kódy, které nezabezpečují data po jednotlivých bitech, ale celých bytech (tzv. symbolech). Tyto kódy jsou v dnešní době hojně využívány při zabezpečení dat. BCH kódy jsou využívány například pro zabezpečení ukládaných dat na optických nosičích CD, DVD a Blu-ray. V diskových polích v zapojení jako RAID 6, v modemech xDSL, v digitální satelitní televizi (standard DVB-S2). A dále například také u kódování videa ve videokonferencích (doporučení ITU-T H.261).

1.2 Zařazení BCH kódů a význam jednotlivých pojmů

Uvedli jsme, že BCH kódy patří do skupiny lineárních blokových cyklických kódů, pracující s binárními daty a jsou účinné zabezpečovací kódy přenosu v komunikačních systémech. Pro lepší popsání jejich vlastností budou uvedeny základní vlastnosti lineárních blokových cyklických kódů. Vycházíme z informací uvedených v [8].

1.2.1 Blokové kódy

Podle [8] mají blokové kódy přesně určená rozložení informačních a zabezpečovacích míst v kódové kombinaci. Každá kódová kombinace systematického kódu o délce n, obsahuje k informačních a $r = n - k$ zabezpečovacích prvků, mluvíme o (n, k) kódu. Vždy platí, že $n > k$.

1.2.2 Lineární kódy

Libovolnou kódovou kombinaci lineárního kódu můžeme odvodit jako lineární kombinaci z ostatních kódových informací (využívá se lineární algebry). Bývají zadány pomocí vytvářecí matice G o k řádcích a n sloupcích. Jako řádky matice G jsou použity libovolné lineárně nezávislé kombinace.

1.2.3 Cyklické kódy

Cyklické kódy jsou druhem lineárních kódů (n, k), u kterých jsou ve vytvářecí matici G jednotlivé řádky obsahující stejné prvky, jen posunuty o jedno místo v jednom směru. Tyto kódy jsou zadány tzv. generujícím (vytvářecím) mnohočlenem $G(x)$. Podle řádu mnohočlenu je určen počet zabezpečovacích prvků $r = (n - k)$. Cyklické kódy mají v kódové kombinaci o délce n prvků na prvních k místech prvky nezabezpečené zprávy a na zbývajících r místech zabezpečovací prvky.

1.3 Přehled důležitých znalostí z lineární algebry

V této části kapitoly budeme vycházet z informací uvedených v [1, 8, 9]. Operace spojené se zabezpečením signálových prvků BCH kódy se provádí pomocí prvků

tzv. Galoisova tělesa $GF(p^r)$, kde p je základ číselné soustavy a r je stupeň rozšíření Galoisova tělesa. Je tvořeno konečným počtem prvků $n = p^r$, což odpovídá celkovému počtu kódových prvků v mnohočlenu zabezpečené zprávy a vznikne rozšířením konečného tělesa Z_p.

1.3.1 Konečné těleso Zp

je algebraická struktura určená počtem svých prvků a to je mocnina prvočísla p. Tato struktura je tvořena zbytky po dělení celých kladných čísel prvočíslem p, kde p určuje základ soustavy. V praxi je významné konečné těleso Z_2 tvořené množinou prvků 0, 1. Algebraická operace součtu „$\oplus$“ je známá operace nonekvivalence (XOR) a násobení „$\bullet$“ jako logický součin (AND). V tělese Z_2 je součet i rozdíl dvou prvků stejná operace. Operace prováděné nad konečným tělesem musí splňovat asociativní, komutativní a distributivní zákony. Pro výpočty v binárním tvaru využíváme modulo-2 součet a násobení, tyto operace jsou podobné jako obyčejná aritmetika jen s tím rozdílem, že číslo 2 považujeme za rovno 0 (tj. $1 + 1 = 2 = 0$) [5].

1.3.2 Galoisovo těleso $GF(p^r)$

vznikne rozšířením konečného tělesa stupněm r. Rozšířením vznikne množina vektorů o r prvcích, kde každý prvek je ze Z_p. Celkový počet prvků Galoisova tělesa je p^r. Pro binární kódy se používá Galoisovo těleso $GF(2^r)$, které vznikne rozšířením tělesa Z_2. Toto těleso můžeme vyjádřit pomocí zbytků po dělení mnohočlenu ze Z_2 určitým mnohočlenem (vytvářecím mnohočlenem), pro který podle [9] platí:

- Je nerozložitelný v uvažovaném okruhu mnohočlenů
- je stupně r
- dělí beze zbytku $(x^n - 1)$ a žádný jiný dvojčlen nižšího stupně tohoto tvaru.

Např.: Galoisovo těleso $GF(2^4)$ je množina čtveřic prvků ze Z_2. Určené pomocí mnohočlenu z rozkladu dvojčlenu
$(2^{15} + 1) = (x + 1) \cdot (x^2 + x + 1) \cdot (x^4 + x^3 + x^2 + x + 1) \cdot (x^4 + x + 1) \cdot (x^4 + x^3 + 1)$
Při určování Galoisova tělesa $GF(2^4)$ se hodí pouze mnohočleny stupně r (zde $r = 4$). Zvolený mnohočlen se nazývá vytvářecí mnohočlen $GF(2^4)$ a zpravidla se označuje $G(x)_{GF}$.

1.3.3 Algebraické operace nad Galoisovým tělesem:

Součet „$\oplus$“ se provádí jako sčítání po členech mnohočlenu: $A(x) \oplus B(x) = C(x)$, $c_1(x) = a_1(x) \oplus b_1(x)$, kde $A(x)$, $B(x)$, $C(x)$ jsou mnohočleny vytvářející prvky $GF(p^r)$.
Násobení „$\bullet$“ je určeno jako zbytek po dělení součinu dvou prvků vytvářecím mnohočlenem $GF(2^4)$, tj $G(x)_{GF}$.

V Galoisově tělese se nenulové prvky vyjadřují ve tvaru mocnin α^0, α^1, α^2,..., α^{r-1}. Mocnina určuje prvek buďto přímo, nebo prvek vyjádříme pomocí zbytku R[i] po dělení tohoto prvku s „velkým“ exponentem vytvářecího mnohočlenu tělesa $G(x)_{GF}$ za předpokladu, že prvek α je primitivní. Tato podmínka je zajištěná, použijeme-li pro generování $GF(2^r)$ primitivní mnohočlen.

1.3.4 Příklady výpočtů s použitím Galoisova tělesa

V této kapitole uvedeme příklady použití operací nad $GF(2^4)$, pro jednotlivé výpočty, budeme využívat prvky Galoisova tělesa, které bylo generováno polynomem $G(X)_{GF} = 1 + X + X^4$ viz tab.1.1. Více příkladu uvedeno např. v [5].

Mějme následující lineární rovnice nad $GF(2^4)$ jejíž prvky jsou uvedeny v tab.1.1

$$\begin{aligned} X + \alpha^7 Y &= \alpha^2, \\ \alpha^{12} X + \alpha^8 Y &= \alpha^4. \end{aligned} \tag{1.1}$$

Vynásobením druhé rovnice α^8 dostaneme

$$X + \alpha^7 Y = \alpha^2,$$

$$X + \alpha^{11} Y = \alpha^7.$$

Sečtením těchto rovnic dostaneme

$$(\alpha^7 + \alpha^{11}) Y = \alpha^2 + \alpha^7,$$

$$\alpha^8 Y = \alpha^{12},$$

$$Y = \alpha^4.$$

Dosazením $Y = \alpha^4$ do první rovnice z 1.1 dostaneme $Z = \alpha^7$. Druhým způsobem jak vypočítat tyto rovnice je využitím Kramerova pravidla.

Tab. 1.1: Galoisovo těleso generované $G(X)_{GF} = 1 + X + X^4$

Exponenciální tvar	Polynomiální tvar	Binární tvar
0	0	0 0 0 0
1	1	1 0 0 0
α	α	0 1 0 0
α^2	α^2	0 0 1 0
α^3	α^3	0 0 0 1
α^4	$1 + \alpha$	1 1 0 0
α^5	$\alpha + \alpha^2$	0 1 1 0
α^6	$\alpha^2 + \alpha^3$	0 0 1 1
α^7	$1 + \alpha + \alpha^3$	1 1 0 1
α^8	$1 + \alpha^2$	1 0 1 0
α^9	$\alpha + \alpha^3$	0 1 0 1
α^{10}	$1 + \alpha + \alpha^2$	1 1 1 0
α^{11}	$\alpha + \alpha^2 + \alpha^3$	0 1 1 1
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	1 1 1 1
α^{13}	$1 + \alpha^2 + \alpha^3$	1 0 1 1
α^{14}	$1 + \alpha^3$	1 0 0 1
$\alpha^{15} = \alpha^0$	$\alpha^0 = 1$	1 0 0 0

Dalším příkladem si ukažme jak vyřešit rovnici

$$f(X) = X^2 + \alpha^7 X + \alpha = 0$$

Definovanou nad $GF(2^4)$. K výsledku se dostaneme jednoduchým způsobem, nahradíme-li prvky z tab.1.1 za neznámou X, zjistíme, že $f(\alpha^6) = 0$ a $f(\alpha^{10}) = 0$, protože

$$f(\alpha^8) = (\alpha^6)^2 + \alpha^7 \cdot \alpha^6 + \alpha = \alpha^{12} + \alpha^{13} + \alpha = 0,$$

$$f(\alpha^{10}) = (\alpha^{10})^2 + \alpha^7 \cdot \alpha^{10} + \alpha = \alpha^5 + \alpha^2 + \alpha = 0.$$

Proto prvky α^6 a α^{10} jsou kořeny $f(X)$, a $f(X) = (X+\alpha^6)\cdot(X+\alpha^{10})$. Tyto výpočty jsou typické pro kódování a dekódování BCH a Reed-Solomonových kódů.

2 BINÁRNÍ BCH KÓDY

V této kapitole popíšeme základní vlastnosti binárních BCH kódů. Při návrhu binárních BCH kódů potřebujeme navrhnout vytvářecí mnohočlen $g(x)$, podle [5] se vychází z Bose-Chaudhuriho teorému.

Po zvolení primitivního mnohočlenu a sestrojení Galoisova tělesa $GF(q^m)$, jsou určeny minimální mnohočleny $m_j(x)$ pro α^j, kde $j = 1, 2, \ldots, 2t$, a α je symbol Galoisova tělesa. Vytvářecí mnohočlen je dán nejmenším společným násobkem (LCM) minimálních mnohočlenů jak je uvedeno v následující rovnici:

$$g(x) = LCM[m_1(x) + m_2(x) + \cdots + m_{2t-1}(x)] \tag{2.1}$$

BCH kódy jsou podle [5] definovány s následujícími parametry:

Bloková délka:

$$n = 2^m - 1, \tag{2.2}$$

Počet informačních bitů v kodovém slově:

$$k \geq n - mt, \tag{2.3}$$

Minimální vzdálenost:

$$d_{min} = 2t + 1. \tag{2.4}$$

Důležitým parametrem kódů je také informační rychlost kódu definovaná jako

$$R = \frac{k}{n} \tag{2.5}$$

2.1 Kodér BCH kódu

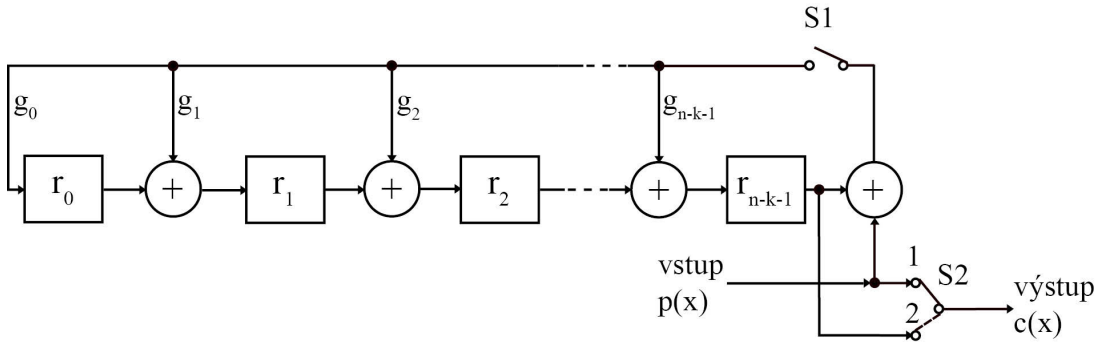
BCH kódy patří do skupiny cyklických kódů, proto jejich kodér můžeme realizovat obvyklým způsobem pro cyklické kódy, tj. pomocí děličky mod $g(x)$ [1, 8] ve které je vytvořen mnohočlen zabezpečené zprávy $c(x)$ [13].

Postup kódování můžeme shrnout do tří kroků:

1. Vynásobení mnohočlenu nezabezpečené zprávy $p(x)$ členem $x^{(n-k)}$, neboli zprávu $p(x)$ rozšíříme o $(n-k)$ nul a tím zvýšíme její řád.
2. Rozšířený mnohočlen zprávy vydělíme vytvářecím mnohočlenem $g(x)$, tím dostaneme zbytek podělení.
3. Zbytek podělení je přičten k rozšířenému mnohočlenu který byl získán v bodě 1. A tím je získán mnohočlen zabezpečené zprávy $c(x)$.

