ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ
KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

DIPLOMOVÁ PRÁCE

Implementace algoritmů density evolution

Dominik Škubla **2017**

zadání

Abstrakt

Předkládaná diplomová práce se zabývá prostudováním a implementací algoritmů density evolution pro ohodnocení výkonnosti opravných kódů LDPC. LDPC kódy jsou jen stručně popsány, podstatnou částí je popisování a naprogramování knihoven density evolution v diskretizované a aproximované verzi. V závěru práce je porovnána časová náročnost kódů a rozdíl jejich výstupů.

Klíčová slova

Iregulární LDPC kódy, Regularní LDPC kódy, Density evolution, Threshold, Gaussovská aproximace, Pravděpodobnostní funkce, Kvantizace.

Abstract

This master thesis deals with research and implementation of density evolution algorithms for evaluating the performance of LDPC correction codes. The LDPC codes are briefly described and the essential part of this thesis is the description and programming of density evolution libraries in a discretized and approximated version. At the end of the thesis, the computing time required for specific simulations is measured and the outputs are compared.

Key words

Irregular LDPC codes, Regular LDPC codes, Density evolution, Gaussian approximation, Probability mass function, Quantization.

Prohlášení

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

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

V Plzni dne 16.5.2017 Dominik Škubla

.....

podpis

Obsah

SEZNAM ZKRATEK

BSC.....	Binary Symmetric Channel
BEC.....	Binary Erasure Channel
BSEC.....	Binary Symmetric and Erasure Channel
AWGN.....	Addtive White Gaussian Noise
ISO/OSI.....	International Standards Organization/Open System Interconnection
LDPC.....	Low Density Parity Check
LLR.....	Log-Likelihood Ratio

Seznam obrázků

Seznam tabulek

1 ÚVOD

Tato práce se zabývá algoritmy density evolution a jejich implementací. Tyto algoritmy jsou schopné ohodnotit výkonnost opravných kódů rodiny LDPC. Po získání výsledků je možné tuto výkonnost vhodně navýšit úpravou LDPC kódů a tím získat lepší vlastnosti. Hlavní část práce se zabývá prostudováním a implementací dvou rozdílných algoritmů density evolution. Jedná se o diskretizovanou a aproximovanou verzi density evolution. Cílem této práce je pak zejména porovnání časové náročnosti výpočtu obou algoritmů a rozdíly chybovosti výstupů mezi diskretizovanou a aproximovanou verzí. Přiloženy jsou knihovny s použitými algoritmy a funkcemi.

Teoretická část je rozdělena do několika na sebe navazujících kapitol. V první části jsou popsány některé základní pojmy a dva abstraktní modely používané v datových komunikacích- Shannonův a ISO/OSI model. Na ni navazují dvě kapitoly, které se postupně zabývají LDPC kódy a Tannerovy grafy. V dalších kapitolách jsou již rozebírány už samotné density evolution algoritmy pro regularní a iregulární kódy. Nejdříve je popsána gaussovská aproximace a na ni následně navazuje diskretizovaná verze density evolution. V předposlední kapitole jsou uvedeny výsledky naprogramovaných algoritmů. V závěru jsou jednotlivé algoritmy porovnány a je diskutována možnost využití těchto kódů v souvislosti s optimalizačními algoritmy pro návrh distribučních funkcí hodností uzlů Tannerova grafu.

2 Digitální komunikace

2.1 Teorie přenosu signálu

Základním úkolem přenosu je přenést danou informaci od vysílače (zdroje) k přijímači (příjemci). Předpokládáme zprávu A u vysílače, která se liší od zprávy B přijímače vlivem různých poruch a rušení v přenosovém kanálu, které vznikají například nedokonalým vedením nebo vlivem okolních zařízení na vedení. Tomuto rušení se snažíme zabránit různými opatřeními. Příkladem mohou být opravné kódy LDPC.

2.1.1 Informace a Entropie

Zdrojem informace může být například člověk nebo technické zařízení. Je to abstraktní pojem, který vyjadřuje obsah sdělení. Tím rozumíme, že se jedná o sdělení stavu objektu, instrukci pro čidlo a mnoho dalších. Informace může mít různé podoby, a to například verbální, textovou nebo obrazovou. Množství informace I pro diskrétní zprávu o n prvcích, z nichž každý může nabývat m různých stavů o pravděpodobnosti p_i, je definována jako:

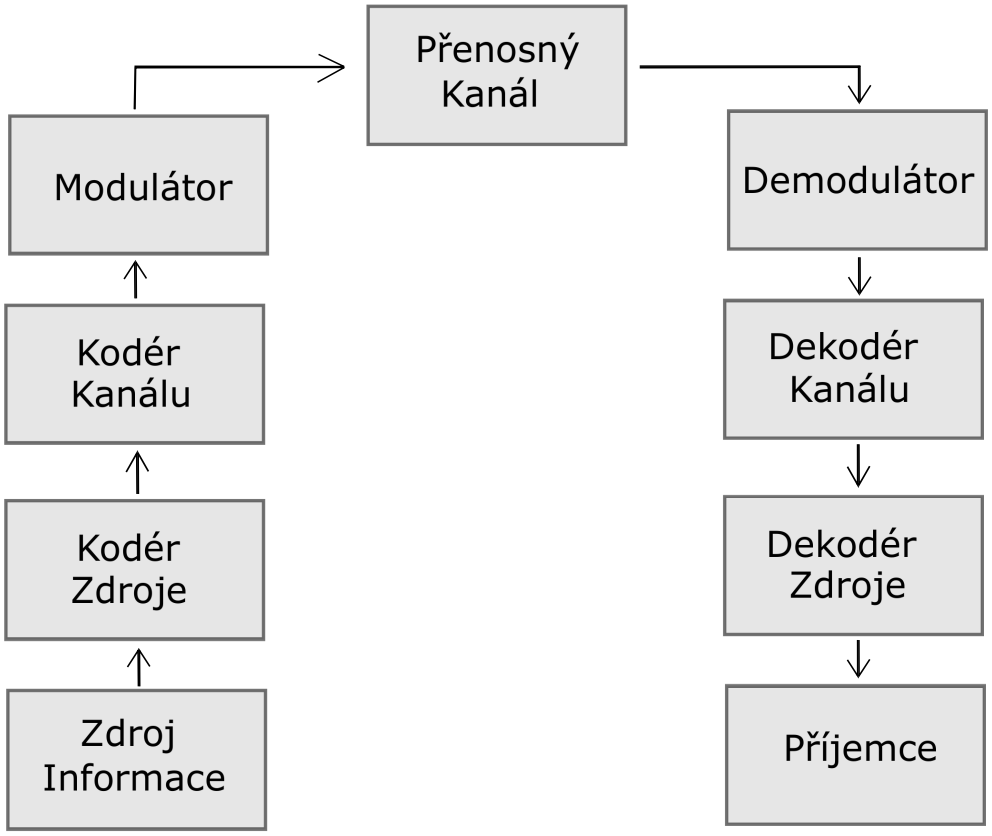
$$I = -n \sum_{i=1}^{m} p_i \log_2 p_i \tag{2.1}$$

Informační entropii jako první definoval Claude Elwood Shannon v roce 1948 jako střední hodnotu informace na jeden symbol zprávy. S pojmem entropie se tedy setkáme všude, kde hovoříme o pravděpodobnosti možných stavů daného systému či soustavy. Entropie a redundance souvisí s kompresí dat. Entropie se značí H a vypočítá se:

$$H(x) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{2.2}$$

2.2 Základní blokové schéma komunikačního systému podle Shannona

Shannon v 50. letech 20. století dokázal, že všechny komunikační systémy se dají zobrazovat v obecném blokovém komunikačním schématu. Toto schéma je zobrazeno do řetězce.



Obrázek 2.1: Shannonovo blokové schéma komunikačních systémů

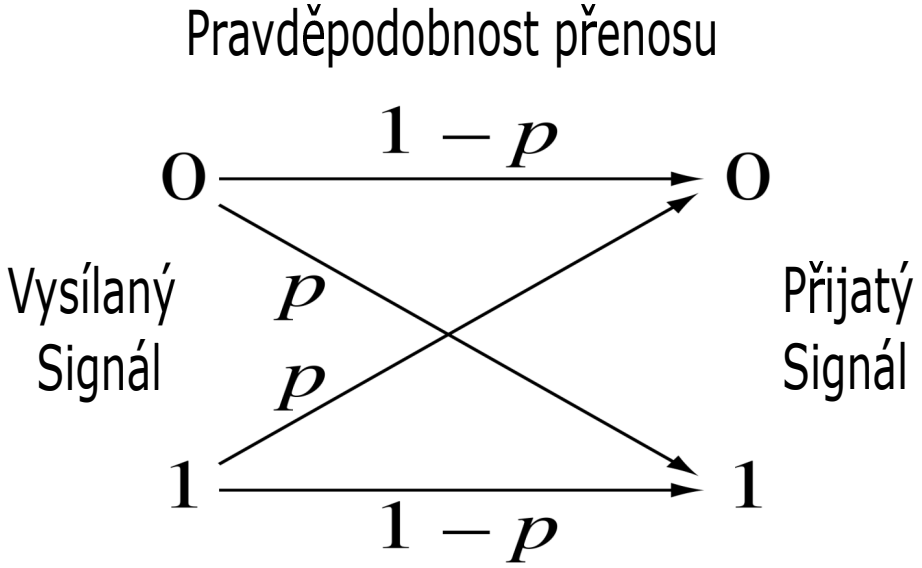
- Zdroj informace - generuje zprávy nebo jejich posloupnosti, které jsou následně určeny k přenosu k přijímači.
- Kodér zdroje - provádí kódování zprávy. Kódování zprávy se může provádět za účelem šifrováním, anebo za účelem komprese, aby měla zpráva pro přenos co nejmenší počet znaků. Jinak řečeno redukuje redundanci zprávy a zvyšuje její entropii.
- Kodér kanálu - zajišťuje spolehlivost přenosu, a to tím, že přidá redundantní informaci, která pomáhá na straně přijímače odstranit případné chyby a šumy vzniklé při přenosu.
- Modulátor - má za úkol upravit signál tak, aby byl přenositelný kanálem.
- Přenosový kanál - medium, po kterém se daná informace přenáší. Dochází zde k poruchám, nebo se zanáší šum do informace.

- Demodulátor - převádí přijatá data z kanálu na posloupnost symbolů, která je srozumitelná pro dekodér kanálu.
- Dekodér kanálu - opraví chyby vzniklé při přenosu za pomoci redundantní informace.
- Dekodér zdroje - převede signál do podoby před přidáním redundantní informace pro zabezpečení přenosu. Jinak řečeno, převede signál do podoby, aby mu příjemce rozuměl.

2.2.1 Modely kanálu

Modely kanálu se mmohou dále rozdělit podle typu vysílaných dat buď na diskrétní kanál nebo spojitý kanál. Diskrétní kanál se také nazývá číslicový nebo digitální kanál, a jeho signál je nespojitý v čase a v amplitudě. Spojitý kanál přenáší spojitý signál v čase a s vlivem Gaussova aditivního šumu. Tato práce se především zaměřuje na diskrétní kanál, který můžeme následně rozdělovat.

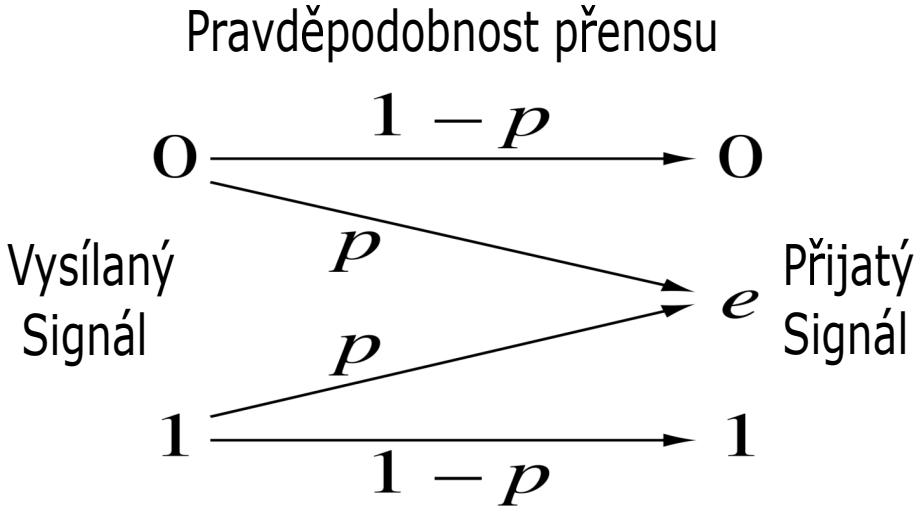
BSC (Binary Symmetric Channel) - jedná se o diskrétní kanál bez paměti, který je charakterizován určitou pravděpodobností chybovosti přenosu. Do tohoto kanálu je vysláno slovo a s určitou pravděpodobností se některé bity změní z 0 na 1 a naopak.



Obrázek 2.2: Binary Symmetric Channel

p - je pravděpodobnost chybovosti

BEC(Binary Erasure Channel) - diskrétní kanál s určitou pravděpodobností chybové detekce. Do tohoto kanálu je vysláno slovo a s určitou pravděpodobností se některé bity vymažou, takže výsledné slovo nebude dávat smysl.

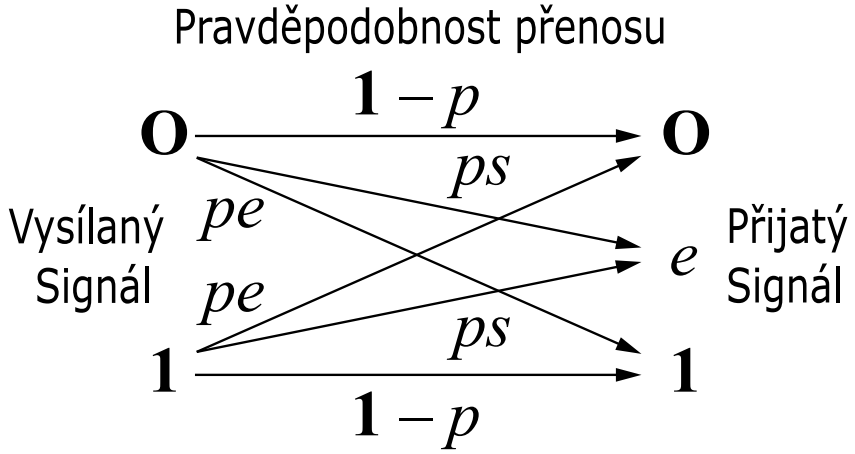


Obrázek 2.3: Binary Erasure Channel

p - je pravděpodobnost výmazu

e - symbol je v tomto případě smazán, anebo se nedá určit

BSEC(Binary Symmetric and Erasure Channel)- je ve své podstatě kombinace dvou předchozích kanálů (BSC a BEC)



Obrázek 2.4: Binary Symmetric and Erasure Channel

ps - je pravděpodobnost chyby

pe - je pravděpodobnost výmazu

e - symbol je v tomto případě smazán, anebo se nedá určit

p - je součet pravděpodobnosti chybovosti a výmazu.

AWGN (Addtive White Gaussian Noise) - je jeden z nejznámějších kanálů, který je používaný především na simulace a k porovnávání efektivity přenosových systémů. V tomto kanálu se nejedná o zeslabení přenosového kanálu nebo zanesení jiných chyb jako únik, vícecestné šíření informace a tak dále. U AWGN kanálu je jediný zdroj chyb, a to je aditivní šum, který je širokopásmový (ideálně bílý šum) s gaussovsky rozloženou amplitudou.

Rayleigh fading - Rayleigh únik vzniká tak, že signál od zdroje nemá přímou cestu k přijímači. Přijímač může být například zastíněn. Dále se zde uvažuje s Dopplerovým jevem a mnohacestným šířením vln, k němuž dochází kvůli odrazům vln od okolních objektů. Tento jev má za následek efekt, který způsobuje hluboké a rychlé kolísání signálu. Tento kanál se ze všech výše zmiňovaných kanálů blíží nejvíce k reálnému kanálu.

2.2.2 Vlastnosti komunikačního systému

Jedním z nejdůležitějších vlastností komunikačního kanálu je modulační rychlost v_m. Vypočítá se pomocí rovnice (2.3).

$$v_m = \frac{1}{a} \tag{2.3}$$

Modulační rychlost určuje počet signálových prvků přenesených za jednotku času. Jinak řečeno, je to rychlost, kterou se mění jednotlivé stavy signálu. Rychlost změny signálu prvků v kanálu je omezena šířkou pásma kanálu. Proměnná a určuje délku trvání charakteristického intervalu.

Přenosová rychlost je značena v_p a je dána počtem bitů informace v jednotkovém intervalu přenášených za jednotku času.

$$v_p = v_m \log_2(Q) \tag{2.4}$$

Kde Q určuje počet stavů signálu.

Další důležitou jednotkou kanálu je kapacita, která se vypočítá podle rovnice (2.5).

$$C = B \log_2(1 + \frac{S}{N}) \tag{2.5}$$

Jedná se o maximální dosažitelnou rychlost v daném kanálu. Jinak řečeno, je to prostředí, které představuje horní limit pro přenosovou rychlost a nelze ho překročit. Proměnná B určuje šířku pásma kanálu a $\frac{S}{N}$ je poměr výkonu signálu k výkonu šumu, kde S je tedy střední hodnota výkonu signálu a N je střední hodnota výkonu šumu.

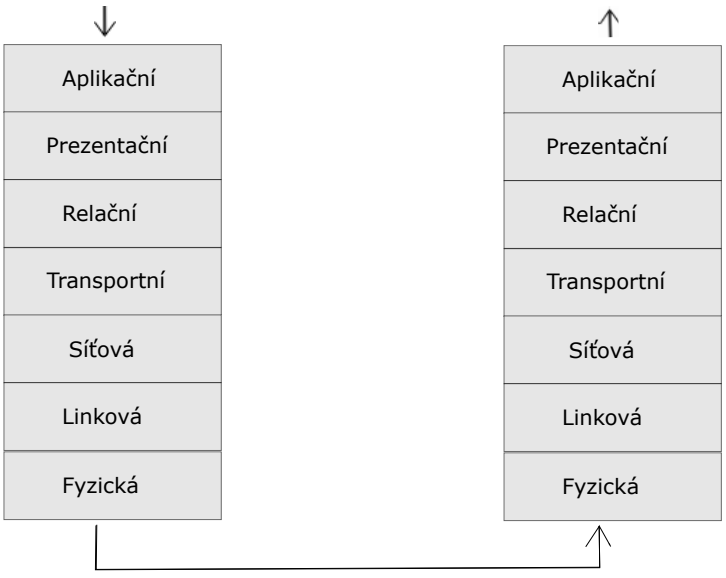
Informační poměr R udává velikost poměru informace k délce kódového slova. Vypočítá se podle rovnice (2.6).

$$R = \frac{\log_q |c|}{n} = \frac{\log_q |q^k|}{n} = \frac{k}{n} \tag{2.6}$$

Kde q určuje počet prvků abecedy, k je počet prvků informační zprávy a n je délka kódového slova.

2.3 ISO/OSI Model

ISO/OSI je označení Mezinárodní organizace pro normalizaci / propojení otevřených systémů (International Standards Organization / Open Systen Interconnection). Tento model byl definován organizací ISO v roce 1983 za účelem standardizace počítačových sítí. ISO/OSI rozděluje standardní komunikaci mezi dvěma zařízeními do sedmi vrstev, někdy nazývaných jako protokoly. Každá vrstva pak dále přebírá informace od nižší vrstvy a vykonává nebo poskytuje určité služby další vyšší vrstvě. V tomto procesu nezatěžuje nižší vrstva vrstvu vyšší žádnými podrobnými informacemi. Předávají se jenom nezbytné informace pro jejich správnou činnost. Tím se zajišťuje jisté odlehčení složitosti a lepší funkčnost sítě. V každé vrstvě se pak postupně přidávají další informace k základní zprávě v podobě packetu. Následně pak záleží na druhu a typu komunikace. Komunikace se mohou lišit v v důsledku toho, které vrstvy jsou, či nejsou použity.



Obrázek 2.5: ISO/OSI Model

- Fyzická vrstva - definuje přenosové médium, konektory, elektrické úrovně, tvar pulzu a synchronizace. Doporučenými standardními obvody můžou být například RS 485 nebo 232 a další.
- Linková vrstva - slouží pro vytváření rámců, rozpoznání konce a začátku rámce, délky přenášených rámců, způsob detekce chyb a zabezpečení dat. Případně pro určení pravidel přenosu na sběrnici a automatické opakování přenosu.
- Síťová vrstva - pracuje s packety. Slouží k jejich směrování a předávání řetězcem stanic. Jinými slovy, je zde zajištěno předání dat do správných uzlů. K tomuto účelu se ke zprávě přiřadí adresa, rozlišení priorit zpráv.
- Transportní vrstva - zajišťuje bezpečný přenos mezi stanicemi. Rozložení a složení packetů, kontrolu, potvrzení, příjem a opakované vysílání v případě chyby.
- Relační vrstva - řídí interakce mezi procesy, jejich rozdělení mezi stanice, případná pravidla.
- Prezentační vrstva - provádí konverzi dat mezi obecným formátem, definovaným protokolem a formátem definovaným pro daný koncový systém. Převádí kódy, kompresuje data, šifruje data a mnoho dalšího.
- Aplikační vrstva - poskytuje rozhraní k uživatelskému softwaru. To znamená, že uživateli jsou definovány určité funkce, například pro přenos souborů nebo další. Uživatel nemusí vědět nic o síti a co se na ní odehrává.

3 LDPC kódy

3.1 Kódy, kódování

Kód je množina symbolů, která vyjadřuje jednotlivé stavy systému a je předem stanovena pomoci tabulky nebo dohodnutých pravidel. Kódování je činnost, při které využíváme kódy a převádíme pomocí nich symboly na jiné symboly za pomocí tabulky nebo předem dohodnutých pravidel. Je velmi důležité, že se při kódování může změnit množství znaků a tím se změní i pravděpodobnost výskytu jednotlivých symbolů. Během tohoto procesu dochází ke změně redundance. Je tedy zřejmé, že kodér zdroje redundanci odstraňuje, protože redundanci není nutno přenášet. Je možné ji při dekódování do zprávy opět doplnit. Kodér kanálu následně redundanci přidává, aby zabezpečil spolehlivost přenosu. V případě reálných kódů se musí vždy uvažovat s jistým druhem rušení. Proto se do kódů v kodéru kanálu záměrně zavádí redundance, která umožňuje tyto chyby detekovat a případně udělat korekci.

3.2 Rozdělení kódů

Kódy se můžou dělit různými způsoby (více informací [1]). LDPC kódy (které budou popsány v kapitole 3.3) dělíme na regulární (rovnoměrné) a iregulární (nerovnoměrné), kde regulární kódy mají všechny znaky (symboly) tvořeny stejným počtem znaků. Naopak iregulární mají znaky tvořené různým počtem symbolů. Postupem času se vytvořilo hned několik základních kódů, a to telegrafní kód, ISO-7, ASCII, atd. V souladu se zvyšovanými nároky na přenos a uchování informace jsou čím dál tím podstatnější bezpečnostní kódy, které můžeme dělit na detekční a korekční. (Cyklické kódy, Hammingův kód, LDPC kódy)

3.3 Definice LDPC kódů

LDPC (Low-Density Parity-Check) kód je lineární kód pro detekci a opravu chyb. Lineární kódy se vyznačují tím, že lineární kombinace dvou nebo i více kódových slov je opět kódové slovo. LDPC kód byl poprvé publikován již roku 1962 Robertem Gallagerem. I proto se v mnoha literaturách nazývá Gallagerův kód. LDPC kódy byly na svojí dobu tak nadčasové, že upadly na necelých 30 let do zapomnění, protože tyto kódy byly příliš náročné a nevyužitelné pro tehdejší hardware. Roku 1981 definoval Michael Tanner grafickou interpretaci paritní matice H, což ovšem stále nevzkřísilo LDPC kódy. Ty byly vzkříšeny až v 90. letech profesorem Univerzity Cambridge Davidem MacKayem, který dokázal jejich výborné opravné schopnosti a také to, že se LDPC kódy přibližují k Schannonovu kapacitnímu limitu s délkou kódu (pro více informací [4]). Dnes se LDPC a turbo kódy považují za jedny z nejlepších opravných kódů. LDPC kódy jsou využity v několika

standardech a jejich další výhodou je že nejsou nijak patentově chráněny. Více informací ohledně LDPC kódu lze nalézt [2] a [3]

Blokové kódy jsou symetrické lineární kódy o pevně dané délce zakódované zprávy n a délce posloupnosti informačních bitů. K vytvoření blokových kódů je využita generující matice G.

$$G = \begin{bmatrix} I_k & P \end{bmatrix} \tag{3.1}$$

P - Paritní submatice o velikosti $k\,(n-k)$

I - Jednotková submatice $(k\,k)$

Tato matice slouží ke kódování, což je popsáno následujícím vztahem.

$$\mathbf{c} = \mathbf{m}\,G \tag{3.2}$$

$\mathbf{m}$ - Posloupnost informačních znaků.

$\mathbf{c}$ - Vektor kódové zprávy, který je výsledkem zakódování $\mathbf{m}$ do G.

Pro operaci kódování se využívá nejčastěji operace modulo dvě. Následně se pak definuje kontrolní matice H. Ta se uvádí v systematickém tvaru a tento tvar lze odvodit z G matice. Jedná se tedy o paritní matice.

$$H = \begin{bmatrix} I_{c-k} & P^T \end{bmatrix} \tag{3.3}$$

A tedy platí

$$G\,H^T = 0 \tag{3.4}$$

LDPC kódy jsou lineární blokové kódy, které obsahují matice G a H. Kódování je velmi obdobné jako u blokových kódů, a proto vychází ze zmíněného vztahu 1.1. Zakódované slovo se dostane násobením (modulo dvě) sloupců matice G s vektorem $\mathbf{m}$. LDPC kódy kladou největší význam na matici H, a proto zde musí platit.

$$c\,H^T = 0 \tag{3.5}$$

Jak již naznačuje název LDPC (Low Density Parity Check), tak kontrolní matice H má většinu prvků nulových. Následně podle rozdělení jedniček je kontrolní matice H dělena na rovnoměrné (Regular) LDPC kódy, anebo nerovnoměrné (Irregular) LDPC kódy. To naznačuje, že vytváření kontrolních matic H je klíčové, platí zde mnoho pravidel a je definováno mnoho přístupů, jak dosáhnout co nejlepších výsledků. Pro měření kvality LDPC kódu se využívá simulace chybovosti.

3.4 Tannerovy grafy

LDPC kódy celkově spadají do lineárních blokových kódů, které mohou být reprezentovány nejen maticově, ale i graficky. K zobrazení blokových kódů můžeme využít hned několik metod. Jednou z nejznámějších metod je Trellis diagram, na kterém je mimo jiné založen dekódovací systém. Jako další grafické zpracovaní, které je často využíváno, si můžeme uvést Tannerův graf, který je využívaný pro LDPC kódy. Tento graf byl poprvé publikovaný Michaelem Tannerem v roce 1981 a zobrazuje vztahy mezi kódovými bity a paritními bity. Tannerův graf tedy zobrazuje kontrolní matici H, která je rozdělena v Tannerově grafu na dvě sady uzlů. Variable uzly v_i, které jsou odvozeny od délky kódu n. Check uzly c_j, které jsou odvozeny od počtu řádků kontrolní matice. Následně jsou oba uzly převedeny do grafické podoby. Hrany jsou pak vytvořeny jen mezi jednotlivými skupinami uzlů a pouze v případech, kde je v příslušné pozici kontrolní matice hodnota rovna jedničce. Tannerovy grafy se vytváří proto, aby se lépe popsaly dekódovací algoritmy.

3.5 Distribuční funkce hodnosti uzlů

Iregulární LDPC kódy jsou definovány distribučními funkcemi hodností uzlů v Tannerově grafu. Neboli je to pravděpodobnost náhodně vybraného uzlu, že má spojení nebo sousední uzly. Nejedná se o konektivitu, jak je často špatně vyloženo, ale o počet hran či spojů, které vedou z daného uzlu do ostatních uzlů.