2.1.1 Kodér BCH kódu realizovaný děličkou g(x)

Postup zabezpečení výše popsaný je podle [13] založený na dělení vytvářecím mnohočlenem $g(x)$. Je realizován v děličce mod $g(x)$, která je tvořena kruhovým registrem. Kruhový registr je tvořen zpětnými vazbami a sčítačkami modulo2, které jsou vhodně umístěny podle vytvářecího mnohočlenu $g(x)$. Příklad zapojení posuvného registru je uveden na obr. 2.1, základní podoba převzata z [2, 5].



Obr. 2.1: Zapojení děličky mod $g(x)$

Posuvný kruhový registr obsahuje $(n-k)$ paměťových buněk $(r_0 \ldots r_{n-k-1})$. Počet paměťových buněk, který je potřeba je určen z řádu vytvářecího mnohočlenu $g(x)$.

Z $g(x)$ zjistíme, před které členy je potřeba vložit sčítačku mod2, jsou to členy před kterými je v $g(x)$ znaménko „+“. Pro nejvyšší řád $g(x)$ už není potřeba mít paměťovou buňku v posuvném registru a sčítačka, která mu náleží je umístěna před buňkou r_0.

Popis funkce kodéru znázorněného na obr. 2.1

Vstup do kodéru (děličky) je délky k bitů. Spínač $S2$ je v poloze 1 – všechny vstupní informační bity jsou přenášeny v nezměněné podobě i na výstup. Spínač $S1$ je po tuto dobu sepnut. Díky tomuto nastavení se vstupní bity dostávají i do jednotlivých buněk posuvného registru. Tento děj se opakuje od 1 do k cyklů.
V cyklech od $k + 1$ do n probíhá posouvání posuvného registru a všechna data jsou tak přenášena na výstup kodéru. V této době je spínač $S1$ rozepnut a spínač $S2$ v poloze 2. Posunování v registru je prováděno po dobu n cyklů. Po provedení celého postupu je na výstupu připraveno kódové slovo – zabezpečená zpráva $c(x)$. Následně je tento postup opakován pro další bitovou posloupnost.

Postup kódování a dekódování binárních BCH kódů je téměř schodný s postupem pro nebinární kódy, který je popsán v následujících kapitolách 4 – 5.5. S jediným rozdílem, že u binárních BCH kódů nemusíme provádět výpočet hodnoty chyb, jelikož je hodnota chyby vždy 1.

3 NEBINÁRNÍ BCH KÓDY

V této kapitole se budeme zabývat nebinárními BCH kódy. Nebinární BCH kódy se oproti binárním liší, že nebinární kódové slovo BCH kódu se skládá ze symbolů z $GF(2^m)$, kdežto binární obsahují v kódovém slově pouze binární bity (složené z jedniček a nul). Proto také veškeré aritmetické operace v nebinárních kódech už nejsou pouze jednoduchý XOR a AND, ale jsou prováděny nad $GF(2^m)$. Jinak jsou veškeré vlastnosti spjaté s binárními BCH kódy aplikovány v nebinárních. Veškeré informace v této kapitole jsou čerpány z [5, 6, 15].

Stejně jako u binárních BCH kódů, nebinární BCH kód opravující t chyb, délky

$$n = q^z - 1, \tag{3.1}$$

kde $q = 2^m$ a $z \geq 3$ obsahující $\alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$, kde α je primitivní element z $GF(q^z)$. Vytvářecí mnohočlen je pak konstruován pomocí nejmenšího společného násobku (LCM) následně:

$$g(X) = g_0 + g_1X + g_2X^2 + \cdots + g_{k-1}X^{k-1} = LCM[m_1(X), m_2(X), \cdots, m_{2t}(X)] \tag{3.2}$$

kde $g_i \in GF(2^m)$ a $m_i(X)$ je minimální mnohočlen z α^i. Minimální hammingova vzdálenost $d_{min} \geq 2t+1$. Důležité je mít na paměti, že u nebinárních kódů znamená, že kód který opravuje t chyb, znamená t symbolů chyb. Také je nutno nezapomenout rozlišovat dvě Galoisova tělesa, která jsou v nebinárních BCH kódech využívána, jsou to: $GF(2^m)$ a $GF(q^z)$, kde $q = 2^m$. Z prvního uvedeného GF pochází koeficienty vytvářecího polynomu a také symboly kódového slova, a z druhého má vytvářecí polynom své kořeny.

Pro příklad si uveďme návrh nebinárního BCH kódů daného rovnicí 3.1, pokud dosadíme za $m = 3$ a za $z = 3$ (pro nejmenší možný případ) dostaneme z rovnice 3.1 kód o délce $n = 2^{3^3} - 1 = 2^9 - 1 = 512 - 1 = 511$ symbolů. Jelikož výsledkem práce je, předvedení práce s Galoisovým tělesem při zakódování a dekódování, je pro tento účel vhodnější využít Reed-Solomonovy kódy, které mají podle [5, 6] vždy $z = 1$ a tím je jejich možná délka kratší. Reed-Solomonovy kódy budou představeny v pozdější kapitole.

4 KÓDOVÁNÍ NEBINÁRNÍCH BCH KÓDŮ

Kódování nebinárních BCH kódů je prováděno podobně jako u binárních kódů, jen jsou výpočty prováděny nad $GF(2^m)$. Prvně definujme používané mnohočleny:

$p(x)$ – mnohočlen nezabezpečené zprávy,

$g(x)$ – vytvářecí (generující) mnohočlen,

$m(x)$ – mnohočlen podílu,

$d(x)$ – mnohočlen zbytku po dělení,

$c(x)$ – mnohočlen zabezpečené zprávy (kódové slovo),

$r(x)$ – mnohočlen přijaté zprávy,

$e(x)$ – chybový mnohočlen.

Podle [1, 8] můžeme způsob zabezpečení matematicky popsat

$$\frac{p(x) \cdot x^{n-k}}{g(x)} = m(x) + \frac{d(x)}{g(x)} \tag{4.1}$$

Rovnici 4.1 můžeme přepsat na

$$p(x) \cdot x^{n-k} = m(x) \cdot g(x) + d(x) \tag{4.2}$$

Přepsáním pomocí algebry mod2 dostaneme

$$c(x) = p(x) \cdot x^{n-k} + d(x) = m(x) \cdot g(x) \tag{4.3}$$

Z rovnice 4.3 vidíme, že mnohočlen $c(x)$ je dělitelný $g(x)$ bezezbytku. Na této zkutečnosti je založena kontrola správnosti přijatého mnohočlenu $r(x)$ v procesu dekódování.

5 DEKÓDOVÁNÍ NEBINÁRNÍCH BCH KÓDŮ

Dekódování nebinárních BCH kódů je opět podobné jako u binárních kódů jen s tím rozdílem, že u nebinárních musíme provést jeden krok navíc a to: nalezení hodnoty chyby. U binárních kódů stačilo zjistit pouze polohu chyby, protože následná oprava byla už jednoduchá, jelikož stačilo nahradit nulu za jedničku a naopak. U nebinárních kódů u kterých využíváme symbolů které pocházejí z $GF(2^m)$ je potřeba zjistit po nalezení pozice chyby ještě hodnotu chybného symbolu.

Mějme kódové slovo $c(x)$, které bylo při přenosu zatíženo chybou. Na vstup přijímače tedy dostaneme pozměněné kódové slovo $r(x)$. Potom podle [8] můžeme napsat vztah:

$$r(x) = c(x) + e(x) \tag{5.1}$$

Pokud tedy přijatý mnohočlen $r(x)$, který obsahuje t nebo méně chyb, snažíme se najít chybový mnohočlen $e(x)$. Pokud v přijatém mnohočlenu zprávy $r(x)$ je více chyb, než t byla překročena zabezpečovací schopnost kódu a tím proces dekódování skončí neúspěšně. Proto se proces dekódování nebinárních BCH kódů skládá z následujících kroků[15]:

1. Vypočítání syndromů S_i pro $i = 1, 2, \ldots, 2t$
2. Nalezení chybového lokalizačního mnohočlenu $\Lambda(x)$
3. Nalezení kořenů chybového lokalizačního mnohočlenu
4. Výpočet hodnoty chyby
5. Opravení chyb

První krok je prováděn nad $GF(q^z)$ ostatní jsou nad $GF(2^m)$ [15].

5.1 Výpočet syndromů

Prvním krokem dekódování je výpočet syndromů, ten nám slouží k ověření zda při přenosu došlo k chybám. To můžeme zjistit tak, že postupně dosazujeme kořeny vytvářecího mnohočlenu $g(x)$ do mnohočlenu přijaté zprávy $r(x)$. Tím získáme syndromové rovnice S_j. Jelikož

$$g(\alpha) = g(\alpha^2) = \cdots = g(\alpha^{2t}) = 0, \tag{5.2}$$

z toho nám vyplývá že kódové slovo $c = (c_0, \ldots, c_{n-1})$ s polynomem $c(x) = c_0 + \cdots + c_{n-1}x^{n-1}$ má podle [6]

$$c(\alpha) = \cdots = c(\alpha^{2t}) = 0$$

pro přijatý polynom $r(x) = c(x) + e(x)$ máme

$$S_j = r(\alpha^j) = e(\alpha^j) = \sum_{k=0}^{n-1} e_k \alpha^{jk}, \qquad j = 1, 2, \ldots, 2t \tag{5.3}$$

Hodnoty $S_1, S_2, \ldots, S_{2t}$ jsou nazývány syndromy přijatých dat. Za předpokladu, že $r(x)$ obsahuje v chyb na pozicích $i_1, i_2, \cdots, i_v$ s odpovídající chybovou hodnotou v lokacích $e_{i_j} \neq 0$. Tak podle [6]

$$S_j = \sum_{l=1}^{v} e_{i_l} (\alpha^j)^{i_l} = \sum_{l=1}^{v} e_{i_l} (\alpha_l^i)^j. \tag{5.4}$$

Nechť

$$X_l = \alpha^{i_l}.$$

Pak můžeme psát dle [6]

$$S_j = \sum_{l=1}^{v} e_{i_l} X_l^j \qquad j = 1, 2, \ldots, 2t. \tag{5.5}$$

Jestliže známe X_l tak potom také známe pozici chyby. Například předpokládejme, že známe $X_l = \alpha^3$. To znamená, že z definice X_l je $i_1 = 3$; to znamená, že chyba se nachází v čísle r_3. Tímto X_l nazýváme podle [6] lokátor chyby. Dalším krokem je nalezení lokalizačního mnohočlenu.

5.2 Chybový lokalizační mnohočlen

Z rovnice 5.5 dostaneme sadu rovnic

$$\begin{aligned} S_1 = &\quad e_{i_1} X_1^j + e_{i_2} X_2^j + \cdots + e_{i_v} X_v^j \\ S_2 = &\; e_{i_1} X_1^{2j} + e_{i_2} X_2^{2j} + \cdots + e_{i_v} X_v^{2j} \\ &\vdots \\ S_2 t = &e_{i_1} X_1^{2tj} + e_{i_2} X_2^{2tj} + \cdots + e_{i_v} X_v^{2tj} \end{aligned} \tag{5.6}$$

Dostali jsem $2t$ rovnic ve v neznámých lokacích. Jelikož přímý výpočet těchto nelineárních rovnic je výpočetně náročný, je dán lokalizační mnohočlen, který je definován podle [6] takto

$$\Lambda(x) = \prod_{l=1}^{v}(1 - X_l x) = \Lambda_v x^v + \Lambda_{v-1} x^{v-1} + \cdots + \Lambda_1 x + \Lambda_0, \tag{5.7}$$

kde $\Lambda_0 = 1$. V dalších podkapitolách si uvedeme některé algoritmy pro nalezení lokalizačního mnohočlenu $\Lambda(x)$.

5.2.1 Peterson-Gorenstein-Zierler algoritmus

Peterson-Gorenstein-Zierlerova metoda (dále jen PGZ) je založena na inverzi matice syndromové [3]:

$$\Lambda = S^{-1} \cdot S. \tag{5.8}$$

Základem řešení je následující teorém. Vandermondova matice S složená ze syndromů je nesingulární a je možné ji invertovat, je-li její dimenze $v \times v$, pokud by byla dimenze větší než v (kde v označuje počet vzniklých chyb) je matice singulární a není možné ji invertovat.