Nechť

$$\lambda(x) = \sum_{i=2}^{dl} \lambda_i \, x^{i-1} \tag{3.6}$$

a

$$\rho(x) = \sum_{i=2}^{dr} \rho_i \, x^{i-1} \tag{3.7}$$

Je distribuční funkce hodností pro variable a check uzly, respektive λ_i je poměr hran, který patří do stupňů-i variable uzlů. Proměnná ρ_j je obdobně jakoλ_i poměr hran, který ale patří do stupňů-j check uzlů, d_l vyjadřuje maximální variable stupeň a d_r je pak maximální check stupeň. Při použití těchto předpokladů je navrhovaná informační rychlost kódu dána rovnicí (3.8) (více informací [5]).

$$r = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \tag{3.8}$$

4 Density Evolution

Teď, když jsme si popsali základní informace ohledně LDPC kódování a dekódování, začneme věnovat analýze kódů. Jednou z analytických technik je density evolution, která je využívána k pochopení limitních vlastností LDPC kódů. Soubor LDPC kódů je definován pomocí paramterů ρ a λ, které reprezentují distribuční hodnoty v Tannerovu grafu. Density evolution se využívá pro zjištění opravných schopností kódu s danými distribučními funkcemi hodností ρ a λ, pokud se jeho délka blíží k nekonečnu. (Hledáme takový šum kanálu, pro který jsme schopni zaručit přenos informace.) Maximalní hodnotu šumu, při které je nejmenší pravděpodobnost chyb, budeme reprezentovat pomocí Thresholdu. Threshold je limita v souvislosti s LDPC kódy, pro kterou dekodér konverguje. Tím, že konverguje logaritmický průměr zpráv k nekonečnu, dekodér funguje spolehlivě. Pokud logaritmický průměr zpráv konverguje k určité hodnotě, vzniká nezanedbatelná chyba a dekodér funguje nespolehlivě. Jinak řečeno threshold je parametr kanálu, který určuje výkonnost kódu a pomáhá najít vhodné distribuční funkce hodností ρ a λ, s jejichž pomocí je možné navrhnout co nejlepší možný LDPC kód pro dané distribuční funkce hodností ρ a λ. To je možné pouze za předpokladu nekonečně dlouhého kódového slova. Cílem je co nejvíce se přiblížit k Shannonovu limitu. Density evolution může být naprogramováno několika způsoby. Tato práce se zaměřuje především na diskreditovanou density evolution a gaussovskou aproximaci pro density evolution.

4.1 Gaussova aproximace pro density evolution

4.1.1 Teorie gaussovské aproximace pro density evolution

V předchozí kapitole bylo již zmíněno, že u dlouhých LDPC kódů je možné za pomoci density evolution najít threshold. V této kapitole bude popsán jeden ze způsobů, jak vypočítat threshold LDPC kódu, který umožní zjistit odchylku od Shannonovi hranice. Distribuční funkce hodností jsou jedním z důležitých vlastností LDPC kódů. Právě ty slouží jako designové cíle pro návrh kódů. Následující kapitoly se soustředí především na binární AWGN kanál.

Dekódování je založeno na stromové architektuře, proto se v podgrafu neopakují žádné uzly a objem grafu je dostatečně velký, tudíž podgrafy mají stromovou strukturu. To znamená, že je možné analýzu provádět přímo, protože příchozí zprávy do každého uzlu jsou nezávislé, jak bylo dokázáno v článku [5]. Nicméně pokud je kód dostatečně dlouhý, je dokázáno, že pro každý náhodně vygenerovaný kód a pro skoro všechny vstupy budou vlastnosti dekodéru blízké vlastnostem dekodéru ve stromové struktuře s vysokou pravděpodobností.

Pro definici thresholdu uvažujeme následující příklad. Všechna nulová kódová

slova $\mathbf{c} = [00 ::: 0]$ byla odeslána. Pokud je použito klíčování fázovým posuvem znamená to, že všechna jedničková slova $\mathbf{x} = [+1 + 1.....1]$ budou přeposlána po kanálu. Chyba se vyskytne na dekodéru po dosažení maximálního množství iterací, pokud některé z proměnných uzlů LLRs zpráv od variable uzlů k check uzlům, budou mít záporné znaménko. Nechť $p_v^{(l)}$ je hustota pravděpodobnosti zprávy m_v, která je odeslána od variable node k check node během l-té interakce. V tomto případě nedojde k žádné chybě.

$$\lim_{l\to\infty} \int_{-\infty}^{0} p_v^{(l)}(\tau) d\tau = 0 \tag{4.1}$$

Je důležité si uvědomit, že $p_v^{(l)}(\tau)$ je závislé na parametru kanálu σ. Threshold $\sigma*$ je definován rovnicí (4.2).

$$\sigma* = \sup\{\sigma : \lim_{l\to\infty} \int_{-\infty}^{0} p_v^{(l)}(\tau) d\tau\} \tag{4.2}$$

To znamená, že pro různé hodnoty σ dostaneme různé výsledky density evolution. Díky tomu jsme schopni navrhnout vhodný kód.

Nechť je v logaritmický poměr pravděpodobnosti (Log-Likelihood Ratio (LLR)) zprávy od variable uzlu do check uzlů. V součtovém dekódování je v rovno součtu všech příchozích LLR zpráv.

$$v = \sum_{i=0}^{dv-1} u_i \tag{4.3}$$

Kde $u_i, (i = 1, \ldots, d_v - 1)$ jsou příchozí LLR zprávy od sousedních variable uzlů s výjimkou check uzlu, který dostane zprávu v, a u_0 je sledovaná LLR zprava (signál) výstupního bitu spojený s variable uzlem. Pravidlo pro aktualizování zpráv pro check uzly může být získáno ze sledování duality mezi variable a check uzly a výsledným vztahem Fourierovy transformace (více informací [6]). Navíc tímto způsobem zjistíme z procházející dekódované zprávy její pravidlo zpracování pro check uzly, které je ukázáno v rovnici (4.4).

$$u = 2\tanh^{-1}(\prod_{j=1}^{dc-1} \tanh(\frac{v_j}{2}) \tag{4.4}$$

Po následné úpravě je získána rovnice.

$$\tanh\frac{u}{2} = \prod_{j=1}^{dc-1} \tanh\frac{v_j}{2} \tag{4.5}$$

Kde $v_j, (j = 1, \ldots\ldots, d_c - 1)$ jsou přicházející LLR zprávy od $d_c - 1$ sousedů a stupně d_c check uzlu a u je výstup LLR zprávy odeslaný zbývajícímu (příslušnému) sousedovi.

4.1.2 Gaussova aproximace pro density evolution regulárních LDPC kódů

Pro LLR zprávu u_0 z kanálu AWGN. Tato zpráva je Gaussian se střední hodnotou $m = \frac{2}{\sigma_n^2}$, a s odchylkou $\sigma^2 = \frac{4}{\sigma_n^2}$ kde σ_n^2 je odchylka od šumu v kanálu. Distribuce zpráv může být popsána za použití pouze jednoho parametru m. Tím pádem pokud všechna $u_i, i \geq 1$ jsou nezávislá, identicky distribuovaná a jsou také Gaussian z rovnice (4.3), potom výsledný součet je také Gaussian, protože je to součet nezávislých náhodných gaussovských proměnných. I kdyby vstup nebyl Gaussian, tak podle centrálního limitního theorému by součet vypadal jako Gaussian (pro více informací [5]). Myšlenka gaussovké aproximace je projít skrz celou density evolution a sledovat střední hodnotu m zprávy, aby multidimenzionální výpočty mohly být převedeny na jednodimenzionální výpočet.

Nechť m_u případně m_v je značení pro střední průměrnou hodnotu u (v). Součet nezávislých náhodných gaussovských proměnných má střední hodnotu rovnou součtu předchozích průměrů. Takže m_v^l může být spočítáno jako násedující rovnice (4.6).

$$m_v^{(l)} = m_{u0} + (d_v - 1)m_u^{(l-1)} \tag{4.6}$$

Kde m_{u0} je střední hodnota u_0 a l označuje $l$tou interakci. Vynechali jsme index i, protože u_i jsou nezávislé a identicky distribuované pro $1 \leq i < d_v$ a mají stejný význam jako střední hodnota m_u. Dále je dáno, že $m_{u0} = \frac{2}{\sigma^2}$ a je to počáteční střední hodnota zprávy v_0. Střední hodnota $m_u^{(0)} = 0$ protože počáteční zpráva u_0 se rovná vždy nule.

Navíc m_u^l může být spočítáno pomocí pravděpodobnosti na obou stranách rovnice.

$$m_u^{(l)} = \phi^{-1}(1 - [1 - \phi(m_{u0} + (d_v - 1)m_u^{(l-1)})]^{dc-1}) \tag{4.7}$$

Definujeme pomocnou funkci pro $\phi(x)$.

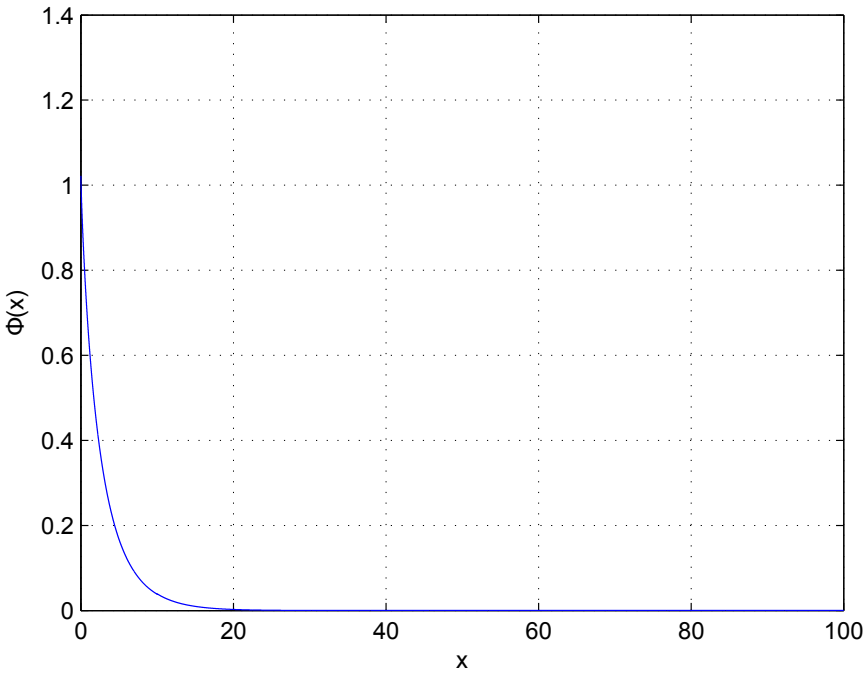
$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{\mathbb{R}} \tanh \frac{u}{2} \exp -\frac{(u-x)^2}{4x} du & \text{pokud } x > 0 \\ 1 & \text{pokud } x = 0 \end{cases} \tag{4.8}$$

Kde $\phi(x)$ je definováno jenom pro $x \geq 0$, protože uvažujeme, že byly přeneseny samé jedničky, proto jsou všechny střední hodnoty zpráv pozitivní. Může být dokázáno, že $\phi(x)$ je kontinuální a klesající pro $x \geq 0$ s $\lim_{x \to 0} \phi(x) = 1$ a $\lim_{x \to \infty} \phi(x) = 0$.

Bez výrazné změny v přesnosti výpočtů byla v pozdějších letech nalezena aproximace $\phi(x)$(4.9).

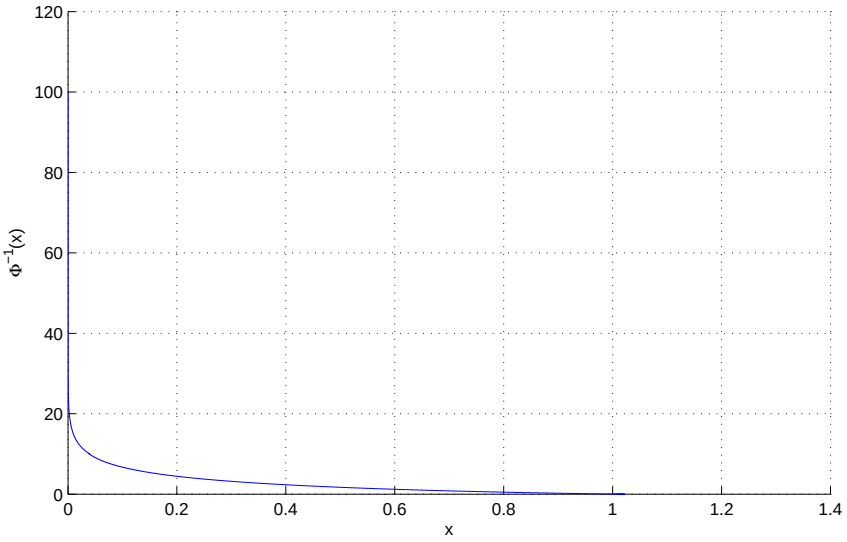
$$\phi(x) = \begin{cases} e^{-0{,}4527x^{0{,}86} + 0{,}0218} & \text{pokud } x < 10 \\ \sqrt{\frac{\pi}{x} e^{-\frac{x}{4}} \left(1 + \frac{1}{14x} - \frac{3}{2x}\right)} & \text{pokud } x \geq 10 \end{cases} \tag{4.9}$$

Tato aproximace je názorně ukázaná v grafu 4.1.



Obrázek 4.1: $\phi(x)$ aproximace zobrazená graficky

Bohužel aproximace $\phi(x)$ z rovnice (4.9) lze invertovat jenom z části a to pro $x < 10$. Proto je nutné invertování této aproximace vytvořit graficky jak je vidět na obrázku 4.2, anebo pomocí look up tabulky. V look up tabulce je možné nadále vyhledávat inverzní hodnoty pro aproximaci $\phi(x)$.