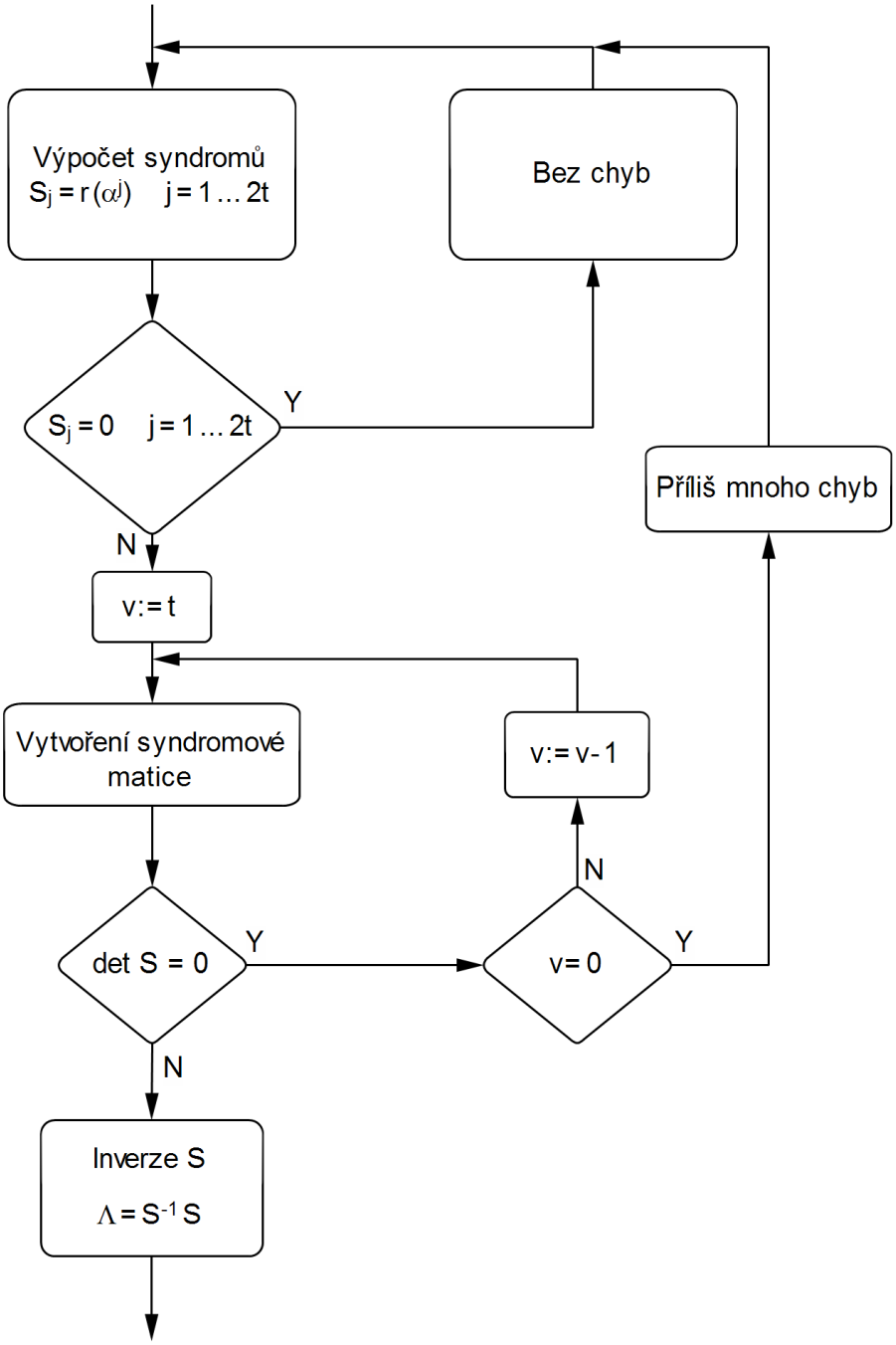
Musíme prvně zjistit v, abychom mohli být schopni invertovat S pro vyřešení rovnice 5.8. Na začátku je nastaveno $v = t$, jelikož t je maximum možných chyb, a vypočítán determinant S. Pokud je $\det(S) = 0$ je $v = t$ a můžeme pokračovat s inverzí matice. Jinak musíme v dekrementovat dokud se $\det(S) \neq 0$ a tím je nalezen počet vzniklých chyb v. Následně vypočítáme S^{-1} maticovou inverzí a dostaneme

$$\Lambda = S^{-1} \cdot S. \tag{5.9}$$

Následuje nalezení pozice chyb, které se provádí za pomoci Chienova vyhledávání, které je popsáno v kapitole 5.3. Po nalezení pozice je potřeba určit hodnotu chyb. Může být využito Forneyova algoritmu, nebo inverzí matice získanou ze syndromových rovnic 5.6, které můžeme přepsat do matice tvaru

$$M = P^{-1} \cdot S. \tag{5.10}$$

Kde P značí matici známých pozic chyb (získané pomocí Chienova vyhledávání), která musí být invertována, abychom dostali vektor M obsahující hodnoty chyb. Podobně jako u matice S, má P Vandermondovu strukturu tudíž je nesingulární a může být invertována, pokud došlo k přesně v chybám je dostačující podle [3] vyřešit jich pouze v neznámých hodnot chyb. Celý algoritmus PGZ metody je uveden ve vývojovém diagramu na obr. 5.1, převzato z [3].



Obr. 5.1: Vývojový diagram Peterson-Gorenstein-Zierler algoritmu

5.2.2 Berlekamp-Massey algoritmus

V této kapitole bude popsán pro stanovení chybového lokalizačního mnohočlenu Berlekamp-Massey algoritmus (dále jen BM algoritmus). BM algoritmus je založen na postupných iteracích, kde se v jednotlivých iteracích hledá co nejkratší posuvný registr (LFSR) k nalezení jednotlivých syndromů S [4, 14]. Po vypočtení prvního syndromu je počítán další podle vztahu

$$S_{i+1} = -\sum_{n=1}^{l^i} \Lambda_n^i S_{i+1-n}. \tag{5.11}$$

Následně je počítána odchylka (angl. discrepancy) $d^{(i)}$, která je dána podle [3]:

$$d^{(i)} = S_{i+1} + \sum_{n=1}^{l^i} \Lambda_n^i S_{i+1-n} = \sum_{n=0}^{l^i} \Lambda_n^i S_{i+1-n}. \tag{5.12}$$

Po zjištění, že $L_0 \neq 1$ pro iteraci $i = 1$ z rovnice 5.12 dostaneme

$$d^{(i)} = \Lambda_0^i S_2 + \Lambda_1^i S_1 = S_2 + S_1. \tag{5.13}$$

Pokud odchylka je $d^{(i)} = 0$ tak současný LFSR správně vytváří následující syndrom S_2. Následující úpravou je pak:

$$\begin{aligned} l^{(i+1)} &:= l^i \\ \Lambda^{(i+1)}(x) &:= \Lambda^i(x) \\ i &:= i + 1. \end{aligned}$$

Pokud $d^{(i)} \neq 0$ musí být LFSR upravován dokud není $d^{(i)} = 0$. Podle [3] je pak prodloužen o polynom

$$\Lambda^{i+1}(x) = \Lambda^i(x) - x^{i-m} \frac{d^{(i)}}{d^{(m)}} \Lambda^{(m)}(x), \tag{5.14}$$

kde m a d_m jsou z $m$té iterace, která vytvořila chybný syndrom S_m a x^{i-m} posouvá na požadovanou pozici LFSR.

Tyto kroky BM algoritmu mohou být formálně shrnuty Blahutovou interpretací [3] a jsou znázorněny na obr.5.2. Podle znázorněného algoritmu jsou jednotlivé fáze popsány níže [3]:

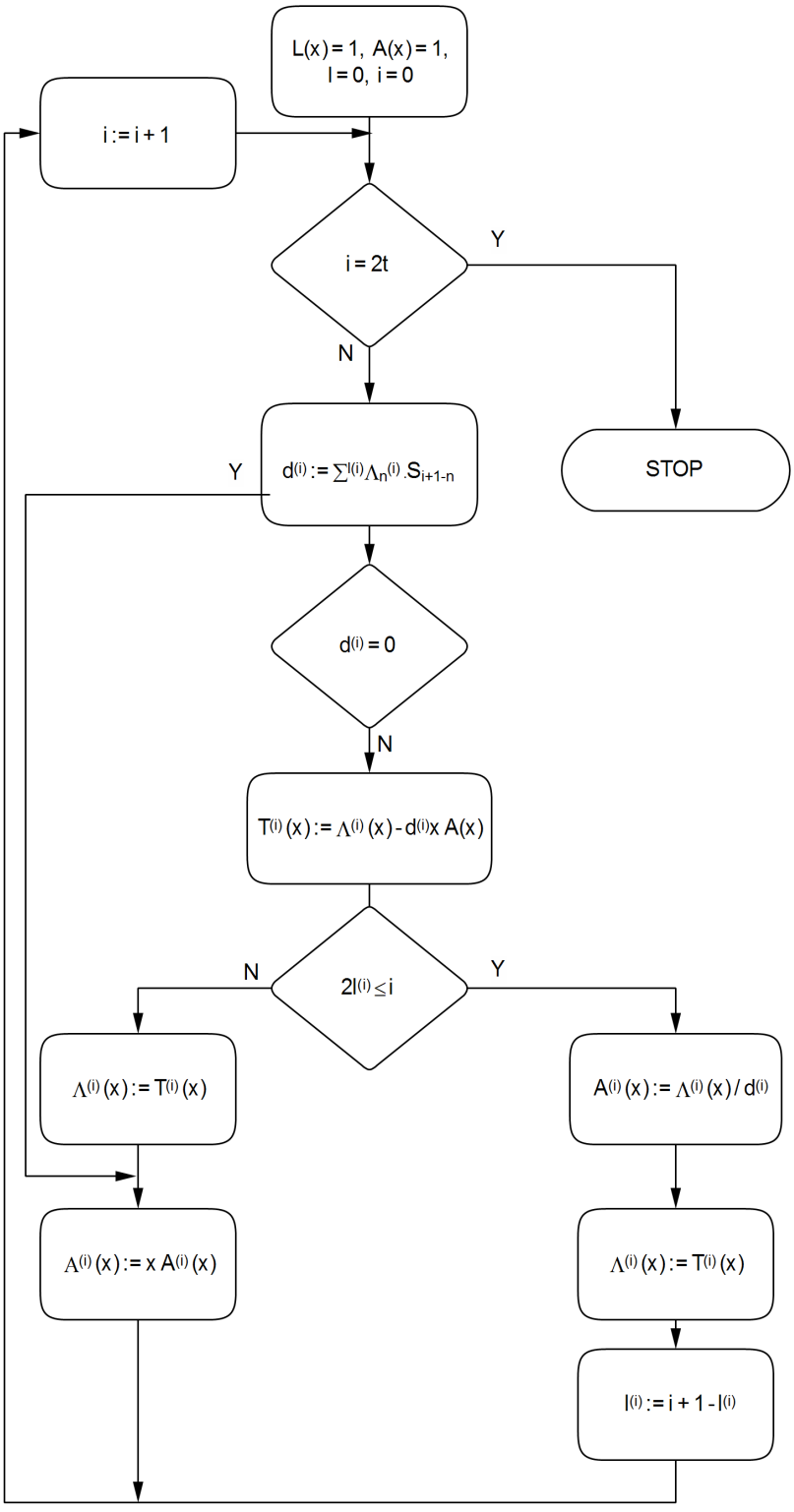
1. Základní nastavení

   index iterace: $i = 0$

   délka LFSR: $l^{(1)} = 0$

   chybový lokalizační mnohočlen: $\Lambda^{(1)}(x) = 1$

   pomocný mnohočlen: $A^{(1)}(x) = 1$



Obr. 5.2: Vývojový diagram Berlekamp-Massey algoritmus

2. Kontrola zda $i = 2t$, pokud ano konec iterace
3. Výpočet odchylky nebo chyby, která souvisí s vytvořením dalšího syndromu podle rovnice 5.12

$$d^{(i)} = \sum_{n=0}^{l^i} \Lambda_n^i S_{i+1-n}$$

4. Kontrola, zda je odchylka rovna nule $d^{(i)} = 0$, tak aktuální LFSR o délce $l^{(i)}$ a chybového mnohočlenu $\Lambda^{(i)}(x)$ vytváří další syndrom S_{i+1}. Tak přejděte ke kroku 5 jinak ke kroku 6.
5. Posunutí o jednu pozici pomocného mnohočlenu LFSR

$$A^i(x) = xA^{(i)}(x) \tag{5.15}$$

a dále pokračujte krokem 12.

6. Jelikož $d^{(i)}$, tak opravíme dočasný chybový mnohočlen $T^{(i)}(x)$, přidáním posunutého pomocného chybového mnohočlenu $A^{(i)}(x)$, k současnému chybovému mnohočlenu $\Lambda^{(i)}(x)$, podle vztahu:

$$T^i(x) = \Lambda^{(i)}(x) - d^{(i)}xA^{(i)}(x). \tag{5.16}$$

7. Kontrola zda je potřeba LFSR zvětšit. Pokud je $2l^{(i)} \leq i$ tak pokračujte krokem 8, jinak až krokem 9.
8. Aktualizace chybového mnohočlenu podle vztahu:

$$\Lambda^i(x) := T^{(i)}(x) \tag{5.17}$$

a proveďte opět krok 5.

9. Normalizace nejaktuálnějšího chybového mnohočlenu $\Lambda(x)$ jeho dělením s $d^{(i)} \neq 0$ a uložení výsledku do pomocného LFSR $A^{(i)}(x)$, následně:

$$A^i(x) := \frac{\Lambda^{(i)}(x)}{d^{(i)}} \tag{5.18}$$

a pokračujte krokem 10.

10. Nyní můžeme přepsat $\Lambda^{(i)}(x)$, jelikož byl v předchozím kroku uložen do $A^{(i)}(x)$, takže $\Lambda^{(i)}(x)$ může být aktualizován podle:

$$\Lambda^i(x) := T^{(i)}(x) \tag{5.19}$$

a pokračovat krokem 11.

11. Nyní je podle kroku 7 potřeba prodloužit LFSR, podle následujícího vztahu:

$$l^{i+1} = i + 1 - l^{(x)} \tag{5.20}$$

    a následně pokračujme krokem 12.