Obrázek 4.2: Inverzní $\phi(x)$ aproximce zobrazená graficky

4.1.3 Gaussova aproximace pro density evolution iregulárních LDPC kódů

Předchozí analýzu lze snadno rozšířit na iregulární LDPC kódy. Pro iregulární LDPC kódy je zapotřebí si definovat distribuční funkce hodností $\lambda(x)$ a $\rho(x)$, které byly již jednou definovány v kapitole 3.5. Od teď se bude uvažovat jen s náhodným nepravidelným souborem kódů, které jsou blíže specifikovány jako distribuční funkce hodností $\lambda(x)$ a $\rho(x)$.

$$m_u^{(l)} = \sum_{j=1}^{dc} \rho_j \phi^{-1}(1 - [1 - \sum_{i=1}^{dv} \lambda_i \phi(m_{u0} + (d_v - 1)m_u^{(l-1)})]^{j-1}) \tag{4.10}$$

Jak již bylo zmíněno, Gaussova Aproximace je aproximační algoritmus, který předpokládá že pravděpodobnostní funkce mají přibližně tvar Gaussovy křivky. Z toho vyplívá, že je možné velmi rychle vypočítat threshold bez jakéhokoliv zhoršení přesnosti výpočtů. Je ale nutné zdůraznit, že tento algoritmus je možné aplikovat pouze pro AWGN kanály.

4.2 Diskretizovaná density evolution

4.2.1 Teorie Diskretizované density evolution

Diskretizovaná density evolution je druhým známým způsobem výpočtu analýzy density evolution. Obdobně jako u gaussovské aproximace si i zde musíme definovat LLR zprávy, které byly již jednou definovány v kapitole 4.1.1. Tato kapitola především čerpá z materiálů [6] a [7]. Dále je zapotřebí si definovat distribuční funkce hodností $\lambda(x)$ a $\rho(x)$ (více informací v kapitole 4.1.3).

V tomto algoritmu pro density evolution se využívá funkce $Q(z)$, neboli kvantizované zprávy. Nechť je tedy funkce $Q(z)$ definována jako:

$$Q(w) = \begin{cases} \left\lfloor \frac{z}{\Delta} + \frac{1}{2} \right\rfloor \Delta & \text{pokud } z \geq \frac{\Delta}{2} \\ \left\lceil \frac{z}{\Delta} - \frac{1}{2} \right\rceil \Delta & \text{pokud } z \leq -\frac{\Delta}{2} \\ 0 & \text{jindy} \end{cases} \tag{4.11}$$

Kde Q je kvantizační operátor a z je zpráva, kterou chceme zkvantizovat. Proměnná Δ je kvantizační interval, $\lfloor x \rfloor$ je větší celé číslo příslušné x a $\lceil x \rceil$ je celé číslo, které není větší než x.

LLR zprávy a jejich pravděpodobnosti se stanou diskrétními poté, co jsou přepočítány touto kvantizační funkcí. Po kvantizaci se z rovnice (4.3) stane $\bar{v} = \sum_{i=0}^{dv-1} \bar{u}_i$, kde $\bar{v} = Q(v)$ a $\bar{u}_i = Q(u)$ pro $i = 0, \ldots, d_v - 1$. Následně je zapotřebí definovat pravděpodobnostní funkci kvantizační zprávy jako $\bar{z}$ tak, že $p_z[k] = \text{Pravděpodobnost}(\bar{z} = k\Delta)$ pro všechny $k \in \mathbb{Z}$. To znamená, že p_v souvisí s vstupní pravděpodobnostní funkcí podle vzorce.

$$p_v = \bigotimes_{i=0}^{dv-1} p_{ui} \tag{4.12}$$

Kde p_v je pravděpodobnostní funkce $\bar{v}$. Obdobně je tomu s p_{ui}, které přísluší pravděpodobnostní funkci u_i a $\bigotimes$ je diskrétní konvoluce. Protože je $\bar{u}_i$ nezávisle a identicky distribuováno můžeme rovnici (4.12) psát jako:

$$p_v = p_{u0} * \left(\bigotimes^{dv-1} p_u \right) \tag{4.13}$$

Kde $p_u = p_{ui}$, i náleží hodnotám od $1 \leq i < d_v$ a $*$ je diskrétní konvoluce. Tento výpočet může být vypočítán efektivněji, pokud se použije rychlá Fourierova transformace.

Dále je zapotřebí definovat operátor R, který spočítáme následovně:

$$R(x,\, y) = Q\left(2 \tanh^{-1}\left(\tanh\frac{x}{2} \tanh\frac{y}{2}\right)\right) \tag{4.14}$$

Kde x i y jsou kvantizované zprávy. Tato operace může být dále zefektivněna tím, že se pro operátor R před počítá tabulka, která se bude následně využívat pro ostatní funkce. Její použití je vysvětleno v kapitole 5.3. Za použití této operace můžeme dále spočítat kvantizované zprávy $\bar{u}$ z rovnice (4.5) následovně:

$$\bar{u} = R\left(\bar{v}_1,\, R\left(\bar{v}_2,\, \ldots,\, R\left(\bar{v}_{dc-2},\, \bar{v}_{dc-1}\right)\ldots\right)\right) \tag{4.15}$$

Nechť $c = R\left(x, y\right)$. Pravděpodobnostní funkci p_c z c vypočítáme následovně:

$$p_c\left[k\right] = \sum_{(i,\,j):k\Delta = R(i\Delta, j\Delta)} p_x[i]\, p_y[j] \tag{4.16}$$

Což můžeme také zapsat jako $p_c = R\left(p_a, p_b\right)$.

Protože p_{vi} jsou nezávislé a identicky distribuované, můžeme je definovat jako $p_v = p_{vi}$ pro $1 \leq i < d_c$. Pak je možné napsat, že $p_u = R\left(p_v,\, R\left(p_v,\, \ldots,\, R\left(p_v,\, p_v\right)\ldots\right)\right)$ je stejné jako $p_u = R^{dc-1} p_v$.

Pro iregulární LDPC kódy s distribuční funkcí hodností λ a ρ můžou dále napsat tyto dvě rovnice:

$$\lambda(p) = \sum_{i=2}^{dl} \lambda_i \bigotimes^{i-1} p \tag{4.17}$$

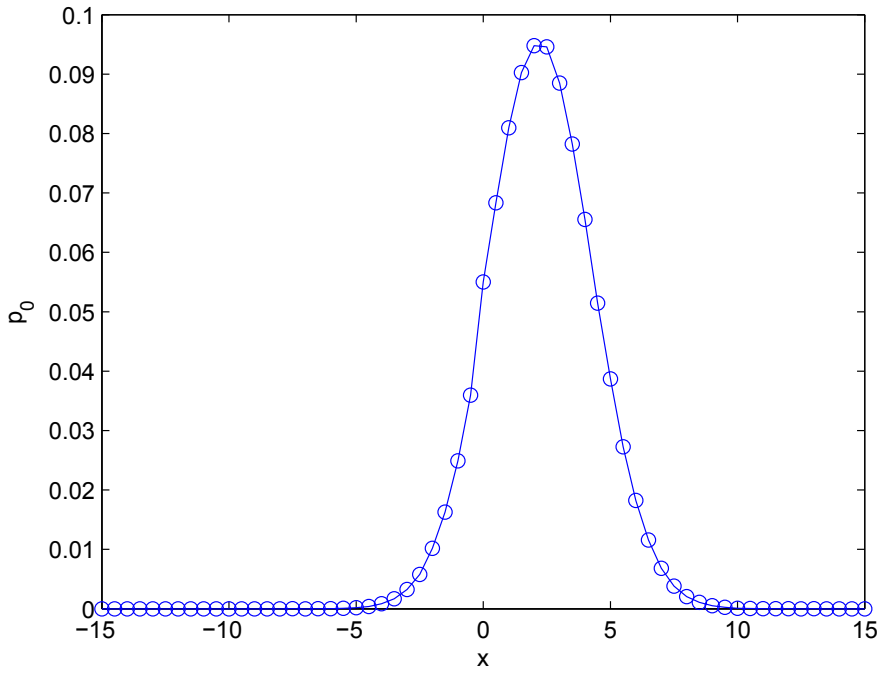
a

$$\rho(p) = \sum_{j=2}^{dr} \rho_j R^{j-1} p \tag{4.18}$$

Tyto operátory jsou definovány pro jakékoliv pravděpodobnostní funkce p. Následně můžeme definovat diskretizovanou density evolution pro iregulární LDPC kódy a spočítat jí jako.

$$p_v^l = p_0 * \lambda\left(\rho\left(p_v^{l-1}\right)\right) \tag{4.19}$$

Kde p_0 je počáteční pravděpodobností funkce sledovaného kanálu zobrazena v grafu 4.3. Pravděpodobnostní vektor p_0 je jinak řečeno pmf normalizované gaussovské funkce. Proměnná l označuje l-tou interakci.



Obrázek 4.3: Výsledná p_0.

5 Programová realizace

5.1 Zjištění threshold pro density evolution za pomocí Gaussovy Aproximace pro AWGN kanál

Pro zjištění thresholdu byl napsán program v Matlabu, který je často využíván pro vědecko-technické a inženýrské výpočty. Základní program `Threshold_funkce_for_Gaussian.m` může být zahrnut do dalšího programu a dále využit jako funkce. Tato funkce má dva vstupní parametry, `Ro` a `Lamda`, které definují distribuční funkce hodností λ a ρ. `Threshold_funkce_for_Gaussian.m` využívá metodu půlení intervalů pro urychlení výpočtů. Ke správnému nastavení funkce je využita proměnná `citlivost`, která funguje pro nastavení zastavovacího argumentu. Tento program dále zahrnuje několik podfunkcí:

- `Fi_funkce.m` - Vstupní argument je `X` a výstupní je `Fi`. Tato funkce odpovídá rovnici (4.9).

- `Inverzní_Fi_funkce_debug.m` - Tato funkce vypočítává inverzní `Fi` (značené `Fi_inv`) za pomoci vyhledávací tabulky. V tomto souboru je zahrnuto několik režimů, které byly velmi užitečné při vývoji, a mohly by se využít i v budoucím zdokonalování funkce. Režimy se mění pomocí proměnné `cinnost`. Pokud se proměnná `cinnost` rovná 0, pak se vytvoří vyhledávací tabulka a uloží se do souboru `Lookup_table.mat` i s ostatními důležitými proměnnými. Pokud se proměnná `cinnost=1`, je nastaven režim vyhledávání v matici. V první řadě se načte soubor `Lookup_table.mat` a následně se v něm najde hodnota `Fi_inv` k příslušné hodnotě `X`. Další možností je nastavit `cinnost=2`. Tato možnost slouží pro kontrolu inverzního `Fi_inv`. Po využití toho režimu dostaneme stejný výsledek jako z předešlé funkce `Fi_funkce.m` s tím rozdílem, že tato funkce využívá k dohledání výsledku soubor `Lookup_table.mat`. Pokud se proměnná `cinnost` rovná hodnotě 3, vykreslí se `Fi_inv` do grafu.

- `Inverzní_Fi_funkce_inic.m` - zastává stejnou funkci jako část již popsané funkce `Inverzní_Fi_funkce_debug.m` v nastaveném režimu `cinnost`=1. Jinak řečeno vytvoří tabulku, z které jsme pak schopni přečíst `Fi_inv` a uloží ji s ostatními důležitými proměnnými do `Lookup_table.mat`.

- `Inverzní_Fi_funkce.m` - Tato funkce načte soubor `Lookup_table.mat`, ve kterém následně vyhledá `Fi_inv` příslušnou k požadované hodnotě `X`.

- `Mean_of_variable_iregular.m` - využívá předešlých dvou funkcí a jejími vstupními proměnnými jsou vektory `Ro`, `Lamda`, `sigma` a hodnota `pocet_iteraci`. Výstupní proměnnou je `mu`, která je definována jako střední hodnota předávaných zpráv a vypočítá se podle rovnice (4.10).

5.2 Ukázka praktického nasazení kódu

Ukázka použití `Threshold_funkce.m`. Zde bude popsáno jak vygenerovat threshold s využitím naprogramovaných funkcí. Funkce pro vyhledání thresholdu se zavolá v Matlabu příkazem `[Threshold]=Threshold_funkce_for_Gaussian(Ro, Lamda, pocet_iteraci)`. Kde `Ro`, `Lamda` a `pocet_iteraci` jsou vstupní proměnné. Pro tento příklad byl použit regulární kód.

```
Lamda=[0.0, 0.0, 1];
Ro=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0];
pocet_iteraci=500;
```

Proměnné, mezi kterými se bude hledat Threshold, se nacházejí mezi `sigma_1` a `sigma_2`, kde `sigma_1=0` a `sigma_2=2`. Z důvodu urychlení programu jsme využili půlení intervalů. Pomocí půlení intervalů je vypočítána testovaná proměnná sigma. V této funkci je tedy nejdříve vypočítána střední průměrná zpráva, a pak se kontroluje, jestli konverguje k nekonečnu, nebo ne. To je ovlivněno i `poctem_iteraci=500`, které určují, kolik iterací bude mít střední průměrná zpráva, a tím je dána přesnost výsledku. `Threshold_funkce_for_Gaussian` je hlavní funkcí, která využívá několik podfunkcí. Threshold je v této hlavní funkci výstupní proměnná, která je v našem příkladu rovna:

```
Threshold= 0.87466
```

Ukázka `Inverzní_Fi_funkce_inic.m`. Tato funkce je poprvé volána hned na začátku programu. V tomto případě tato funkce nemá žádné výstupní ani vstupní proměnné. Jinak řečeno vytvoří vyhledávací tabulku, pomocí které se zjistí hodnota `Fi_inv`. Následně ji uloží s dalšími důležitými proměnnými do souboru s názvem `Lookup_table.mat`.

Ukázka použití `Mean_of_variable_iregular.m`. Střední průměr zprávy se vypočítá zavoláním funkce `mu = Mean_of_variable_iregular(Ro, Lamda, sigma, pocet_iteraci)` a je vypočítána podle rovnice (4.10). `Ro`, `Lamda`, `sigma` a `pocet_iteraci` jsou stejné proměnné, jako v příkladu kdy byla volána funkce `Threshold_funkce_for_Gaussian`. Proměnná `sigma` je parametr kvality kanálu, který se právě testuje a vypočítává se půlením intervalů ve funkci `Threshold_funkce_for_Gaussian`. Po rozpůlení intervalů v tomto případě je jako první testována hodnota 1. Velikost vektoru `mu` je vždy roven proměnné `pocet_iteraci`. Po zavolání a výpočtu funkce se vektor středního průměru zprávy rovná (zkráceno)

```
mu=[0, 0.12120, 0.16110, 0.17610, 0.18200, 0.18440, 0.18530, ...]
```

Ukázka použití `Mean_of_variable.m`. Tato funkce je velmi obdobná funkci Mean_of_variable_iregular ale je vytvořena jen pro regulární kódy podle rovnice (4.7). Zavolá se příkazem `[mu]=Mean_of_variable(Ro, Lamda, sigma, pocet_iteraci)` a je vypočítána podle rovnice (4.7). Pro tento příklad jsme použili vstupní hodnoty:

```
Lamda=[0.0, 0.0, 1];
Ro=[0.0, 0.0, 0.0, 0.0, 0.0, 1.0];
sigma=1;
pocet_iteraci=200;
```

Po zavolání funkce dostaneme střední průměrnou hodnotu zprávy `mu`.

```
mu=[0, 0.12120, 0.16110, 0.17610, 0.18200, 0.18440, 0.18530, ...]
```

Ukázka použití `Fi_funkce`. Tato funkce reprezentuje rovnici, která je podrobněji rozebrána v teorii 4.9, proto si zde jenom nastíníme, jak tuto funkci používat. Jako první musíme funkci zavolat a to následovně `Fi=Fi_funkce(X)`, která je součástí rovnice (4.10). V našem případě je vstupní hodnotou do této funkce `X`=2 a pro tuto hodnotu je pak výstupní hodnota.

```
Fi=0.4494
```

Ukázka `Inverzní_Fi_funkce.m`. Tato funkce má jednu vstupní proměnnou `X` a jednu výstupní proměnnou `Fi_inv`. Po zavolání funkce se načte soubor `Lookup_table.mat`, který obsahuje vyhledávací tabulku a další proměnné, díky kterým jsme schopni najít pro příslušné `X` inverzní hodnotu `Fi`. Například pokud vstupní `X` se rovná hodnotě 0.4494, potom výstupní `Fi_inv` je rovno 2.

5.3 Zjištění threshold pro density evolution za pomoci diskretizace pro AWGN kanál

Stejně jako v předchozí kapitole, tak i v této, byl program napsán v programovém prostředí Matlabu. Hlavní funkcí se v tomto případě nazývá `Threshold_funkce_for_Discretized`. Tato funkce může být opět zahrnuta do rozsáhlejšího programu. Vstupními hodnotami pro hlavní funkci jsou `Ro` a `Lamda`, které určují distribuční funkce hodností kódu. Výstupní hodnotou je `Threshold`, který je určen na základě výpočtů ostatních podfunkcí. Vysvětlení tohoto programu lze nalézt v kapitolách 4.2.1 a 6.3.

- `Quantized_function.m` - Jedná se o kvantizační funkci 4.11. Tato funkce využívá dvě vstupní proměnné `w` a `Delta`, kde `Delta` je kvantizační parametr. Výstupem je `Q_x` kvantizovaná proměnná.

- `p0_function.m` - Funkce má vstupní proměnné `Sigma` a `delku` a výstupní proměnné jsou `Delta`, `p0` a `zero`. Proměnná `p0` se vypočítá z gaussovského rozložení, jak je uvedeno v kapitole 4.2.1 , `delku` pak určuje kolik náhodných hodnot bude vygenerováno přes funkci `randn()`. Proměnná `zero` určuje, kde se nachází 0 na ose x a `Delta` se vypočítá podle rovnice (6.1).

- `R_function_inic.m` - Vstupními proměnnými jsou `Delta` a `delka`. Tato funkce nemá žádný výstup, protože vytvoří tabulku, která je následně uložena jako `R_function_table.mat`. Tato tabulka obsahuje matici `R`, která je vypočítána pomocí rovnice (4.14).

- `R_function.m` - Tato funkce využívá matici `R` ze souboru `R_function_table.mat`. Po načtení rovnice se vypočítají dva výstupní vektory `i_vektor` a `j_vektor`. Vstupními argumenty do této funkce je `Delta` a `k`. Více informací je popsáno v teorii 4.2.1.

- `pc_function.m` - Funkce má tři vstupní argumenty, kterými jsou dva vektory `pa`, `pb` a `Delta`. Výstupem je pak vektor o stejné délce `pc`. Rovnice, která se používá na výpočet je více popsána v teorii 4.16.

- `Ro_function.m` - Je funkce, která přepočítá pravděpodobnostní funkci a přidá k němu parametr `Ro`, který určuje jednu z distribučních funkcí hodností kódu. Vstupními argumenty jsou `Ro`, `pmf` a `Delta`. Tento výpočet je proveden podle rovnice (4.18) a tato funkce má jako výstup vektor `Ro_p`.

- `Lamda_function.m` - V této funkci přepočítáváme pravděpodobnostní funkci a zaneseme do ní druhý parametr distribuční funkce hodnosti `Lamda` jak je uvedeno v teorii 4.17, kde vstupními proměnnými je `Lamda`, `pmf` a `zero`. Výstupem je pak `Lamda_p` a `zero_out`. `zero_out` určuje nulu na ose x.

- `Discretized_density_evolution.m` - Tato funkce má celou řadu vstupních proměnných `Lamda`, `Ro`, `Sigma` a `počet_iteraci`. Jejím úkolem je spojit dohromady všechny předchozí funkce a to podle rovnice (4.19). Jejím výstupem je pak spočítaná matice `pv` a

`zero_sub`. Proměnná `zero_sub` určuje nulu na ose x, podle které se můžeme orientovat.

- `Threshold_funkce_for_Discretized.m` - Tato funkce je obdobná jako u gaussovské aproximace. Opět pomocí půlení intervalů vypočítme `Sigma`, která se nachází na práhu a je tedy rovna thresholdu. LDPC kód je určen dvěma vstupními parametry `Ro` a `Lamda`. Poslední vstupním parametrem je `počet_iteraci`. Výstupem je `Threshold`.

5.4 Ukázka praktického nasazení kódu:

Ukázka použití `Threshold_funkce_for_Discretized.m`. Tato funkce se zavolá příkazem function `[Threshold]=Threshold_funkce(Ro, Lamda, pocet_iteraci)`. Kde `Ro`, `Lamda` a `pocet_iteraci` jsou vstupní proměnné. V tomto příkladu budou rovny:

```
Lamda= [0.0, 0.15, 0.55, 0.3];
Ro= [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0];
pocet_iteraci= 10;
```

Proměnné, mezi kterými se bude hledat `Threshold`, se nacházejí mezi `Sigma_1` a `Sigma_2`, kde `Sigma_1=0` a `Sigma_2=2`, jak již bylo zmíněno. Z důvodu urychlení programu jsme využili půlení intervalů. Pomocí půlení intervalů je pak vypočítáno testovaná proměnná `Sigma_test`, která se dále přepočítá v několika iterací. Každá iterace pak má rozdílnou pozici svého maxima `pv` na ose x. Tato hodnota se buď pohybuje doprava anebo nikoliv. Následně se podle toho rozhodne další interval, pro který se bude hodnota přepočítávat. (Celý tento proces je popsán v kapitole 6.3). Funkce `Threshold_funkce_for_Discretized.m` je hlavní funkcí, která využívá několik podfunkcí. `Threshold` je v této hlavní funkci výstupní proměnná, která je v našem příkladu rovna.

```
Threshold= 0.8651
```

Ukázka použití `Discretized_density_evolution.m`. Funkce se zavolá pomocí příkazu `[pv,zero_sub]=Discretized_density_evolution(Lamda,Ro,Sigma,pocet_iteraci)`. `Lamda`, `Ro`, `Sigma` a `pocet_iteraci` jsou vstupní proměnné. Pro tento příklad byly použity stejné proměnné jako v příkladu funkce `Threshold_funkce_for_Discretized.m`.

```
Sigma=0.85
```

Výstupem je matice diskretizované density evolution `pv` a `zero_sub`. Proměnná `zero_sub` určuje pozici nuly vektoru pv v l-té iteraci na ose x. Tato pozice je velmi důležitá, aby mohly být porovnány jednotlivé hodnoty a jejich posun na ose x ve funkci `Threshold_funkce_for_Discretized.m`. Každá hodnota ve vektoru odpovídá dané iteraci. U matice `pv` každý řádkový vektor odpovídá dané iteraci (zkráceno).

$$pv = \begin{bmatrix} ..., & 0.0507, & 0.0503, & 0.0496, & 0.048, & ...; \\ ..., & 4.3686e-48, & 1.4863e-45, & 1.8782e-43, & 1.1742e-41, & ...; \\ ..., & 0, & 0, & 0, & 0, & ...; \\ ..., & ..., & ..., & ..., & ..., & ... \end{bmatrix}$$

```
zero_sub= [51, 151, 334, 334, 334, 334, 334, ...]
```

Ukázka `Quantized_function.m`. Tato funkce se volá `[Q_z]=Quantized_function(z, Delta)`. Kde `w` je hodnota, kterou bude zkvantizována a `Delta` je kvantizační parametr.

```
z= 0.256;
Delta= 0.5;
```

Výstupní hodnotou je zkvantizovaná hodnota `Q_z`.

```
Q_z= 0.5
```

Ukázka `p0_function.m`. Funkce slouží k vytvoření pravděpodobnostní funkce pro zprávy v první iteraci. Funkce se volá pomocí příkazu `[p0,Delta,zero]=p0_function(Sigma,delka)`, kde jak již bylo řečeno, jsou dva vstupní parametry:

```
Sigma= 0.85;
delka= 100000;
```

Parametr `delka` určuje, kolik hodnot bude mít vytvořené Gaussovo rozdělení hodnot. Následně je gaussovské rozdělení zkvantizováno a přepočítáno na pravděpodobnostní funkci. Výstupní proměnné jsou `Delta`, `zero` a vektor `p0`. Proměnná `zero` určuje pozici nuly na ose x (zkráceno):

```
p0= [..., 0.00130, 0.0019, 0.0025, 0.0034, 0.0046, 0.0060, ...]
Delta= 0.3
Zero= 51
```

Ukázka `Ro_function.m`. Tato funkce přepočítává, obdobně jako Lamda funkce, hodnoty pmf a přidává k ní parametr kanálu. Funkce se zavolá příkazem `Ro_p= Ro_function(Ro, pmf, Delta)`, kde vstupní vektor Ro je stejný jako ve funkci `Threshold_funkce_for_Discretized.m`. Pmf pak odpovídá pravděpodobnostnímu vektoru, který je přepočítán, a delta je stejná jako na výstupu funkce `p0_function.m`. Předpokládejme, že jsme použili totožný vektor s výstupem předchozí funkce a tedy `pmf=p0`.

Tato funkce využívá několik podfunkcí, které budou vypsány níže. Její výstupní hodnotou je `Ro_p` (zkráceno).

```
Ro_p= [..., 2.5447e-05, 7.2725e-05, 0.0002, 0.0005, 0.0011, ...]
```

Ukázka `Lamda_function.m`. Funkce se zavolá pomocí příkazu `[Lamda_p,zero_out]= Lamda_function(Lamda,pmf,zero)`, kde `Lamda` je stejná jako v případě `Threshold_funkce_for_Discretized.m`, `pmf` je pravděpodobnostní vektor, který přepočítáváme a `zero` je pozice nuly na ose x. Pro tento příklad předpokládejme, že byl využit výsledek z předchozí funkce `pmf=Ro_p`

V této funkci je důležité přepočítat pozici nuly na ose x, protože po provedení konvoluce se zvětší pravděpodobnostní vektor o (velikost vektoru A + velikost vektoru B) − 1. Výstupem je pak přepočítaná pravděpodobnostní funkce `Lamda_p` a pozice nuly na ose x daná proměnnou `zero_out`.

```
Lamda_p= [..., 0.0002, 0.0004, 0.0008, 0.0016, 0.0030, 0.0054, ...]
zero_out=451
```

Ukázka `pc_function.m`. Tato funkce využívá podfunkci `R_function.m`. Zavolá se pomocí příkazu `pc=pc_function(pa, pb, Delta)`, kde `pa` a `pb` jsou pravděpodobnostní vektory (zkraceno). `Delta` je stejná jako na výstupu funkce `p0_function.m`.

```
pa= [..., 0.0013, 0.0018, 0.0025, 0.0034, 0.0045 0.0060, 0.0076, ...]
pb= [..., 0.0013, 0.0018, 0.0025, 0.0034, 0.0045 0.0060, 0.0076, ...]
```

Tyto vektory jsou pak mezi sebou násobeny podle rovnice (4.16). Výstupem je vektor pc (zkráceno).

```
pc= [..., 0.0008, 0.0014, 0.0022, 0.0034, 0.0051, 0.0075, 0.0114, ...]
```

Ukázka `R_function_inic.m`. Tato funkce se volá v každé iteraci programu `Discretized_density_evolution.m`. Zavolá se příkazem `R_function_inic(Delta,delka)`. V tomto případě tato funkce nemá žádné výstupní proměnné ale má dvě vstupní proměnné. Jinak řečeno, vytvoří vyhledávací tabulku z rovnice (4.14). Tato tabulka je vytvořena, aby se nemusely opakovaně vypočítávat a hledat jednotlivé hodnoty. Následně se tato vypočítaná matice uloží do `R_function_table.mat`.