12. Inkrementace iteračního indexu i a návrat ke kroku 2.

Po provedení všech výše uvedených kroků je nalezen chybový lokalizační mnohočlen. Dalším krokem je využití Chienova vyhledávání k nalezení jeho kořenů, které je popsáno v kapitole 5.3

Tento algoritmus je popsán ve spoustě literatur zabývající kódováním a oprav chyb, pro jeho podrobnější studium doporučuji [3, 5, 6, 11].

5.2.3 Euklidův algoritmus

Pro nalezení chybového lokalizačního mnohočlenu může být využit Euklidův algoritmus [6, 15, 14] Základem je klíčová rovnice – chybového mnohočlenu $e(x)$, která je podle [6, 15] definována:

$$e(x) = S(x)\Lambda(x) \quad (mod x^{2t}). \tag{5.21}$$

Známe pouze $S(x)$ a t, snažíme se určit chybový lokalizační polynom $\Lambda(x)$ a mnohočlen velikosti chyb $e(x)$. Vyřešit tuto rovnici se nám může zdát jako nemožné. Avšak rovnice 5.21 podle [6] znamená že:

$$\Theta(x)(x^{2t}) + \Lambda(x)S(x) = e(x) \tag{5.22}$$

pro libovolný polynom Θ. Podle [6] rozšířený Euklidův algoritmus pro dvojici prvků (a, b) vrací dvojici (s, t) takovou, že:

$$as + bt = c,$$

kde c je nejmenším společným dělitelem z a a b. V našem případě po aplikaci rozšířeného Euklidova algoritmu dostaneme množinu polynomů $\Theta^{[k]}(x)$, $\Lambda^{[k]}(x)$ a $e^{[k]}(x)$ splňující

$$\Theta^{[k]}(x)(x^{2t}) + \Lambda^{[k]}(x)S(x) = e^{[k]}(x).$$

Jakmile dostaneme $e(x)$

Kroky pro dekódování za použití Euklidova algoritmu, se pak mohou shrnout podle [6, 7] do:

1. Výpočet syndromů a syndromového polynomu $S(x) = S_1 + S_2x + \cdots + S_{2t}x^{2t-1}$.
2. Pokud $S(X) = 0$ tak je přijatý vektor považován za kódový vektor.
3. Pokud $S(X) \neq 0$ tak je zpočátku algoritmus nastaven na

$$\begin{aligned} r_{-1}(X) &= X^{n-k} \\ r_0(X) &= S(X) \\ t_{-1}(X) &= 0 \\ t_0(X) &= 1 \\ i &= -1 \end{aligned}$$

4. Parametry rekurze jsou určeny takto

$$\begin{aligned} r_i(X) &= r_{i-2}(X) - q(X)r_{i-1}(X) \\ t_i(X) &= t_{i-2}(X) - q(X)t_{i-1}(X) \end{aligned}$$

   rekurze probíhá dokud je stupeň $(r_i(X)) \geq t$
5. Pokud je stupeň $(r_i(X)) < t$ tak rekurze je ukončena a je možné určit:

$$\begin{aligned} e(x) &= r_i(X) \\ \Lambda(X) &= t_i(X) \end{aligned}$$

Po provedení kroků výše uvedených je znám chybový lokalizační polynom a je pokračováno v dalších krocích dekódování popsané v kapitolách 5.3 a 5.4. Pro detailnější studium Euklidova algoritmu doporučuji [5, 6, 7].

5.3 Nalezení kořenů chybového lokalizačního mnohočlenu

Dalším krokem po nalezení lokalizačního mnohočlenu, je hledání kořenů chybového polynomu, které se obvykle provádí pomocí Chienova vyhledávání [6]. Princip je založen na postupném dosazování všech prvků z Galoisova tělesa $\mathrm{GF}(q^m)$, (které bylo použito pro tvorbu vytvářecího polynomu), do rovnice lokalizačního mnohočlenu (5.7).

Například předpokládejme, že $v = 2$ lokalizační mnohočlen je pak podle vztahu 5.7 tvaru:

$$\Lambda(x) = \Lambda_0 + \Lambda_1 x + \Lambda_2 x^2 = 1 + \Lambda_1 x + \Lambda_2 x^2. \tag{5.23}$$

Pro každý nenulový prvek Galoisova tělesa dosadíme za $x = 1, x = \alpha, x = \alpha^2, \ldots, x = \alpha^{q^m-2}$. Dostaneme:

$$\begin{aligned}
\Lambda(1) &= 1 + \Lambda_1(1) + \Lambda_2(1)^2 \\
\Lambda(\alpha) &= 1 + \Lambda_1(\alpha) + \Lambda_2(\alpha)^2 \\
&\vdots \\
\Lambda(\alpha^{q^m-2}) &= 1 + \Lambda_1(\alpha^{q^m-2}) + \Lambda_2(\alpha^{q^m-2})^2
\end{aligned}$$

Prvky Galoisova tělesa pro které vyjde $\Lambda = 0$ označují svým exponentem pozici chyby.

5.4 Výpočet hodnoty chyby

Pro výpočet hodnoty chyby je možné vyjít ze vztahu 5.5. Pokud známe lokalizační mnohočlen a jeho kořeny, je možné hodnotu přímo vypočítat ze soustavy rovnic

$$\begin{bmatrix} X_1 & X_2 & X_3 & \cdots & X_v \\ X_1^2 & X_2^2 & X_3^2 & \cdots & X_v^2 \\ \vdots & & & & \\ X_1^{2t} & X_2^{2t} & X_3^{2t} & \cdots & X_v^{2t} \end{bmatrix} \begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_v} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{2t} \end{bmatrix}. \tag{5.24}$$

Avšak existuje metoda, která je na výpočetně jednodušší. Matice 5.24 je v podstatě Vandermondova matice. Existují algoritmy pro rychlé řešení Valdermondových systémů. Jeden z nich je tzv. Forneyův algoritmus [6]. Před uvedením jeho formulace je nejprve potřeba několik definic. Mnohočlen syndromů je:

$$S(x) = S_1 + S_2 x + S_3 x^2 + \cdots + S_{2t} x^{2t-1} = \sum_{j=0}^{2t-1} S_{j+1} x^j. \tag{5.25}$$

Chybový mnohočlen $e(x)$ je podle [6, 15] definován

$$E(x) = S(x)\Lambda(x) \quad (mod x^{2t}). \tag{5.26}$$

Tato rovnice je nazývána klíčová rovnice (anglicky „key equation“). Efektem výpočtu modulo x^{2t} je zrušení všech členů polynomu o stupni $2t$ nebo vyšším.

A nakonec podle [6, 15] je Forneyův algoritmus definován:

$$e_{ik} = -\frac{e(X_k^{-1})}{\Lambda'(X_k^{-1})}, \tag{5.27}$$

kde Λ' je derivací Λ (lokalizačního mnohočlenu). Podle [7] můžeme vztah úpravou přepsat na

$$M_l = \frac{E(P_l^{-1})}{\prod_j \neq l(1 - P_j P_l^{-1})}, \tag{5.28}$$

kde E je mnohočlen pro odhad chyby, P_l je pozice chyby a M_l je hodnota chyby.

5.5 Opravení chyb

Na závěr dekódovacího procesu, kdy už máme určenu lokaci i hodnotu chyb v přenesené zprávě je vytvořen chybový mnohočlen $e(x)$. Je už pouze potřeba přičíst tento mnohočlen k mnohočlenu přijaté zprávy $r(x)$, dostaneme tak vztah:

$$c(x) = r(x) + e(x) \tag{5.29}$$

a touto úpravou je dekódovaní ukončeno.

6 REED-SOLOMONOVY KÓDY

Nejdůležitější známou podskupinou nebinárních BCH kódů je třída Reed-Solomonových kódů, které byly navrženy v roce 1960 Reedem a Solomonem, nezávisle na BCH kódech. Vztah mezi BCH kódy a Reed-Solomonovy kódy, byla později dokázána Goreinsteinem a Zielerem v roce 1961 [5]. Tyto kódy jsou velmi efektivní pro opravu náhodných symbolových chyb a jejich využití je v komunikacích a ukládání dat široké.

Reed-Solomonovy (RS) kódy jsou podle [5] definovány

Bloková délka:

$$n = q - 1, \tag{6.1}$$

Počet paritních symbolů:

$$n - k = 2t, \tag{6.2}$$

Minimální vzdálenost:

$$d_{min} = 2t + 1, \tag{6.3}$$

Korekční schopnost kódu:

$$t = \frac{n-k}{2}, \tag{6.4}$$

Vytvářecí mnohočlen je definován:

$$g(x) = (x - \alpha)(x - \alpha^2) \cdots (x - \alpha^{2t}). \tag{6.5}$$

Kde α jsou prvky $GF(2^m)$

Pro zakódování a dekódování lze využít teorie, která byla popsána v kapitolách 4 a 5.

6.1 Příklad zakódování

Pro ukázku zakódování vezmeme kód $RS(7, 3)$ určený $GF(2^3)$ nad polynomem $G(x)_{GF} = 1 + x + x^3$, jednotlivé prvky jsou uvedeny v tabulce 6.1.

Nezabezpečenou zprávu např. $p(x) = \alpha^2 + \alpha^3 x + x^2$.

Nejprve je potřeba určit vytvářecí mnohočlen $g(x)$. Podle vztahu 6.4 víme, že je $t = 2$ a pomocí vztahu 6.5, určíme vytvářecí mnohočlen $g(x)$ následovně:

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) = \alpha^3 + \alpha x + x^2 + \alpha^3 x^3 + x^4.$$

Tab. 6.1: Galoisovo těleso generované $G(X)_{GF} = 1 + X + X^3$

Exponenciální tvar	Polynomiální tvar	Binární tvar
0	0	0 0 0
1	1	1 0 0
α	α	0 1 0
α^2	α^2	0 0 1
α^3	$1 + \alpha$	1 1 0
α^4	$\alpha + \alpha^2$	0 1 1
α^5	$1 + \alpha + \alpha^2$	1 1 1
α^6	$1 + \alpha^2$	1 0 1
$\alpha^7 = \alpha^0$	$\alpha^0 = 1$	1 0 0

Podle vztahu 4.3 zabezpečenou zprávu $c(x)$ dostaneme, vynásobením $p(x)$ stupněm $g(x)$, který je dán x^{n-k} a následném sečtení se zbytkem po dělení $d(x)$. Nejprve určíme

$$p(x) \cdot x^{n-k} = (\alpha^2 + \alpha^3 x + x^2) \cdot x^4 = \alpha^2 x^4 + \alpha^3 x^5 + x^6$$
$$d(x) = \frac{p(x) \cdot x^{n-k}}{g(x)} = \alpha^2 + x + \alpha^4 x^2 + \alpha^4 x^3$$
$$c(x) = \alpha^2 + x + \alpha^4 x^2 + \alpha^4 x^3 + \alpha^2 x^4 + \alpha^3 x^5 + x^6$$

6.2 Příklad dekódování

V této kapitole si ukážeme na příkladu jednotlivé kroky dekódování. Pro dekódování RS kódů využijeme postup popsaný v předchozí kapitolách definované obecně pro nebinární BCH kódy.

Použijeme kódu na kterém jsme si předvedli zakódování v kapitole 6.1. $RS(7, 3)$ určený $GF(2^3)$ nad polynomem $G(x)_{GF} = 1+x+x^3$, jednotlivé prvky jsou uvedeny v tabulce 6.1. Výsledné kódové slovo bylo $c(x) = \alpha^2+x+\alpha^4x^2+\alpha^4x^3+\alpha^2x^4+\alpha^3x^5+x^6$. Při přenosu, může dojít ke změně přenášeného slova, to si můžeme lehce simulovat úpravou uvedeného kódového slova a tím dostaneme přijaté slovo $r(x)$. Víme, že kód dokáže opravit až 2 chyby, proto upravíme kódové slovo následovně $r(x) = \alpha^4 + x + \alpha^4x^2 + \alpha^4x^3 + \alpha^2x^4 + \alpha^5x^5 + x^6$. Prvním krokem dekódování je výpočet syndromů.