Ukázka použití `R_function.m`. Funkce využívá předem vypočítané matice `R_function_table.mat`, která byla vygenerována příkazem `R_function_inic(0.3,101)`. Tuto funkci zavoláme příkazem `[i_vector,j_vector]=R_function(Delta, k)`. Vstupními proměnnými jsou `k` a `delta`, která je stejná jako na výstupu funkce `p0_function.m`.

```
k= 3
```

Po spuštění programu se načte matice `R` a vyhledají se všechny hodnoty, které se rovnají `k`* `Delta`. Výstupem jsou dva vektory, které odpovídají pozicím nalezených hodnot.

```
i_vector= [1, 101]
j_vector= [101, 1]
```

6 Výsledky simulace

V této kapitole jsou postupně popsány výsledky jednotlivých analýz pro density evolution s gaussovskou aproximací a diskretizovanou density evolution. V první části jsou vždy popsány výsledky pro regulární kódy. Zbytek kapitoly je zaměřen na iregulární kódy. V této kapitole jsou uvedeny i další důležité parametry jako rychlost výpočtu. Je zapotřebí si uvědomit, že programovací prostředí Matlab nemá ve vektoru nulový index, ale až první. To způsobilo komplikace při programování a rozdílnosti této práce oproti dostupným literaturám.

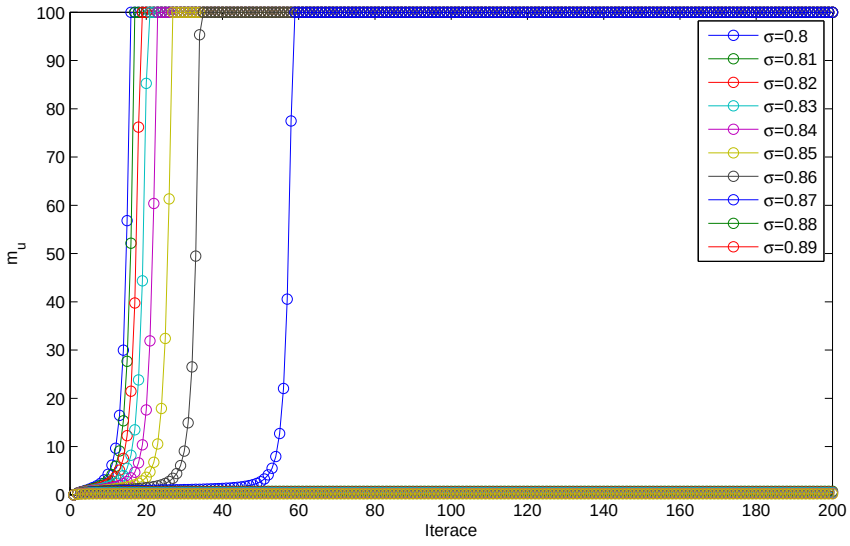
6.1 Gaussova aproximace pro regulární LDPC kódy

První část je zaměřena na regulární kódy. V grafu (6.1) jsou vidět jednotlivé hodnoty m_u pro různé hodnoty σ. Pro regulární kód jsou definovány distribuční funkce hodností λ, ρ a pro tento příklad se použilo 200 iterací.

$\lambda = 1x^2$

$\rho = 1x^5$

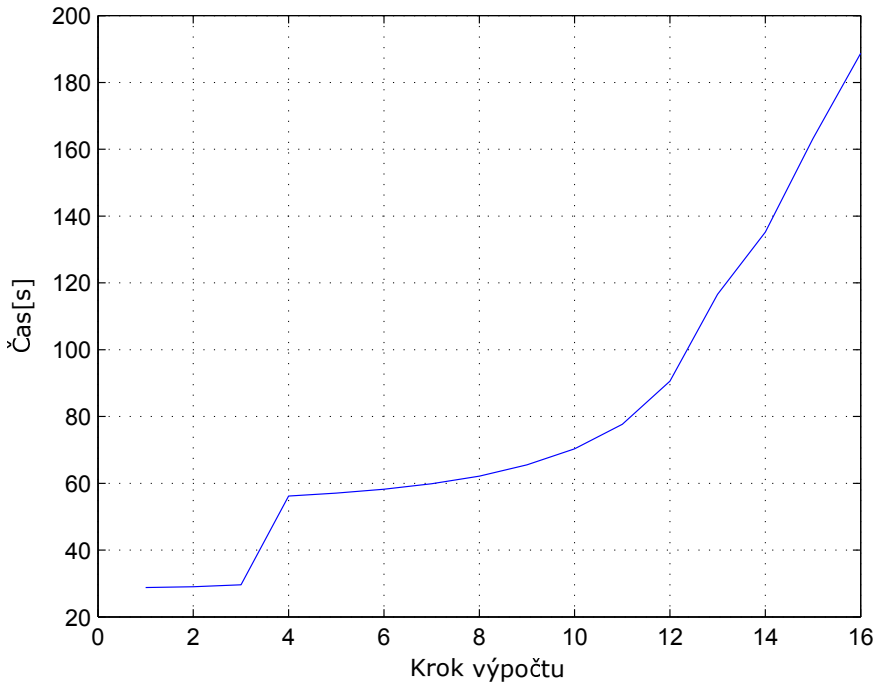
Z grafu 6.1 je možné usoudit, že threshold se bude nacházet kolem hodnoty $\sigma = 0,87$, jelikož v tomto bodě přestane m_u konvergovat s rostoucím počtem iterací k nekonečnu. Postupné výpočty mezi $\sigma = 0,8$ a $\sigma = 0,9$ byly zjemněny, aby byl lépe vidět threshold.



Obrázek 6.1: Výpočet m_u s rozdílnou σ pro regulární kód.

Pro zjištění přesnější hodnoty tresholdu jsme využili funkci `Threshold_funkce_for_Gaussian.m`. Více o této funkci najdeme v kapitole 5.1. Tato funkce využívá půlení intervalů pro rychlejší konvergenci algoritmu. Po dokončení

výpočtů byl obdržen výsledek thresholdu 0.87466. Tento výsledek je dosažen s nastavenou hodnotou *citlivost* $= 0,0001$, která určuje zastavovací podmínku pro půlení intervalů. Výsledek byl dále spočítán v 16-ti krocích po uplynutí 3 min 8,783 s. Následující graf 6.2 ukazuje časový průběh výpočtů m_u v jednotlivých krocích půlení intervalů s rozdílnou hodnotou σ.



Obrázek 6.2: Časový graf výpočtů thresholdu pro regulární kód.

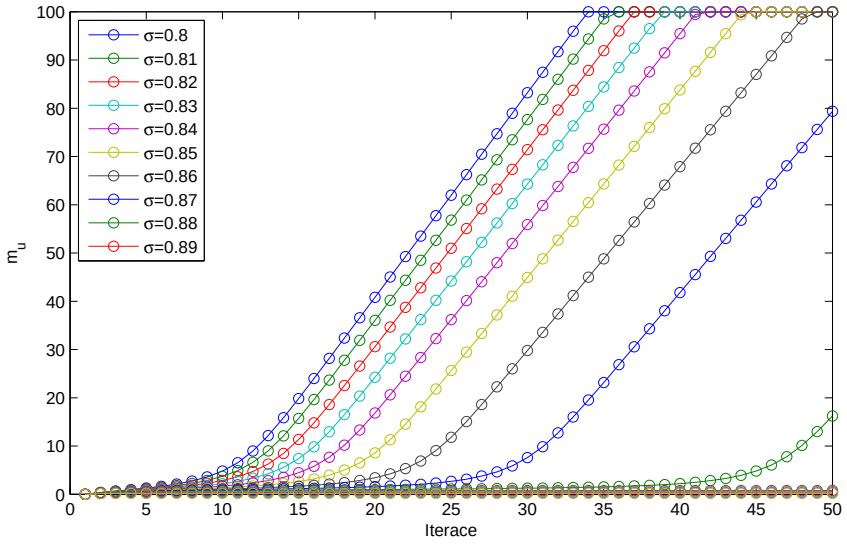
6.2 Gaussova aproximace pro iregulární LDPC kódy

Tato část se zaměřuje na iregulární kódy. Obdobně jako v první části si zde uvedeme graf 6.3 zobrazující jednotlivé hodnoty m_u v závisloti na iteraci a σ. Pro tento graf je počet iterací roven 50 a byly použity distribuční funkce hodností λ a ρ, které jsou rovny:

$\lambda = 0,15 + 0,55x + 0,3x^2$

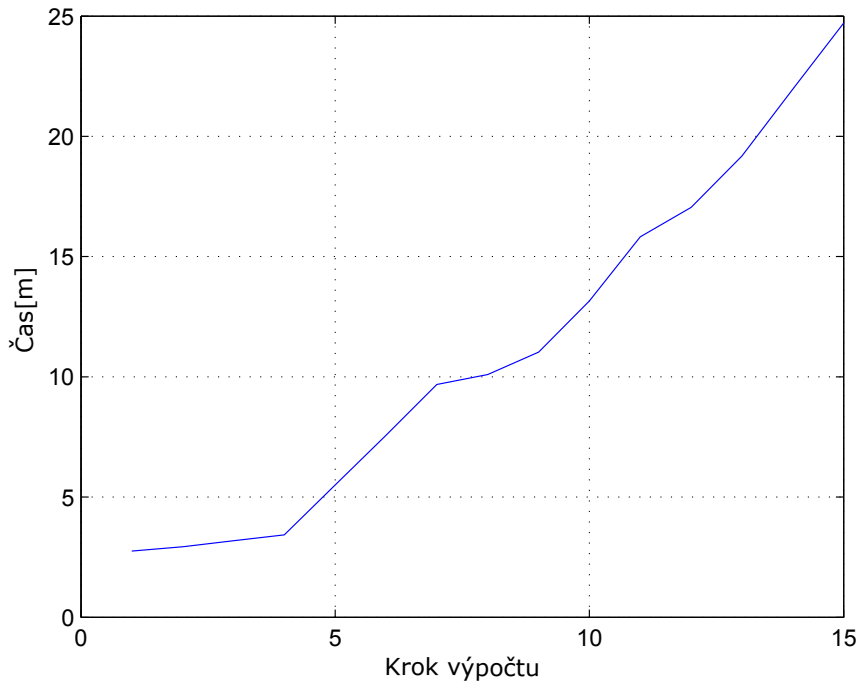
$\rho = 1x^5$

V tomto grafu lze pozorovat, že se threshold posunul na hodnotu přibližně $\sigma = 0,88$. Obdobně jako v předchozí kapitole 6.1 pro nižší hodnoty σ, které jsou pod prahem thresholdu, výpočet m_u konverguje k nekonečnu s rostoucím počtem iterací a pro vyšší hodnoty σ hodnoty m_u konvergují k dané hodnotě. Hodnoty σ byly opět zjemněny mezi 0,8 a 0,9 pro dosažení více průběhů m_u okolo thresholdu.



Obrázek 6.3: Vypočet m_u s rozdílnou σ pro regulární kód.

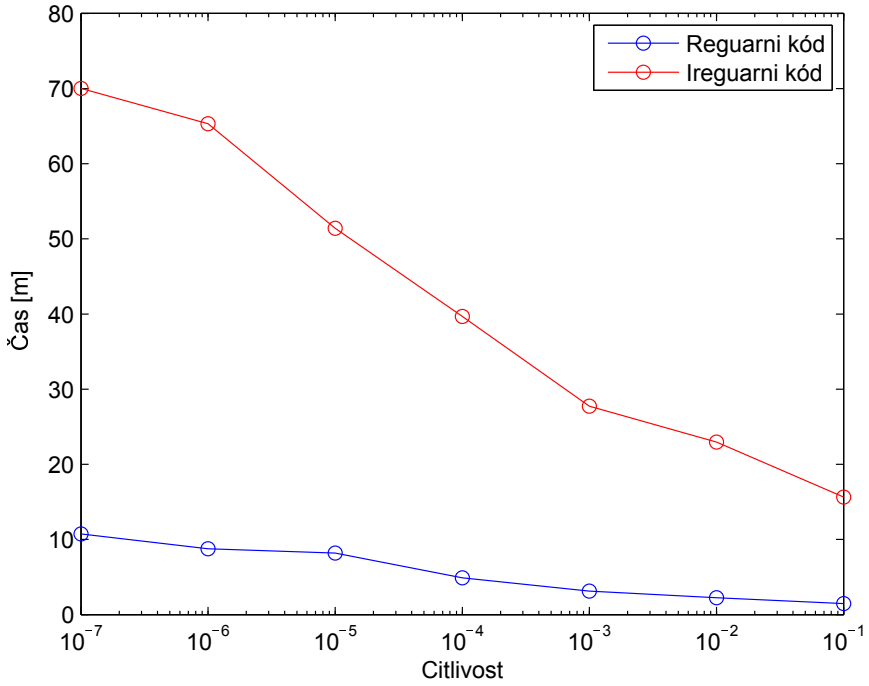
Následně byla zjištěna hodnota thresholdu s přesností $citlivost = 0,0001$. Hodnota thresholdu vyšla 0,8875. Výpočet tohoto thresholdu trval 24 min 42,646 s jak je možné pozorovat v grafu 6.4 a byl vypočítán v 15-ti krocích.



Obrázek 6.4: Časový graf výpočtů thresholdu pro regulární kód.

V grafu 6.5 je vidět porovnání výpočtů thresholdu pro regulárního kód počítaného podle rovnice (4.7) a iregulárního kódu počítaného podle rovnice (4.10). Tyto výpočty

byly provedeny pro *citlivost* od 0,1 do 0,00001 pro stejné distribuční funkce hodností jako v kapitolách 6.1 a 6.2.



Obrázek 6.5: Porovnání časové náročnosti výpočtů pro různé citlivosti.

Výsledky grafu 6.5 jsou dále uvedeny v tabulce 6.1 pro regulární kód (počítaný podle rovnice (4.7)) a v tabulce 6.3 pro iregulární kód (počítaný podle rovnice (4.10)). Tabulka 6.2 ukazuje výsledky pro regulární kód, který je definován distribuční funkcí hodností z kapitoly 6.1 a je počítaný podle rovnice (4.10). V tabulkách jsou zobrazeny výsledky thresholdu pro výpočty s různou citlivostí a čas jednotlivých výpočtů. Z těchto výsledků je zřejmé, že výpočet thresholdu pro iregulární gaussovskou aproximaci podle rovnice (4.10) je časově náročnější, i když použijeme regulární kód.

Citlivost	Threshold	Čas výpočtu
0,1	0,85937500	1 min 28,758 s
0,01	0,87451171	2 min 14,838 s
0,001	0,87463378	3 min 8,256 s
0,0001	0,87467956	4 min 52,914 s
0,00001	0,87468838	8 min 12,017 s

Tabulka 6.1: Výsledky thresholdu regulárního kódu pro různé citlivosti počítané podle vzorce 4.7.