6.2.1 Výpočet syndromů

Pro výpočet syndromů vyjdeme z rovnice 5.3, kdy do přijaté zprávy $r(x)$, dosadíme postupně jednotlivé kořeny vytvářecího mnohočlenu $g(x)$ a tím dostaneme $2t$ syndromů.

$$S_1 = r(\alpha) = \alpha^4 + \alpha + \alpha^4(\alpha)^2 + \alpha^4(\alpha)^3 + \alpha^2(\alpha)^4 + \alpha^5(\alpha)^5 + (\alpha)^6 = \alpha^5$$
$$S_2 = r(\alpha^2) = \alpha^4 + \alpha^2 + \alpha^4(\alpha^2)^2 + \alpha^4(\alpha^2)^3 + \alpha^2(\alpha^2)^4 + \alpha^5(\alpha^2)^5 + (\alpha^2)^6 = \alpha^4$$
$$S_3 = r(\alpha^3) = \alpha^4 + \alpha^3 + \alpha^4(\alpha^3)^2 + \alpha^4(\alpha^3)^3 + \alpha^2(\alpha^3)^4 + \alpha^5(\alpha^3)^5 + (\alpha^3)^6 = \alpha^6$$
$$S_4 = r(\alpha^4) = \alpha^4 + \alpha^4 + \alpha^4(\alpha^4)^2 + \alpha^4(\alpha^4)^3 + \alpha^2(\alpha^4)^4 + \alpha^5(\alpha^4)^5 + (\alpha^4)^6 = \alpha^2$$

6.2.2 Použití PGZ algoritmu

Dalším pokračováním po vypočítání syndromů je nalezení lokalizačního mnohočlenu, zde si ukážeme využití PGZ algoritmu popsaného v kapitole 5.2.1. Pro přehlednost přepíšeme výsledky syndromů:

$$S_1 = \alpha^5,\ S_2 = \alpha^4,\ S_3 = \alpha^6,\ S_4 = \alpha^2.$$

Jelikož není znám přesný počet chyb, ke kterým došlo při přenosu je nejprve nastaveno $v = t$, protože víme, že kód je schopen opravit $t = 2$ chyb. Nejprve vypočítáme determinant S:

$$\det S = \det \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = (S_1 S_3 - S_2^2) = (\alpha^5\alpha^6 - \alpha^{4^2}) = \alpha^4 - \alpha^2 \neq 0.$$

Jelikož se determinant nerovná nule, tak jsme zjistili, že došlo ke $t = 2$ chybám a nyní můžeme vypočítat S^{-1}:

$$S^{-1} = \left[\begin{array}{cc|cc} S_1 & S_2 & \alpha^0 & 0 \\ S_2 & S_3 & 0 & \alpha^0 \end{array}\right] = \left[\begin{array}{cc|cc} \alpha^5 & \alpha^4 & \alpha^0 & 0 \\ \alpha^4 & \alpha^6 & 0 & \alpha^0 \end{array}\right]$$

Teď vynásobíme první řádek α^6 a přičteme k druhému

$$\left[\begin{array}{cc|cc} \alpha^5 & \alpha^4 & \alpha^0 & 0 \\ 0 & \alpha^4 & \alpha^6 & \alpha^0 \end{array}\right]$$

Přičteme druhý řádek k prvnímu

$$\left[\begin{array}{cc|cc} \alpha^5 & 0 & \alpha^2 & \alpha^0 \\ 0 & \alpha^4 & \alpha^6 & \alpha^0 \end{array}\right]$$

První řadek vynásobíme α^2 a druhý α^3

$$\left[\begin{array}{cc|cc} \alpha^0 & 0 & \alpha^4 & \alpha^2 \\ 0 & \alpha^0 & \alpha^2 & \alpha^3 \end{array}\right]$$

Podle vztahu 5.9 dostaneme

$$\Lambda = S^{-1}S = \begin{bmatrix}\Lambda_2 \\ \Lambda_1\end{bmatrix} = \begin{bmatrix}\alpha^4 & \alpha^2 \\ \alpha^2 & \alpha^3\end{bmatrix} \cdot \begin{bmatrix}\alpha^6 \\ \alpha^2\end{bmatrix} = \begin{bmatrix}\alpha^6 \\ \alpha^6\end{bmatrix}$$

a výsledný lokalizační mnohočlen je pak:

$$\Lambda(x) = \alpha^6 x^2 + \alpha^6 x + 1$$

Pomocí Chienova vyhledávání nalezneme pozice chyb, postupným dosazováním prvků $GF(2^3)$, ty jsou uvedeny v tab.6.1.

$$\begin{aligned}
\Lambda(\alpha^0) &= \alpha^6(\alpha^1)^2 + \alpha^6(\alpha^1) + 1 = 1 \\
\Lambda(\alpha^1) &= \alpha^6(\alpha^1)^2 + \alpha^6(\alpha^1) + 1 = \alpha^6 \\
\Lambda(\alpha^2) &= \alpha^6(\alpha^2)^2 + \alpha^6(\alpha^2) + 1 = 0 \\
\Lambda(\alpha^3) &= \alpha^6(\alpha^3)^2 + \alpha^6(\alpha^3) + 1 = \alpha \\
\Lambda(\alpha^4) &= \alpha^6(\alpha^4)^2 + \alpha^6(\alpha^4) + 1 = \alpha^3 \\
\Lambda(\alpha^5) &= \alpha^6(\alpha^5)^2 + \alpha^6(\alpha^5) + 1 = \alpha \\
\Lambda(\alpha^6) &= \alpha^6(\alpha^6)^2 + \alpha^6(\alpha^6) + 1 = 0
\end{aligned}$$

Λ, které se rovnají 0 je potřeba invertovat a z nich dostaneme pozici chyb P_i:

$$\begin{aligned}
P_1 &= (\alpha^2)^{-1} = \frac{\alpha^0}{\alpha^2} = \alpha^5 \\
P_2 &= (\alpha^6)^{-1} = \frac{\alpha^0}{\alpha^6} = \alpha^1
\end{aligned}$$

Nakonec je potřeba určit hodnotu chyb. Pro náš příklad dostaneme dvě rovnice, které bude potřeba vyřešit, jsou to:

$$\begin{aligned}
S_1 &= M_1 + P_1 + M_2 + P_2 \\
S_2 &= M_1 + P_1^2 + M_2 + P_2^2
\end{aligned}$$

$$\begin{aligned}
\alpha^5 &= M_1 + \alpha^5 + M_2 + \alpha^1 \\
\alpha^4 &= M_1 + \alpha^{5\,2} + M_2 + \alpha^{1\,2}
\end{aligned}$$

Vyjádřením z první rovnice M_1 a následným dosazením do druhé rovnice získáme M_2.

$$M_1 = \frac{\alpha^5 + M_2 + \alpha}{\alpha^5} = (\alpha^5 + M_2\alpha)\alpha^2 = \alpha^0 + M_2\alpha^3$$
$$\alpha^4 = (\alpha^0 + M_2\alpha^3)\alpha^3 + M_2 + \alpha^2 = \alpha^3 M_2\alpha^6 + M_2\alpha^2$$

$$\alpha^4 - \alpha^3 = M_2(\alpha^6 + \alpha^2)$$
$$M_2 = \frac{\alpha^4 + \alpha^3}{\alpha^6 + \alpha^2} = a^6$$
$$M_1 = \alpha^0 + \alpha^2 = \alpha^6$$

Teď už máme vypočítanou pozici chyb i jejich hodnotu a proto můžeme napsat chybový mnohočlen $e(x) = \alpha^6 x + \alpha^6 x^5$, přičtením tohoto chybového mnohočlenu k mnohočlenu přijaté zprávy dostaneme zprávu odeslanou $c(x) = r(x) + e(x)$.

6.2.3 Použití BM algoritmu

Dalším možným nalezení lokalizačního mnohočlenu je využití BM algoritmu, který byl popsán v kapitole 5.2.2. Opět pro stejný příklad pro ověření výsledku. Pro přehlednost přepíšeme výsledky syndromů:

$$S_1 = \alpha^5,\ S_2 = \alpha^4,\ S_3 = \alpha^6,\ S_4 = \alpha^2.$$

Dále pokračujeme podle postupu:

1. Základní nastavení
   $i = 0,\ l = 0,\ \Lambda^1(x) = 1,\ A^{(1)}(x) = 1$
2. Kontrola zda bude iterace ukončena $i = 2t$: $0 \neq 4$, proto pokračuj krokem 3
3. Výpočet odchylky,

$$d^{(i)} = \sum_{n=0}^{l^i} \Lambda_n^i S_{i+1-n} = \Lambda_0 S_1 = 1 \cdot \alpha^5 = \alpha^5$$

4. Kontrola, zda je odchylka rovna nule $d = \alpha^5 \neq 0$, přejděte kroku 6.
5.
6. opravíme dočasný chybový mnohočlen $T(x)$

$$T(x) = \Lambda(x) - dxA(x) = 1 + \alpha^5 x$$

7. Kontrola zda je potřeba LFSR zvětšit $2l \leq i$: $0 \leq 0$pokračujeme krokem 9.
8.
9. Normalizace nejaktuálnějšího chybového mnohočlenu $\Lambda(x)$

$$A(x) := \frac{\Lambda(x)}{d} = \frac{1}{\alpha^5} = \alpha^2$$

10. Aktualizace $\Lambda(x)$,

$$\Lambda(x) := T(x) = 1 + \alpha^5 x$$

dále ke pokračujeme krokem 11.

11. Nyní je podle kroku 7 potřeba prodloužit LFSR, $l = i + 1 - l = 0 + 1 - 0 = 1$ a následně pokračujme krokem 12.
12. Zde je ukončena první iterace a je inkrementován její index $i := i + 1 = 1$ a návrat ke kroku 2.

Druhá iterace

1. Aktuální stav
   $i = 1$, $l = 1$, $\Lambda(x) = 1 + \alpha^5 x$, $A(x) = \alpha^2$
2. Kontrola zda bude iterace ukončena $i = 2t$: $1 \neq 4$, proto pokračuj krokem 3
3. Výpočet odchylky,

$$d^{(i)} = \sum_{n=0}^{l^i} \Lambda_n^i S_{i+1-n} = \Lambda_0 S_2 + \Lambda_1 S_1 = 1 \cdot \alpha^4(-\alpha^5)\alpha^5 = \alpha^6$$

4. $d = \alpha^6 \neq 0$, přejděte kroku 6.
5.
6. opravíme dočasný chybový mnohočlen $T(x)$

$$T(x) = \Lambda(x) - dxA(x) = 1 + \alpha^5 x - \alpha^6 x\alpha^2 = 1 + x(\alpha^5 + \alpha) = 1 + \alpha^6 x$$

7. Kontrola zda je potřeba LFSR zvětšit $2l \leq i$: $2 \leq 1$ neplatí a proto pokračujeme krokem 8.
8. $L(x) = T(x) = 1 + \alpha^6 x$, pokračujeme krokem 5
   $A(x) = x \cdot A(x) = \alpha^2 x$
   pokračujeme krokem 12. Zde je ukončena druhá iterace a je inkrementován její index $i := i + 1 = 2$a návrat ke kroku 2.

Dále bychom postupovali stejným postupem, jako jsme si ukázali na prvních dvou iteracích a dostali bychom v třetí iteraci

1. $i = 2$, $l = 1$, $\Lambda(x) = 1 + \alpha^6 x$, $A^{(1)}(x) = \alpha^2 x$
2. iterace pokračuje
3. Výpočet odchylky, $d = \Lambda_0 S_3 + \Lambda_1 S_2 = \alpha^4$
4. Kontrola, zda je odchylka rovna nule $d = \alpha^4 \neq 0$, přejděte kroku 6.
5.
6. $T(x) = 1 + \alpha^6 x$
7. $2 \leq 2$: platí, pokračujeme krokem 9.
8.
9. $A(x) := \alpha^3 + \alpha^2 x$
10. Aktualizace $\Lambda(x) := 1 + \alpha^6 x + \alpha^6 x^2$,
11. prodloužení LFSR, $l = i + 1 - l = 2 + 1 - 1 = 2$
12. $i := i + 1 = 3$a návrat ke kroku 2.

Poslední iterace je obdobná:

1. $i = 3$, $l = 2$, $\Lambda(x) = 1 + \alpha^6 x + \alpha^6 x^2$, $A(x) = \alpha^3 + \alpha^2 x$
2. iterace pokračuje
3. Výpočet odchylky, $d = \Lambda_0 S_4 + \Lambda_1 S_3 + \Lambda_2 S_2 = \alpha^2 + \alpha^5 + \alpha^3 = 0$
4. Kontrola, zda je odchylka rovna nule $d = 0$, přejdeme ke kroku 5.
5. $A(x) = x \cdot A(x) = \alpha^3 x + \alpha^2 x^2$

   Pokračujeme krokem 12 $i := i + 1 = 4$ a nakonec krokem 2 zjistíme ukončení výpočtu.

Tímto jsme došli ke konci BM algoritmu a nalezli jsme lokalizační mnohočlen $\Lambda(x) = 1 + \alpha^6 x + \alpha^6 x^2$.

Tak jako v kapitole 5.2.1 Chien vyhledáváním jsme našli prvky GF, pro které $\Lambda = 0$, je potřeba invertovat a z nich dostaneme pozici chyb P_i:

$$P_1 = (\alpha^2)^{-1} = \frac{\alpha^0}{\alpha^2} = \alpha^5$$
$$P_2 = (\alpha^6)^{-1} = \frac{\alpha^0}{\alpha^6} = \alpha^1$$

Nyní pro výpočet hodnot chyb, které vznikli na námi objevených pozicích, využijeme Forneyova algoritmu popsaného v kapitole 5.4.

Dosadíme do rovnice 5.26

$$\begin{aligned} E(x) &= S(x)\Lambda(x) \quad (mod x^{2t}) \\ E(x) &= (\alpha^5 x + \alpha^4 x^2 + \alpha^6 x^3 + \alpha^2 x^4) \cdot (1 + \alpha^6 + \alpha^6 x^2) \\ &= \alpha^5 x + \alpha^6 x^5 + \alpha x^6 \quad (mod x^4) \\ E(x) &= \alpha^5 x \end{aligned}$$

Dále podle rovnice 5.28, vypočítáme hodnotu chyby

$$M_1 = \frac{E(P_1^{-1})}{1 - P_2 P_1^{-1}} = \frac{\alpha^5 + \alpha^2}{1 - \alpha\alpha^2} = \frac{\alpha^0}{\alpha} = \alpha^6$$
$$M_2 = \frac{E(P_2^{-1})}{1 - P_1 P_2^{-1}} = \frac{\alpha^5 + \alpha^6}{1 - \alpha^5\alpha^6} = \frac{\alpha^4}{\alpha^5} = \alpha^6$$

Výsledný chybový mnohočlen je pak $e(x) = \alpha^6 x \alpha^6 x^5$, přičtením tohoto chybového mnohočlenu k mnohočlenu přijaté zprávy dostaneme zprávu odeslanou $c(x) = r(x) + e(x)$.

6.2.4 Použití Euklidova algoritmu

Další možností jak nalézt lokalizační mnohočlen je využitím Euklidova algoritmu, který byl popsán v kapitole 5.2.3. Opět použijeme stejný příklad pro ověření výsledku. Pro přehlednost přepíšeme výsledky syndromů:

$$S_1 = \alpha^5,\ S_2 = \alpha^4,\ S_3 = \alpha^6,\ S_4 = \alpha^2,$$

následně je přepíšeme do syndromového polynomu

$$S(x) = \alpha^5 + \alpha^4 x + \alpha^6 x^2 + \alpha^2 x^3.$$

Jelikož se $S(x) \neq 0$ pokračujeme v dekódování. Jednotlivé kroky iterací jsou znázorněny v tabulce 6.2. Pokud je stupeň ve sloupci $r_i(x)$ menší než $t_i(x)$ tak je algoritmus ukončen a je možné určit:

$$e(x) = \alpha^2$$
$$\Lambda(x) = \alpha^3 x^2 + \alpha^3 x + \alpha^4$$

Následně je potřeba upravit do normovaného polynomu, vynásobením prvkem z použitého GF. Pro náš příklad je potřeba vynásobit prvkem α^4, dostaneme:

$$e(x) = \alpha^6$$
$$\Lambda(x) = x^2 + x + \alpha$$

Tab. 6.2: Tabulka iterací Euklidova algoritmu

i	$r_i(x) = r_{i-2}(x) - q_i(x)r_{i-1}(x)$	q_i	$t_i(x) = t_{i-2}(x) - q(x)t_{i-1}(x)$
-1	x^4		0
0	$\alpha^2x^3 + \alpha^6x^2 + \alpha^4x + \alpha^5$		1
1	$\alpha^2x^3 + \alpha^3x^2 + \alpha^4$	$\alpha^5x + \alpha^6$	$\alpha^5x + \alpha^6$
2	$\alpha^4x^2 + \alpha^4x + 1$	1	$\alpha^5x + \alpha^6 + 1$
3	α^2	$\alpha^5x + \alpha$	$\alpha^3x^2 + \alpha^3x + \alpha^4$

Tak jako v kapitole 5.2.1 Chien vyhledáváním jsme našli prvky GF, pro které $\Lambda = 0$,(α^2 a α^6) je potřeba invertovat a z nich dostaneme pozici chyb P_i:

$$P_1 = (\alpha^2)^{-1} = \frac{\alpha^0}{\alpha^2} = \alpha^5$$
$$P_2 = (\alpha^6)^{-1} = \frac{\alpha^0}{\alpha^6} = \alpha^1$$

Nyní pro výpočet hodnot chyb, které vznikli na námi objevených pozicích, musíme ještě určit derivaci lokalizačního polynomu ta je pro náě příklad: $\Lambda' = 1$ a dosazením určíme hodnoty:

$$M_1 = \frac{e(P_1^{-1})}{\Lambda'(P_1^{-1})} = \frac{\alpha^6}{1} = \alpha^6$$
$$M_2 = \frac{e(P_2^{-1})}{\Lambda'(P_2^{-1})} = \frac{\alpha^6}{1} = \alpha^6$$

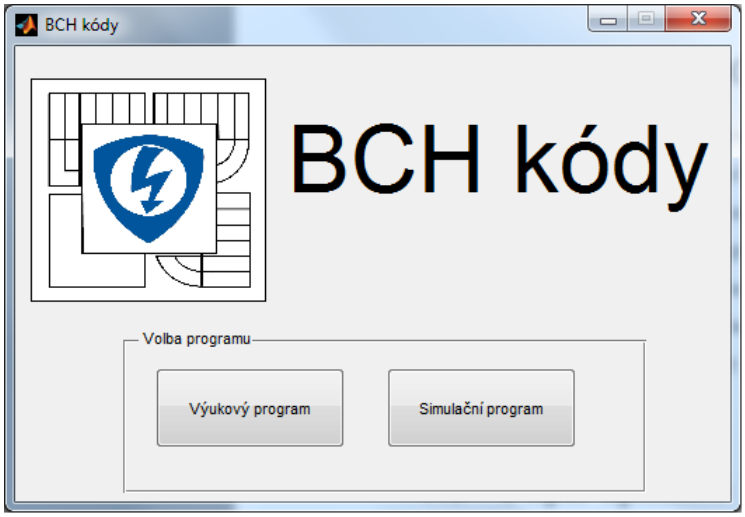
Výsledný chybový mnohočlen je pak $e(x) = \alpha^6x + \alpha^6x^5$, přičtením tohoto chybového mnohočlenu k mnohočlenu přijaté zprávy dostaneme zprávu odeslanou $c(x) = r(x) + e(x)$. Pokud porovnáme nalezené lokalizační mnohočleny BM a Euklidova algoritmu vidíme, že se nepatrně liší. Jak si je možné všimnou tak pokud v našem příkladu lokalizační mnohočlen nalezený Euklidovým algoritmem vynásobíme prvkem α^6 tak výsledkem je stejný lokalizační mnohočlen jako u BM algoritmu [7].

7 IMPLEMENTACE V MATLABU

V první části této práce je uveden, teoretický popis způsobu kódování, dekódování BCH kódů a na jejich nejvýznamnější podskupině RS kódech bylo na příkladu znázorněn celý postup zakódování a následného dekódování. Další částí zadání bylo vytvořit aplikaci v programu Matlab, která by mohla sloužit jako výuková pomůcka a také jako analyzační nástroj. Aplikace byla vytvořena v prostředí Matlab ve verzi 2009b, při vytváření aplikace bylo vycházeno pro základní ovládaní Matlabu z [16, 17], pro jednotlivé popisy zahrnutých funkcí a příklady jejich použití bylo využíváno online dokumentace [12].

Program obsahuje základní grafické rozhraní, které slouží pro volbu mezi výukovým a simulačním programem. Program slouží pro výuku a simulaci BCH kódů v programu jsou rozlišeny binární a nebinární BCH kódy následně, binární kódy jsou označovaný jako BCH a nebinární jsou RS kódy, které jak už bylo uvedené již dříve jsou nejvýznamnější podskupinou BCH kódů.

7.1 Hlavní program



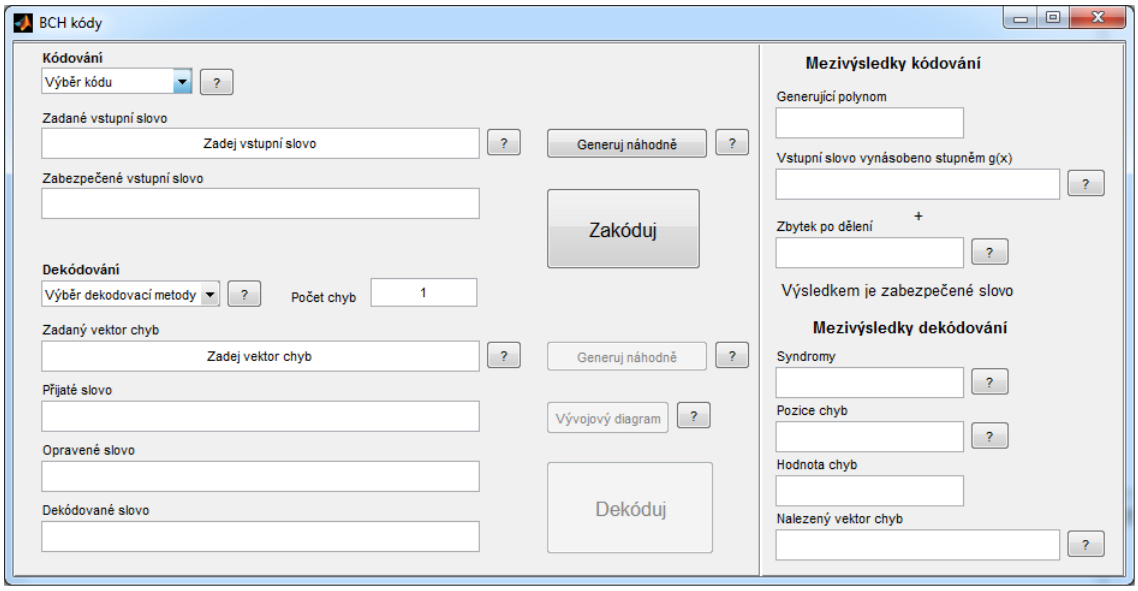
Obr. 7.1: Hlavní program

Na obrázku 7.1 je zobrazen hlavní program, který je úvodní po spuštění vytvořené aplikace. Jejím hlavním účelem je zpřístupnit uživateli volbu mezi výukovým a simulačním programem. Po volbě požadovaného podprogramu se hlavní program

ukončí a je následně spuštěn zvolený program. Po dokončení práce s následným ukončením zvoleného programu je opět automaticky spuštěn hlavní program.

7.2 Výukový program

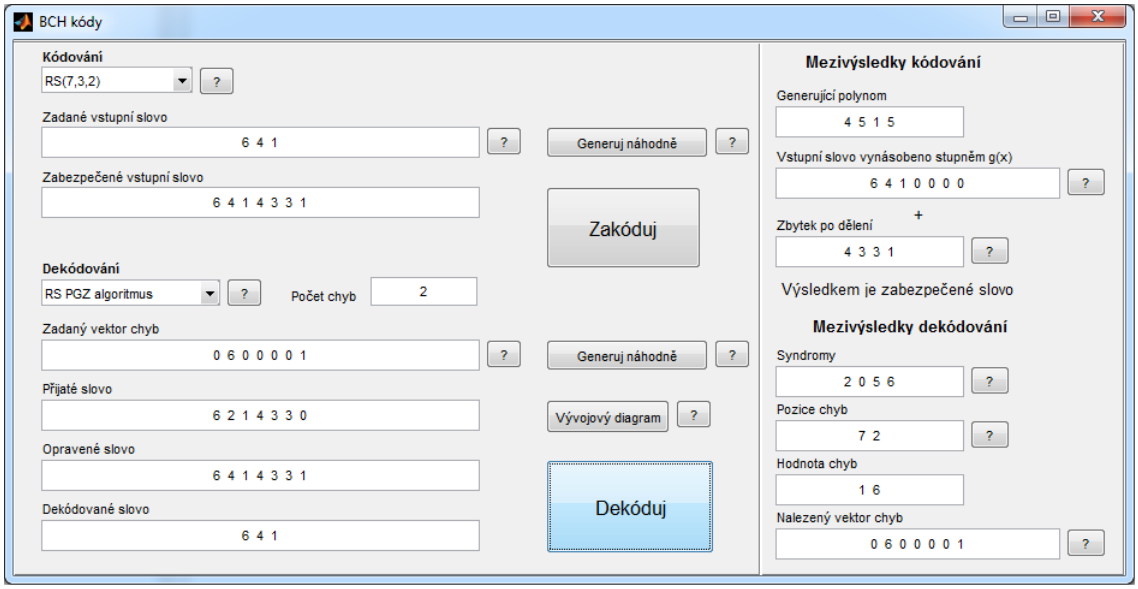
Na obrázku 7.2 je zobrazeno navržené grafické rozhraní aplikace po spuštění.



Obr. 7.2: Výuková aplikace po spuštění

Výukový program se skládá ze tří hlavních částí v první části je potřeba vybrat kód pro který se bude provádět kódování a dekódování. V aplikaci jsou na výběr nebinární kódy určené parametry $RS(n, k, t)$ jsou to $RS(7, 3, 2)$ a $RS(15, 11, 2)$. Oba kódy jsou schopny opravit $t = 2$ chyb. A pro binární kódy $BCH(7, 4, 1)$ a $BCH(15, 7, 2)$. Po výběru kódu je potřeba zadat vstupní slovo pro zakódování, je možné ho zadat ručně, kdy jednotlivé symboly GF jsou zadávány v dekadické formě (v případě binárních kódů v binární formě), odděleny mezerou. Pokud dojde ke špatnému zadání je uživatel upozorněn, aby opravil vstupní slovo. Je také možné využít náhodného vstupu pomocí tlačítka generuj náhodně. Následně je potřeba zakódovat vstupní slovo pomocí tlačítka Zakóduj. Pokud bylo zakódování úspěšné, je zobrazena zakódovaná posloupnost a v pravé horní části programu jsou zobrazeny mezivýsledky kódování jako jsou generující polynom, rozšířené vstupní slovo o stupeň generujícího polynomu a také zbytek po dělení. Uživateli je zpřístupněna

možnost dekódování. Nejprve je však vyžadováno vybrat dekódovací metodu a zadat vektor chyb (program opět upozorní na nedostatečné nebo chybné zadání), který je opět možné zadat ručně nebo generovat náhodně. Po kliknutí na tlačítko Dekóduj jsou zobrazeny v pravé části programu nalezené pozice chyb, jejich hodnoty a nalezený výsledný chybový polynom. Na obr. 7.3 je možné vidět úspěšně provedené zakódování i následné dekódování. Pro dekódování RS kódů byly implementovány algoritmy PGZ a BM, pro BCH kódy algoritmy PGZ, BM a Euklidův.