Citlivost	Threshold	Čas výpočtu
0,1	0,85937500	4 min 29,064 s
0,01	0,87451171	6 min 53,4480 s
0,001	0,87463378	10 min 25,5780 s
0,0001	0,87467956	16 min 8,3280 s
0,00001	0,87468838	32 min 44,0100 s

Tabulka 6.2: Výsledky thresholdu regulárního kódu pro různé citlivosti počítané podle vzorce 4.10.

Citlivost	Threshold	Čas výpočtu
0,1	0,88671875	15 min 38,227 s
0,01	0,88769531	22 min 57,534 s
0,001	0,88757324	27 min 43,554 s
0,0001	0,88748931	39 min 41,112 s
0,00001	0,88748645	51 min 24,594 s

Tabulka 6.3: Výsledky thresholdu iregulárního kódu pro různé citlivosti počítané podle vzorce 4.10.

V tabulce 6.4 jsou uvedeny časy výpočtů a výsledky thresholdu pro různé regulární LDPC kódy definované pomocí distribučních funkcí hodností ρ a λ. Tyto výpočty byly provedeny s počtem iterací 200 a $citlivost = 0,00001$. Threshold je vypočítaný programem uvedeným v kapitole 5.1. Časy výpočtů 1 jsou měřeny při výpočtu thresholdu podle rovnice 4.7. Časy výpočtů 2 jsou měřeny při výpočtech podle rovnice 4.10.

λ	ρ	Threshold	Čas výpočtu 1	Čas výpočtu 2
x^4	x^9	0,79102	10 min 35,01 s	1 h 22 min 30,072 s
x^3	x^9	0,74399	6 min 38,304 s	58 min 26,922 s
x^3	x^7	0,83235	7 min 48,108 s	55 min 14,772 s
x^3	x^5	1,00351	11 min 26,694 s	54 min 0,846 s
x^2	x^{11}	0,62975	9 min 11,508 s	1 h 36 min 1,374 s
x^2	x^8	0,70508	9 min 4,428 s	1 h 7 min 3,468 s
x^2	x^4	1,00024	11 min 44,748 s	44 min 34,716 s
x^2	x^3	1,25140	11 min 19,962 s	29 min 48,522 s

Tabulka 6.4: Výsledky thresholdu pro různé regulární LDPC kódy.

6.3 Diskretizovaná density evolution pro regulární LDPC kódy

V první části se opět tato kapitola zaměřuje na regulární kódy. Jako vstupní hodnoty byly použity distribuční funkce hodností λ a ρ specifikující kódy a počet iterací byl nastaven na 20.

$\lambda = 1x^2$

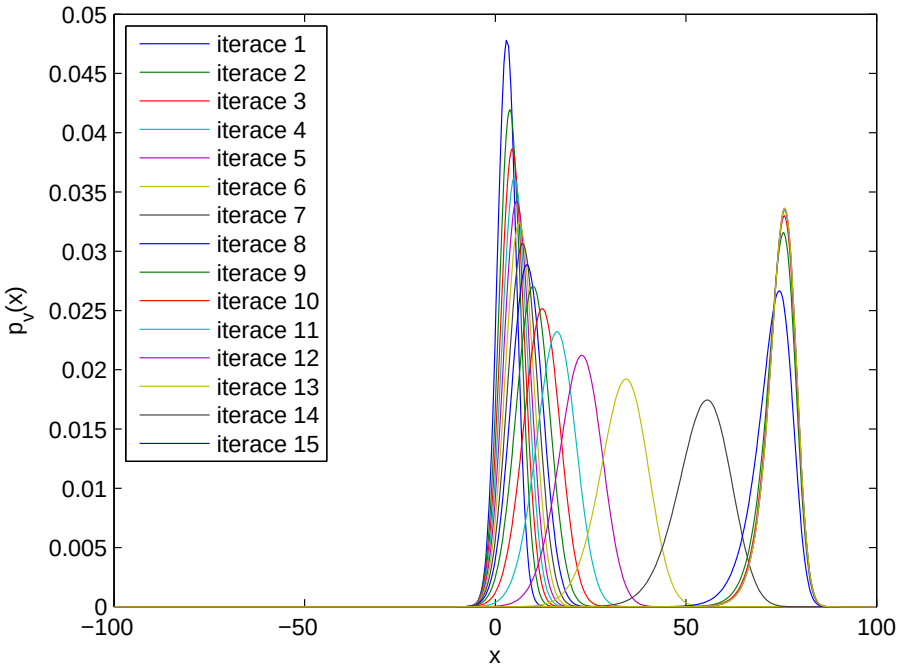
$\rho = 1x^5$

V tomto případě je ještě nutné vhodně nastavit hodnotu kvantizačního intervaluΔ, která se vypočítává z rozsahu p_0 a počtu kvantizačních úrovní. Tento výpočet je uveden v rovnici (6.1).

$$\Delta = \frac{|-15| + |15|}{počet\ kvantizačních\ úrovní} \tag{6.1}$$

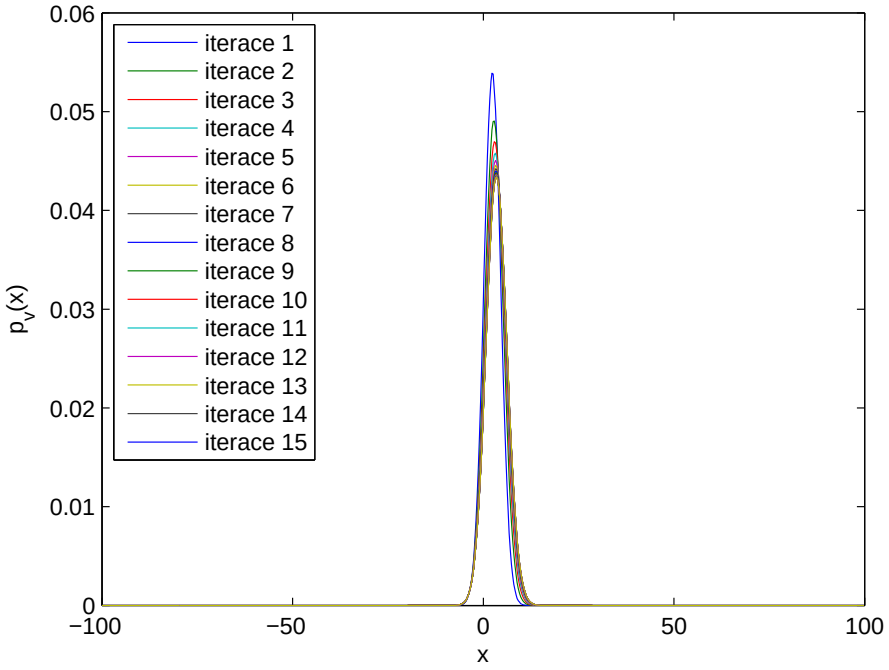
Pro naše výpočty jsme zvolili osu x pro p_0 od -15 do 15 a počet kvantizačních úrovní je roven 101. Po vypočítání rovnice (6.1) vyšlo v tomto příkladu $\Delta = 0,3$.

Hodnoty byly zvoleny s ohledem na rychlost výpočtu a přesnost. V případě diskretizovaného výpočtu sledujeme maximální hodnoty pravděpodobnostních funkcí a jejich pozici na ose x.



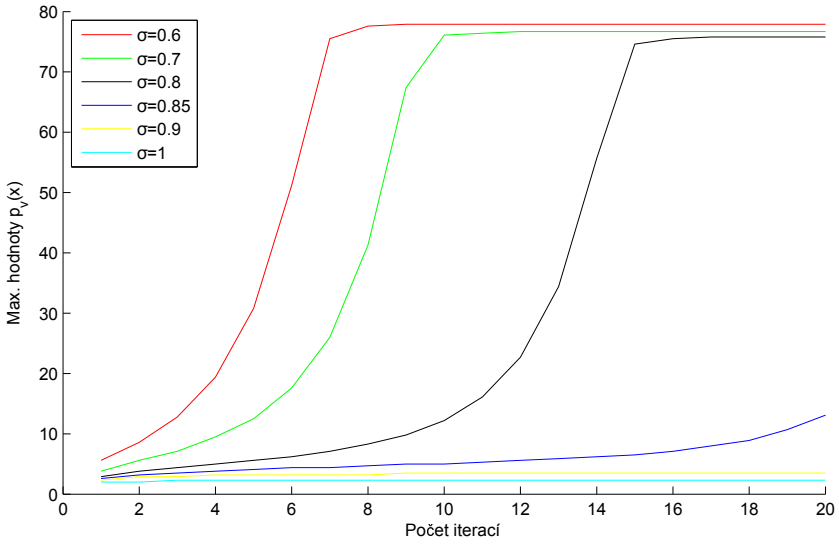
Obrázek 6.6: Výpočty pravděpodobnostních funkcí pro $\sigma = 0,8$ regulárního kódu

Graf 6.6 je vypočítán pro $\sigma = 0,8$ a počet iterací je roven 20. Je vidět, že maximální hodnota pravděpodobnostních funkcí se posouvá doprava na ose x s rostoucím počtem iterací. To znamená, že pro získání hodnoty thresholdu je zapotřebí zvýšit σ. Pro následující graf 6.7 jsme proto zvolili $\sigma = 0,9$.



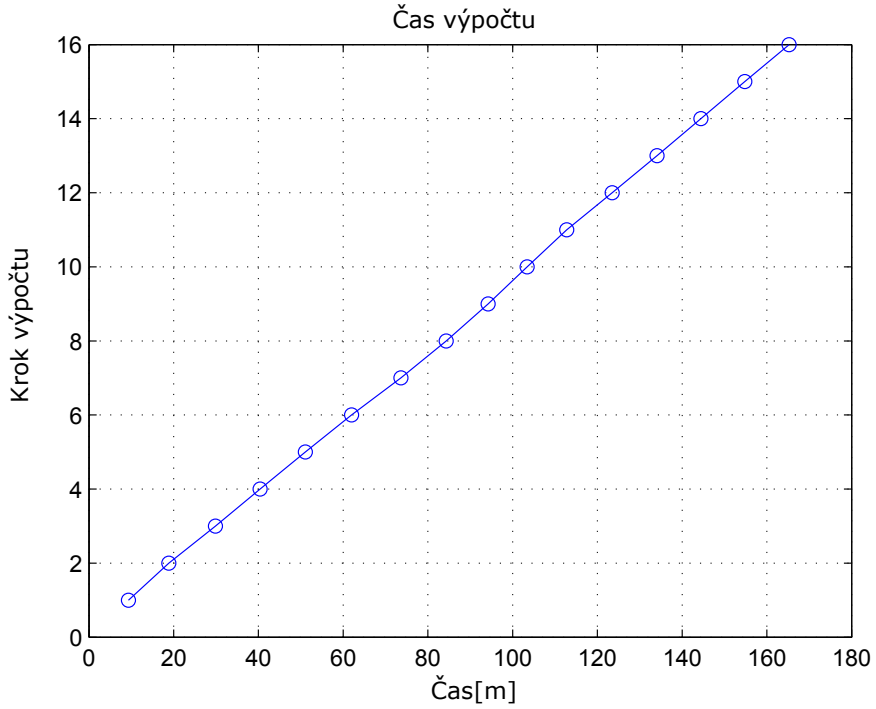
Obrázek 6.7: Výpočty pravděpodobnostních funkcí pro $\sigma = 0,9$ regulárního kódu

V grafu je zřejmé, že maximální hodnota pravděpodobnostních funkcí se neposouvá doprava na ose x. Z toho můžeme usoudit, že hodnota σ musí být nižší než 0,9 pro nalezený threshold. Znázornění argumenty maximálních hodnot pravděpodobnostních funkcí v závislosti na x je lépe zobrazeno v grafu 6.8, který je vypočítaný pro různé hodnoty σ.



Obrázek 6.8: argumenty maximálních hodnot pravděpodobnostních funkcí v závislosti na x pro regulární kód.

Z grafu je tedy patrné, že hodnota thresholdu se nachází mezi hodnotami $\sigma = 0,85$ a $\sigma = 0,9$. Pro zjištění přesnější hodnoty byla využita funkce `Threshold_funkce_for_Discretized.m` více v kapitole 5.3. Funkce využívá půlení intervalů pro zjištění přesné hodnoty thresholdu. Tento výsledek je dosažen s nastavenou *citlivost* $= 0,0001$, která určuje zastavovací podmínku pro půlení intervalů. Počet iterací je také snížen na hodnotu 10 z důvodu zdlouhavého výpočtu. Po dokončení výpočtu se threshold rovnal 0,8603. Výsledek byl spočítán v 16-ti krocích, jak je vidět z grafu 6.9 v čase 2 h 45 min 15,532 s.



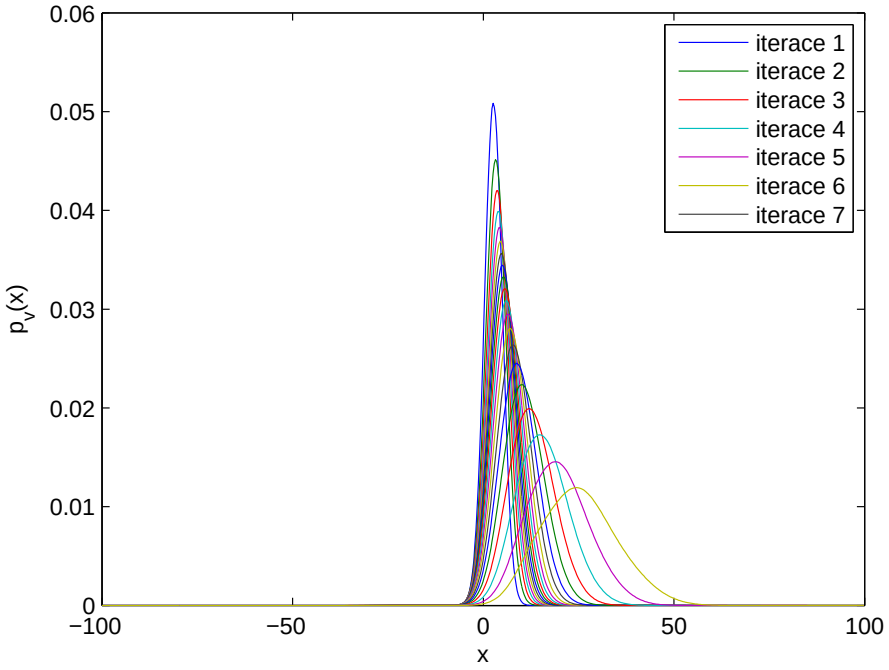
Obrázek 6.9: Čas výpočtu thresholdu pro regulární kód.