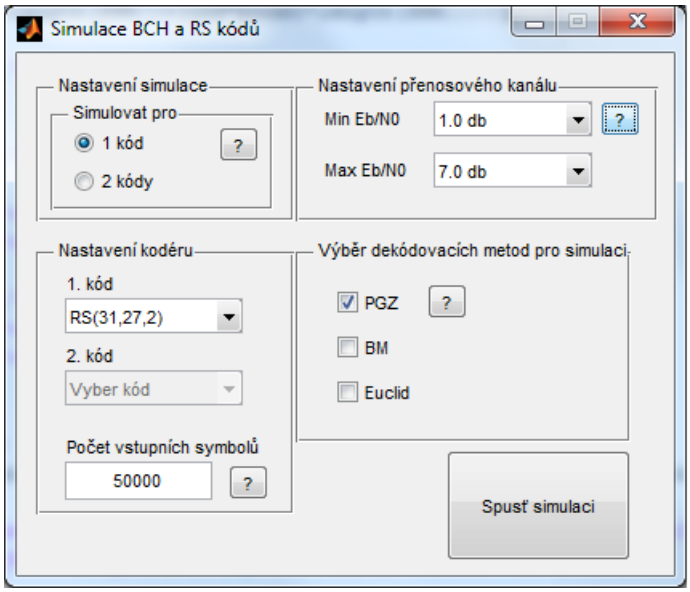


Obr. 7.3: Výuková aplikace po úspěšném dekódování

Pokud by zvolený chybový vektor obsahoval více chyb, než je kód schopen opravit je zobrazena chybová hláška – dekódování selhalo a chyby ani pozice nejsou nalezeny. Po dekódování algoritmy PGZ a BM je možno zobrazit vývojový diagram znázorňující funkci algoritmů.

7.3 Simulační program

Tento program slouží jako analyzační nástroj pro porovnání dosažených parametrů kódů, jeho grafické rozhraní je znázorněno na obrázku 7.4. Program je rozdělen do čtyř části, ve kterých je možno nastavit vstupní parametry pro simulace.



Obr. 7.4: Grafické rozhraní simulační aplikace

7.3.1 Nastavení simulace

Prvním blokem je *Nastavení simulace*, kde je možno vybrat zda simulace bude prováděna pro jeden nebo dva kódy. Podle zvoleného počtu se aktivuje volba v nastavení kodéru pro výběr konkretních kódů. Při simulaci pro dva kódy vzroste doba simulace minimálně na dvojnásobek.

7.3.2 Nastavení kodéru

zde je možno vybrat kódy pro které se bude simulovat. Kódy které je možno vybrat s jejich parametry jsou uvedeny v tabulce 7.1. Např. kód $RS(255, 239)$, je standardně používán v zabezpečení FEC definovaném ITU-T G.709 s použitím v optických sítí podporující až 10 Gb/s. Pole *Počet vstupních symbolů* je určeno k zadání počtu symbolů (pro BCH kódy je to počet bitů), které budou přivedeny na vstup kodéru. Při zadání nízkého počtu vstupních symbolů výsledné grafy nebudou vycházet adekvátně, ale při zadání velkého počtu může doba simulace trvat i několik hodin, proto je doporučeno zadávat minimálně 2000 vstupních symbolů, ideálně 10000. Vstupní hodnoty symbolů jsou generovány náhodně.

Tab. 7.1: Parametry kódů s informační rychlostí

Kód(n,k,t)	R
RS(7,3,2)	0,429
RS(15,11,2)	0,733
RS(31,27,2)	0,871
RS(127,111,8)	0,937
RS(255,239,8)	0,874
BCH(7,3,1)	0,571
BCH(15,7,2)	0,467
BCH(63,39,4)	0,619
BCH(255,207,6)	0,812
BCH(255,223,4)	0,875

7.3.3 Nastavení přenosového kanálu

pro přenos informací a simulaci reálného kanálu byl vytvořen AWGN kanál s BPSK modulací. Můžeme nastavit rozmezí E_b/N_0, pro které bude simulace prováděna.

7.3.4 Výběr dekódovacích metod

pro srovnání výkonnosti dekódovacích algoritmů je možné vybrat ze tří dekódovacích algoritmů. Pro RS kódy byly implementovány algoritmy PGZ a BM, pro binární BCH kódy PGZ, BM a Euklidův algoritmus. Při výběru RS kódu a Euklidova algoritmu je po kliknutí na tlačítko simulace uživatel upozorněn *Euklidův algoritmus nebyl implementován* a simulace nebude spuštěna.

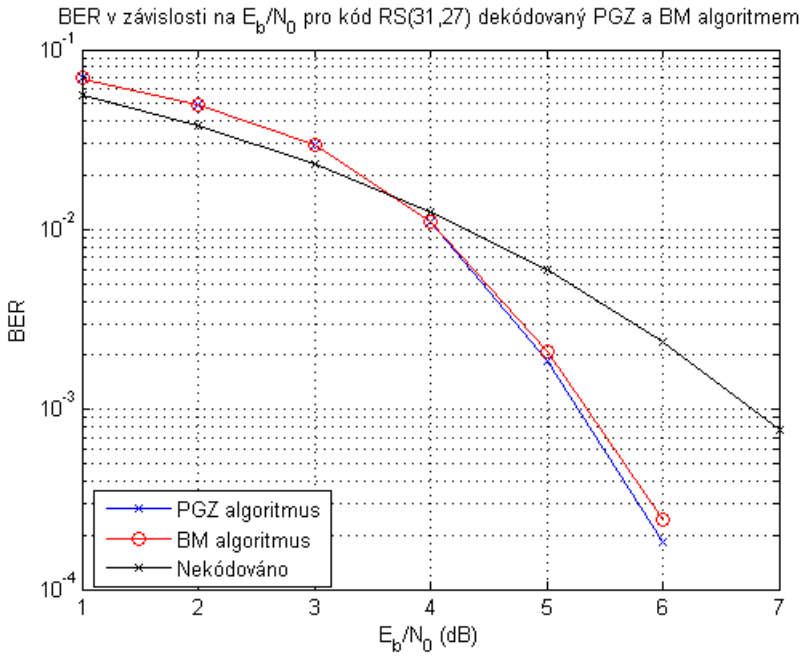
Veškeré části simulačního programu jsou doplněny tlačítky pro zobrazení nápovědy. Při špatném nebo nedostatečném zadání je uživatel upozorněn a vyzván k opravení nastavení. Při správném nastavení je zobrazen tzv. *progres bar*, který zobrazuje postup simulace, po prvním provedení výpočtu jednoho kroku (zakódování, přenesení a dekódování) je zobrazen přibližný čas do konce simulace. Výstupem simulace je BER v závislosti na E_b/N_0. Výsledné grafy se automaticky ukládají do adresáře programu ve formátu *.fig, se jménem např. 100000_BM_kód BCH(255,223).fig, kde 100000 je počet vstupních symbolů, BM – dekódováno BM algoritmem, pro kód BCH(255,233).

8 SIMULACE

V této kapitole bude rozebráno zevrubné srovnání dosažených parametrů, pomocí vytvořeného simulačního nástroje. Veškeré simulace pro jednotlivé kódy byly simulovány se vstupním počtem symbolů 100000, aby bylo dosaženo co nejpřesnějších výsledných grafů.

8.1 Srovnání výkonnosti dekódovacích metod

8.1.1 Pro RS kódy

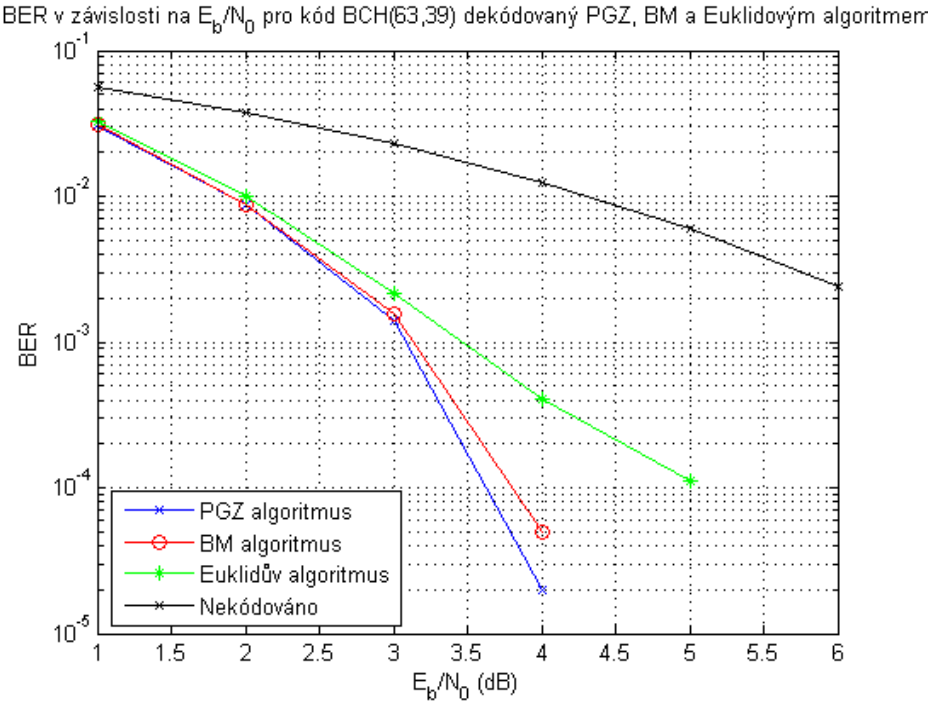


Obr. 8.1: Srovnání výkonnosti PGZ a BM algoritmu na kódu $RS(31, 27)$

Na obrázku 8.1 můžeme vidět srovnání výkonnosti PGZ a BM algoritmu na kódu $RS(31, 27)$, který dokáže opravit 2 chyby. Jde vidět, že výkonnostně se dekódovací algoritmy liší jen nepatrně. Tento kód dokáže zajistit nulovou chybovost pro $E_b/N_0 >$ 6dB.

8.1.2 Pro BCH kódy

Na obrázku 8.2 můžeme vidět srovnání výkonnosti PGZ, BM a Euklidova algoritmu na kódu $BCH(63, 39)$, který dokáže opravit až 4 chyby. Opět jak pro RS kódy se algoritmy PGZ a BM téměř neliší. PGZ a BM algoritmus dokáží opravit všechny chyby už pro $E_b/N_0 > 4$dB a Euklidův algoritmus pro $E_b/N_0 > 5$dB.

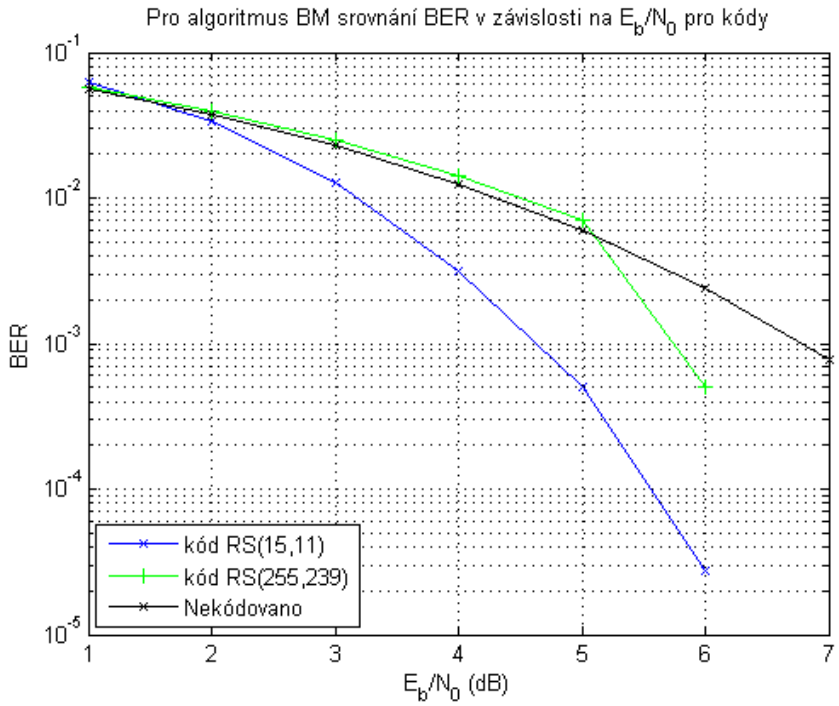


Obr. 8.2: Srovnání výkonnosti vybraných algoritmu na kódu $BCH(63, 39)$

8.2 Srovnání kódů pro různé parametry

8.2.1 Pro RS kódy

Na obr. 8.3 můžeme vidět srovnání kódů $RS(15, 11)$ a $RS(255, 239)$ dekódované BM algoritmem, i když má kód $RS(15, 11)$ menší korekční schopnost vykazuje menší chybovost z důvodu kratší kódové délky a proto je menší pravděpodobnost vzniku více chyb v jednom kódovém bloku. Avšak oba kódy dokáží zajistit nulovou chybovost od $E_b/N_0 > 6$dB.