6.4 Diskretizovaná density evolution pro iregulární LDPC kódy

Tato část se zaměřujeme již jen na iregulární kódy. Podobně jako v první části i zde budou uvedeny grafy pro $\sigma = 0,85$(graf 6.10) a $\sigma = 0,9$(graf 6.11). V tomto případě bylo použito 20 iterací, osa x pro p_0 zůstala stejná od -15 do 15 a počet kvantizačních úrovní bylo použito 101. Po vypočítání rovnice (6.1) vyšlo pro tento příklad$\Delta = 0,3$. Iregulární kód byl definován pomocí distribuční funkce hodností λ a ρ.

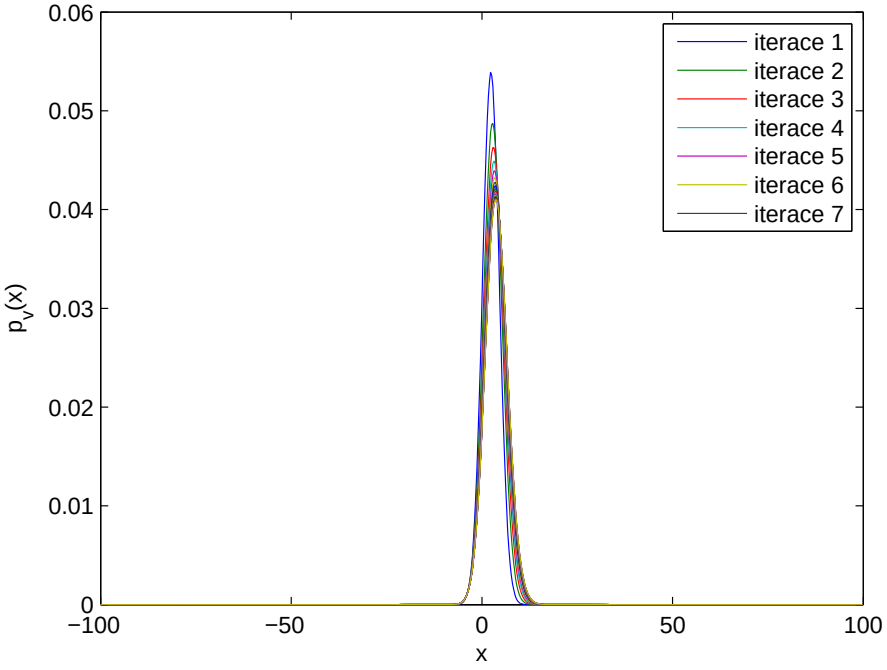
$\lambda = 0,15 + 0,55x + 0,3x^2$

$\rho = 1x^5$



Obrázek 6.10: Výpočty pravděpodobnostních funkcí pro $\sigma = 0,85$ iregulárního kódu.

Maximalní hodnotu šumu, při které je nejmenší pravděpodobnost chyb, budeme reprezentovat pomocí

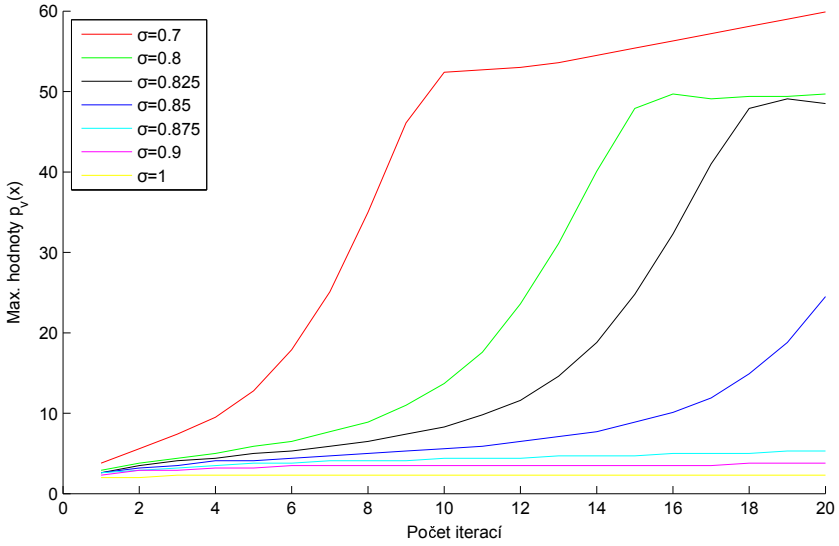


Thresholdu.

Obrázek 6.11: Výpočty pravděpodobnostních funkcí pro $\sigma = 0,9$ iregulárního kódu.

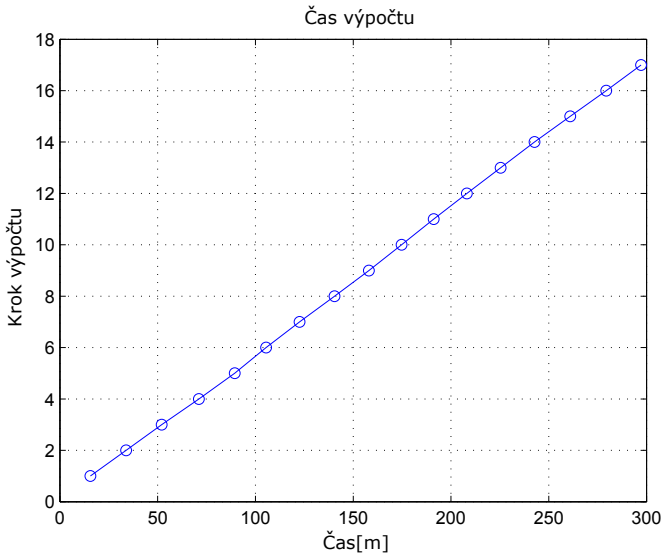
Z těchto grafů je zřejmé, že se threshold bude nacházet mezi těmito hodnotami. Pro

$\sigma = 0,85$ s rostoucím počtem iterací se maximální hodnoty pravděpodobnostních funkcí posouvají doprava na ose x, kdežto pro $\sigma = 0,9$ nikoliv. Následně v grafu 6.12 lze pozorovat argumenty maximálních hodnot pravděpodobnostních funkcí v závislosti na x pro různé hodnoty σ.



Obrázek 6.12: Argumenty maximálních hodnot pravděpodobnostních funkcí v závislosti na x pro iregulární kód.

Přesnější hodnota thresholdu spočítána funkcí `Threshold_funkce_for_Gaussian.m` vyšla 0,8651. Tato hodnota byla spočítána pro *citlivost* $= 0,0001$ a počet iterací byl snížen na 15. Výpočet byl proveden v 17-ti krocích a čas výpočtu iregulárního thresholdu trval 4 h 57 min 13,99 s. Jak je patrné v grafu 6.13.



Obrázek 6.13: Čas výpočtu thresholdu pro iregulární kód.

7 Závěr

Práce byla zaměřena na algoritmy density evolution a jejich implementaci. V průběhu práce byly prostudovány algoritmy density evolution v diskretizované a aproximované verzi. Oba algoritmy density evolution byly následně úspěšně naprogramovány v interaktivním programovém prostředí Matlabu. Distribuční funkce hodností ρ a λ jsou vstupní parametry do funkcí density evolution, které definují soubor LDPC kódů. Výstupem je threshold, který je definován jako maximální hladina šumu, při které pravděpodobnost chyb konverguje k nule s rostoucím počtem iterací k nekonečnu, za předpokladu použití kódu s nekonečnou délkou slova a s distribučními funkcemi hodností ρ a λ.

Z výsledků simulace uvedené v kapitolách 6.1, 6.2, 6.3 a 6.4 je zřejmé, že nejméně časově náročný byl výpočet gaussovské aproximace pro density evolution regulárních kódů podle rovnice (4.7). Gaussovská aproximace pro density evolution regulárních a iregulárních kódů podle rovnice (4.10) byla již výpočetně náročnější. Z tabulek 6.1 a 6.3 je zřejmé, že pro regulární kódy vypočítané podle rovnice (4.7) se pohybujeme v řádu minut, kdežto pro iregulární kódy počítané podle rovnice (4.10) se pohybujeme v řádu desítek minut. Přesnost gaussovské aproximace pro density evolution je dána citlivostí výpočtu thresholdu a počtem iterací. Citlivost určuje zastavovací podmínku půlení intervalů ve výpočtu thresholdu a počet iterací určuje velikost vektoru středních hodnot.

Výpočet diskretizované density evolution byl časově náročnější než výpočet s gaussovskou aproximací, jak je zřejmé z časů uvedených v kapitolách 6.3 a 6.4. Přesnost diskretizované density evolution je dána třemi parametry - citlivostí, počtem iterací a kvantizačním intervalem Δ. Citlivost je zastavovací podmínkou půlení intervalů, stejně jako u gaussovské aproximace pro density evolution. Počet iterací určuje množství vypočítaných pravděpodobnostních funkcí p_v a proměnná Δ určuje kvantizační interval, který je vypočítán z rovnice (6.1). Pro naše výpočty jsme zvolili $\Delta = 0,3$ kvůli menší časové náročnosti, ovšem na úkor přesnosti výsledku. Diskretizovaná verze se i přesto projevila jako pomalejší. Z teorie vyplývá, že diskretizovaná verze density evolution by měla být přesnější než Gaussova aproximace pro density evolution. To se ovšem nepodařilo prokázat kvůli časové náročnosti výpočtů diskretizované verze.

V kapitole 6.1 je postupně otestováno několik kódů definovaných distribučními funkcemi hodností ρ a λ. Kvůli časové náročnosti byly kódy testovány pouze za pomoci gaussovské aproximace pro density evolution. Výsledky těchto testů byly porovnány s literaturou. Z tabulky 6.4 je zřejmé, že výsledky vypočítané programem uvedeným v kapitole 5.1 se schodují s výsledky v literatuře [8]. Mírné komplikace způsobilo rozdílnost indexování Matlabu oproti literatuře.

Jelikož neexistují analytické metody pro návrh distribučních funkčních hodnot s co nejlepší thresholdem, využívají se tzv. optimalizační algoritmy pro řešení tohoto problému.

Naprogramované funkce uvedené v této práci mohou být využívány pro tyto optimalizační algoritmy, které umožní návrh výkonnějších LDPC kódů s dlouhou délkou kódového slova.

Reference

[1] KÓDY A KÓDOVÁNÍ. METODY KÓDOVÁNÍ: Studijní podpora (FRVŠ 3420/2006) [online]. Brno: Vysoké učení technické v Brně, 2006 [cit. 2017-05-10]. Dostupné z: http://www.uai.fme.vutbr.cz/~matousek/TIK/dokumenty/osmera_kap3.pdf

[2] BROULÍM, Jan. Využití samoopravného kódu v bezdrátovém přenosu. Plzeň, 2012. Diplomová práce. Západočeská univerzita v Plzni. Vedoucí práce Doc. Dr. Ing. Vjačeslav Georgiev.

[3] JORGE CASTIÑEIRA MOREIRA a PATRICK GUY FARRELL. Essentials of error-control coding [online]. Chichester: J. Wiley, 2006 [cit. 2017-05-11]. ISBN 04-700-2920-X.

[4] MACKAY: D. David MacKay's Gallager code resources [online]. 2008 [cit. 2017-05-10]. Dostupné z: http://www.inference.phy.cam.ac.uk/mackay/CodesFiles.html

[5] SAE-YOUNG CHUNG, T. J. RICHARDSON a R. L. URBANKE. Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation. IEEE Transactions on Information Theory [online]. 47(2), 657-670 [cit. 2017-05-11]. DOI: 10.1109/18.910580. ISSN 00189448. Dostupné z: http://ieeexplore.ieee.org/document/910580/

[6] SAE-YOUNG CHUNG, G. D. FORNEY, T. J. RICHARDSON a R. URBANKE. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. IEEE Communications Letters [online]. 2001, 5(2), 58-60 [cit. 2017-05-10]. DOI: 10.1109/4234.905935. ISSN 1089-7798. Dostupné z: http://ieeexplore.ieee.org/document/905935/

[7] WANG LIN, XIAO JUAN a GUANRONG CHEN. Density evolution method and threshold decision for irregular LDPC codes. 2004 International Conference on Communications, Circuits and Systems (IEEE Cat. No.04EX914) [online]. IEEE, 2004, 25-28 [cit. 2017-05-10]. DOI: 10.1109/ICCCAS.2004.1345932. ISBN 0-7803-8647-7. Dostupné z: http://ieeexplore.ieee.org/document/1345932/

[8] MOON, Todd K. Error correction coding: mathematical methods and algorithms. Hoboken, N.J.: Wiley-Interscience, 2005. ISBN 978-0-471-64800-0.

[9] HROUZA, Ondřej. LDPC KÓDY [online]. Brno, 2012 [cit. 2017-05-08]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=52086. DIPLOMOVÁ PRÁCE. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ - FAKULTA

ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ. Vedoucí práce Ing. PAVEL ŠILHAVÝ, Ph.D.

[10] Lectures: Density Evolution. Alex Balatsoukas-Stimming: M.Sc. Student & Teaching Assistant [online]. Technical University of Crete, Kounoupidiana Campus, Chania, Crete 73100, Greece: Technical University of Crete, 2009 [cit. 2017-05-08]. Dostupné z: http://www.telecom.tuc.gr/~alex/lectures/lecture11.pdf

Obsah DVD

Program v Matlabu

- Diskretizovaná verze density evolution
- Gaussova aproximace pro density evolution regulárních kódů
- Gaussova aproximace pro density evolution iregulárních kódů