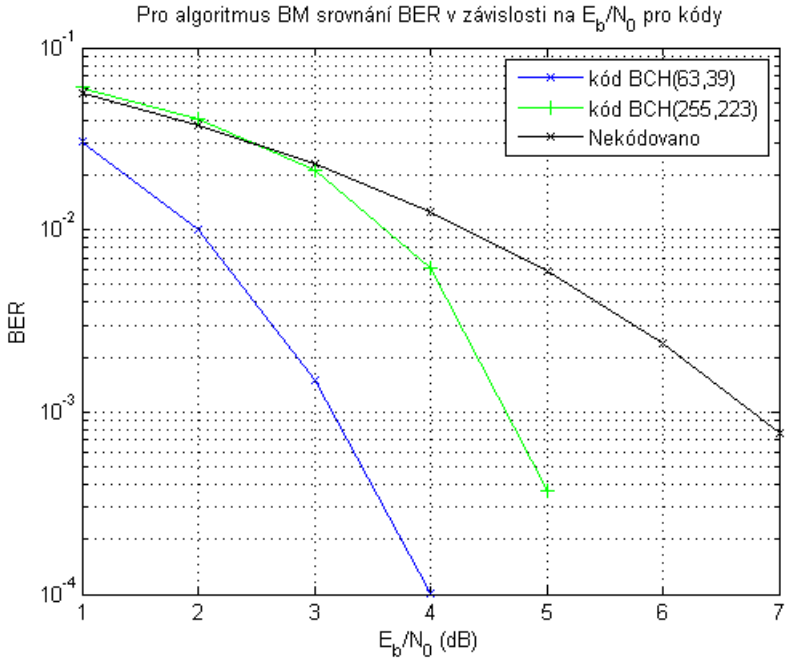
Obr. 8.3: Srovnání kódů $RS(15, 11)$ a $RS(255, 239)$ dekódované BM algoritmem

8.2.2 Pro BCH kódy

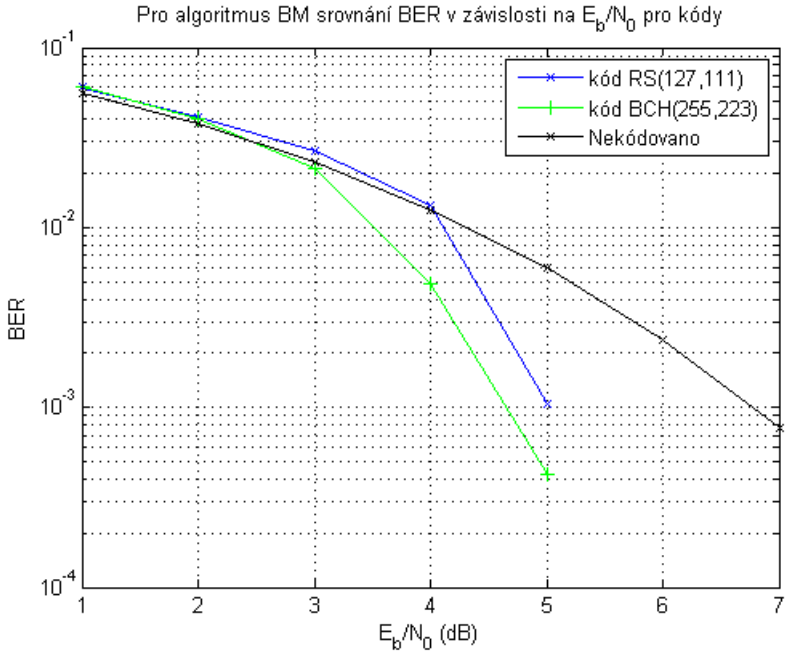
Pokud provedeme srovnání jako u RS kódů, kde použijeme kódy různých blokových délek, které dokáží opravit stejný počet chyb na kódech $BCH(63, 39)$ a $BCH(255, 223)$ vidíme, že opět kód s kratší blokovou délkou má menší chybovost a dokáže zajistit nulovou chybovost už od $E_b/N_0 > 4$ a delší o 1 dB později viz obr.8.4.

8.2.3 Srovnání BCH a RS kódů

Srovnání kódů $RS(127, 111)$ a $BCH(255, 207)$ je zobrazeno na obr. 8.5, oba kódy mají přibližně stejnou informační rychlost $R = 0, 875$, ale různou délku. Jak můžeme vidět tak oba kódy dokáží zajistit nulovou chybovost od $E_b/N_0 > 5$dB.



Obr. 8.4: Srovnání kódů $BCH(63, 39)$ a $RS(255, 223)$ dekódované BM algoritmem



Obr. 8.5: Závislost BER na E_b/N_0 pro kódy $RS(127, 111)$ a $BCH(255, 223)$

8.3 Zhodnocení výsledků simulací

V této kapitole byly prezentovány výsledky simulací, provedené pomocí vytvořeného programu v Matlabu. Srovnáme-li výsledky s jinými např. v literaturách [3, 10], je možné říci, že téměř odpovídají teoretickým předpokladům.

Při využití navrženého systémů kodéru využívající BCH kódy (nebo RS kódy), přenosového AWGN kanálu s BPSK modulací a následným dekódováním, bylo předvedeno, že BER výkonnost se zlepšuje při snižování informační rychlosti. Při použití kódu, který má kratší blokovou délku a menší korekční schopnost je dosaženo menší chybovosti než-li u kódů s delší blokovou délkou a větší korekční schopností, to je způsobeno tím, že kratší kód dokáže ve výsledku opravit více chyb při velkém vstupu. Při stejném počtu vstupujících dat do dekodéru by však u kratších kódů bylo potřeba provést daleko více výpočtů a tím by narůstalo i zpoždění nebo se zvyšovali výpočetní požadavky na dekodér. Proto se v praxi volí kompromis mezi délkou kódu a korekční schopností podle daného prostředí, kde zabezpečovací kodek bude použit.

Další možností jak zlepšit korekční schopnost přenosového systému je využitím prokladače, který odstraní shlukové chyby a tím se dosáhne lepších výsledků, tato funkce však opět zvyšuje počet prováděných operací a tím i zpoždění přenosu.

9 ZÁVĚR

Úkolem této diplomové práce, bylo nastudování problematiky protichybového zabezpečení pomocí BCH kódů. V kap. 1, byl uveden obecný popis BCH kódů spolu s nezbytnými znalostmi kódování a lineární algebry. V kapitolách 2 až 4 jsou popsány vlastnosti binárních a nebinárních BCH kódů a jejich způsob zakódování. Kapitola 5 obsahuje popsáné jednotlivé kroky dekódování a jejich teoretické postupy výpočtů. Nejvíce obsáhlým krokem pro dekódování je nalezení chybového mnohočlenu, který je následně využit pro nalezení pozice a hodnoty chyb. Pro nalezení lokalizačního mnohočlenu jsou v této práci popsány algoritmy Peterson-Goreistein-Ziegler 5.2.1, Berlekmap-Massey 5.2.2 a Euklidův 5.2.3.

Jednou z nejvíce významnou podskupinou BCH kódů jsou Reed-Solomonovy kódy viz kap.6, které mají možnost zabezpečit posloupnost dat po např. celém bajtu. Na konkretním Reed-Solomonovu kódu RS(7,3), byl v kapitole 6.1, předveden způsob zakódování a jeho následné dekódování s využitím Peterson-Goreistein-Zieglerova algoritmu 6.2.2, v kap. 6.2.3 Berlekmap-Massey a nakonec i Euklidovým algoritmem v kap.6.2.4, pro nalezení pozic chyb bylo využito popsaného Chienova výhledávání a nalezeny hodnoty chyb využitím Forneyova algoritmu.

Další částí zadání bylo vytvořit aplikaci v programu Matlab, která bude demonstrovat zabezpečovací proces a bude koncipována jako výuková pomůcka a zároveň sloužit jako analyzační nástroj pro porovnání parametrů kódů. Podle požadavků zadání byla vytvořena aplikace, která má součástí jednoduché grafické prostředí doplněno nápovědou. Vytvořená aplikace byla popsána v kapitole 7.

V poslední části práce bylo pomocí vytvořeného simulačního programu provedeno zevrubné srovnání pro různé parametry kódů. Po srovnání různých parametrů kódů, můžeme říci, že kódy s kratší blokovou délkou dokáží zajistit lepší chybovost než kódy delší což je způsobeno menší pravděpodobností vzniku více chyb u kratší posloupnosti. Při použití BCH i RS kódů dochází k zajištění nulové chybovosti při $E_b/N_0 > 5$dB, kdežto nekódovaná posloupnost dosahuje chybovosti menší než je 10^{-4} až od $E_b/N_0 > 8$. Proto i když použitím zabezpečovacích kódů zvýšíme redundanci a tím i potřebnou větší šířku pásma. Nebo při použití stejné šířky pásma dosáhneme menší informační rychlosti, ale tím zajistíme mnohem dříve bezchybovost. Věřím, že tímto byly splněny veškeré cíle zadání.

LITERATURA

[1] ADÁMEK, J. *Kódování a teorie informace.* Praha : skriptum ČVUT, 1991. ISBN:80-01-00661-1.

[2] ČÍKA, P. *Protichybové zabezpečení BCH kódem* Publikace v internetovém magazínu Elektrorevue. [cit. 1.2.2012]. Dostupné na URL: http://www.elektrorevue.cz/clanky/06015/index.html

[3] HANZO, L. *Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels.* JohnWiley, 2002, ISBN: 0470847263 766

[4] LEE Ch. *Error-Control Block Codes for Communications Engineers* Artech House, 2000, ISBN 13: 978-158-0-53032-3

[5] LIN, S., COSTELLO, D.J. *Error Control Coding: Fundamentals and Applications, second edition.* Prentice Hall: Englewood Cli_s, NJ, 2005, ISBN: 0130426725.

[6] MOON, T.K. *Error Correction Coding: Mathematical Methods and Algorithms.* Wiley-Interscience, 2005, ISBN: 0471648000.

[7] MOREIRA, J.C *Essentials of error-control coding* JohnWiley, 2006, ISBN-13: 978-0-470-02920-6

[8] NĚMEC, K. *Datová komunikace.* skriptum VUT-BRNO, 2007.

[9] NĚMEC, K. *Protichybové kódové zabezpečení s Bose-Chauhury-Hocquenhemovými kódy.* Publikace v internetovém magazínu Elektrorevue. [cit. 10.11.2011]. Dostupné na URL: http://www.elektrorevue.cz/clanky/05015/index.html

[10] SANJEEV, K. *Bit Error Rate Analysis of Reed-Solomon Code for Efficient Comunication System.* International Journal of Comuter Applications, Vol. 30, No.12 September 2011 Dostupné na URL: http://research.ijcaonline.org/volume30/number12/pxc387513.pdf

[11] SWEENEY, P. *Error Control Coding: From Theory to Practice.* Wiley-Interscience, 2002, ISBN: 047084356X.

[12] Matlab Product Documentation - Galois Field [online]. [cit. 10. 12. 2011]. Dostupné na URL: http://www.mathworks.com/help/toolbox/comm/ref/a1037894415.html#fp9427

[13] WALLACE, H. *Error Detection and Correction Using the BCH Code* [online]. [cit. 1. 2. 2012]. Dostupné na URL: http://www.aqdi.com/bch.pdf

[14] Wicker B. S. *Reed-Solomon Codes and Their Applications* Wiley-IEEE Press, 1999, ISBN 13: 978-0-780-35391-6

[15] YUANG JIANG *A Practical Guide to Error-Control Coding Using MATLAB* Artech House, 2010, ISBN 13: 978-1-60807-088-6

[16] ZAPLATÍLEK, K., DOŇAR, B. *MATLAB - tvorba uživatelských aplikací* BEN - technická literatura, Praha 2004, ISBN 80-7300-133-0

[17] ZAPLATÍLEK, K., DOŇAR, B. *MATLAB - začínáme se signály* BEN - technická literatura, Praha 2006, ISBN 80-7300-200-0

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

$AWGN$ Aditive White Gaussian Noise

BCH Bose-Chaudhuri-Hocquenghem

$BPSK$ Binary Phase Shift Keying

BER Bit Error Rate

d_{min} minimální kódová vzdálenost

E_b/N_0 energie bitu ku spektrální výkonové hustotě šumu

FEC Forward Error Correction

$GF(2^m)$ Galoisovo těleso

k počet symbolů nezabezpečené zprávy

n počet symbolů zabezpečené zprávy

t korekční schopnost

v počet očekávaných chyb

α prvek Galoisova tělesa

$c(x)$ mnohočlen zabezpečené zprávy (kódové slovo)

$d(x)$ mnohočlen zbytku po dělení

$e(x)$ chybový mnohočlen

$g(x)$ vytvářecí (generující) mnohočlen

$m(x)$ mnohočlen podílu

$p(x)$ mnohočlen nezabezpečené zprávy

$r(x)$ mnohočlen přijaté zprávy

M_l hodnota chyb

P_i pozice chyb

S_j Syndromy

$A(x)$ pomocný chybový mnohočlen

$E(x)$ mnohočlen pro určení hodnoty chyb

$T(x)$ dočasný chybový mnohočlen

$\Lambda(x)$ chybový lokalizační mnohočlen

Θ libovolný polynom

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

Na přiloženém CD se nachází výsledný text diplomové práce ve formátu PDF spolu se zdrojovými kódy pro výukový a simulační program vytvořený v prostředí Matlab (M-file). Tyto soubory se nacházejí ve složce //Matlab/BCH_kody_zdrojove_soubory. Dále je zde zkompilovaná verze programu (BCH_kody.exe) pro operační systém Windows. Zkompilovaná aplikace se nachází //Matlab/BCH_kody/src/. Spolu se spustitelnou aplikací se v této složce nachází program (MCRInstaller.exe), který je potřeba nainstalovat pokud není v počítači nainstalováno prostředí Matlab.