Kódování – Turbo kódy

Coding – Turbo code

Bc. Jakub Kettner

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jakub KETTNER**
Osobní číslo:	**A08462**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Kódování – Turbo kódy**

Zásady pro vypracování:

1. **Seznamte se s problematikou Turbo kódů.**
2. **Zpracujte literární rešerši vysvětlující jednotlivé aspekty a principy funkce Turbo kódů. Zeměřte se na podrobné vysvětlení principu iterativního dekódování.**
3. **Vypracujte názorné příklady.**
4. **Naprogramujte algoritmus kódování a dekódování Turbo kódů.**
5. **Simulujte působení rušivých faktorů v přenosovém kanálu a na základě vytvořeného algoritmu porovnejte vliv nastavení jednotlivých parametrů Turbo kódů na BER.**
6. **Nastiňte možnosti využití Turbo kódů.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **VLCEK, K.: Komprese a kódová zabezpecení v multimediálních komunikacích, Praha, BEN -- technická literatura, 2004, ISBN 80-86056-68-6.**
2. **BIRKHOFF, G., BARTEE, T., C.: Aplikovaná algebra, Alfa, Bratislava, 1981.**
3. **HANZO, L, LIEW, T. H, YEAP, B. L. Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Wireless Channels, Wiley-IEEE Press, 2002. 768 s. ISBN 978-0-470-84726-8.**
4. **SCHLEGEL, Christian , PEREZ , Lance . Trellis and Turbo Coding. [s.l.] : Wiley-IEEE Press, 2004. 400 s. ISBN 978-0-471-22755-7.**
5. **SOLEYMANI, Mohammad, YINGZI, Gao, VILAIPORNSAWAI, U. Turbo Coding for Satellite and Wireless Communications. USA : Kluwer Academic Publishers, 2002. 248 s. ISBN 1-4020-7197-3.**

Vedoucí diplomové práce: **RNDr. Ing. Miloš Krčmář**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Tato práce poskytuje stručný přehled problematiky týkající se turbo-kódů. V úvodu jsou vysvětleny základní pojmy z oblasti přenosových kanálů, konvolučních a zřetězených kódů. Stěžejní část práce obsahuje popis nejdůležitějších vlastností turbo-kódů, struktury turbo-kodéru a objasnění funkce vybraných typů prokladačů. Dále je podrobně popsán princip dekódovacího algoritmu MAP a strategie iterativního dekódování. Teoretická část je zakončena krátkým pojednáním o aplikacích turbo-kódů a příkladem s demonstrací procesu kódování a dekódování. V praktické části jsou prezentovány výsledky počítačových simulací výkonnosti turbo-kódu v AWGN kanálu, kterých bylo dosaženo prostřednictvím naprogramovaného turbo-kodeku.

Klíčová slova: turbo-kódy, konvoluční kódy, dopředná korekce chyb (FEC), kanálové kódování, prokládání, iterativní dekódování, Maximum A-Posteriori (MAP) algoritmus, BER

ABSTRACT

This thesis is offering a brief overview of turbo-codes problems. The introduction explains fundamental terms of transmission channels, convolutional codes and concatenated codes. The thesis main part contains a description of the key features of turbo-codes, the structure of turbo encoder and the chosen interleaver types function explanation. There si also a chapter describing in detail the principle of decoding algorithm MAP and the iterative decoding strategy. A short discourse of turbo-codes application and an example demonstrating the process of an encoding and decoding closes the theoretical part of this thesis. The practical part contains computer simulations results of AWGN channel turbo-code performance obtained by the programmed turbo-codec.

Keywords: turbo-codes, convolutional codes, forward error correction (FEC), channel encoding, interleaving, iterative decoding, Maximum A-Posteriori (MAP) algorithm, BER

Mé poděkování patří panu RNDr. Ing. Miloši Krčmářovi za odborné vedení práce a cenné rady, panu Prof. Ing. Karlu Vlčkovi, CSc za zasvěcení do problematiky turbo-kódů a příjemné konzultace, a Ing. Jiřímu Gieslovi, za jeho ochotu, zájem a praktické rady. Chtěl bych poděkovat také své rodině za trpělivost a všeobecnou podporu během mého studia.

Počítače se mýlí mnohem přesněji.

Gabriel Laub

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

Narůstající informatizace společnosti přináší stále větší nároky na kvalitní přenos informací. Rostou vzdálenosti, přes které chceme data přenášet, a roste také samotný objem dat. Tento svět digitálních komunikací vděčí za svou existenci a fungování výsledkům mnoha vědních oborů.

V roce 1948 vyšel článek Claude E. Shannona – *A Mathematical Theory of Communication*, který položil základy dvou matematických teorií - teorie informace a teorie kódování. Teorie informace je matematickou teorií, která se zabývá zákonitostmi přenosu a zpracování informace. Teorie kódování se zabývá tím, jak rychle a spolehlivě přenášet informace z jednoho místa na druhé a jak najít takový způsob zápisu informace, který by umožnil dosáhnout hranic stanovených teorií informace. Samotná teorie kódování je pěkným příkladem využití algebraických metod na řešení praktických problémů vznikajících při zpracování informace.

Tato práce je zaměřena na tzv. kanálové bezpečnostní kódy. Cílem kanálového kódování je zabezpečit signál proti chybám vznikajícím při přenosu v komunikačním kanálu. Podstatou zabezpečení je úmyslné a kontrolované zvýšení jeho redundance (např. přidáním jistého počtu kontrolních bitů). Díky této redundantní složce pak algoritmy, které provádějí inverzní operaci (kanálové dekódování), umožňují získat užitečný signál. V roce 1949 Claude E. Shannon zavedl matematické základy pro posuzování šumového přenosového kanálu. Ve své analýze stanovil maximální teoretickou kapacitu (přenosovou rychlost) pro komunikační kanál, tzv. Shannonův limit. V dalších letech bylo navrženo mnoho dobře promyšlených kódů, posouvajících se směrem k hranici Shannonova limitu, všichni uchazeči však pro svou funkci blízko tohoto limitu vyžadovali velmi dlouhá kódová slova, což mělo za důsledek velkou složitost a náklady. Do relativně nedávné doby se tak nejlepší používané kódy dostaly při chybovosti 1:100000 zhruba na hodnotu 3,5 dB od Shannonova limitu.

Až v roce 1993 Berrou, Glavieux a Thitimajshim [1] navrhli novou třídu konvolučních kódů, tzv. *turbo-kódy*, jejichž výkon je z hlediska bitové chybovosti *BER* (*Bit Error Rate*) blízko hranice Shannonova limitu. Na sedmi stránkách autoři popsali přístup ke kódování, které umožňuje dosáhnout hodnoty 0,7 dB od Shannonova limitu. Potenciál výkonnosti, které nabízí turbo-kódy okouzlil akademické i průmyslové pracovníky. Během dalších let

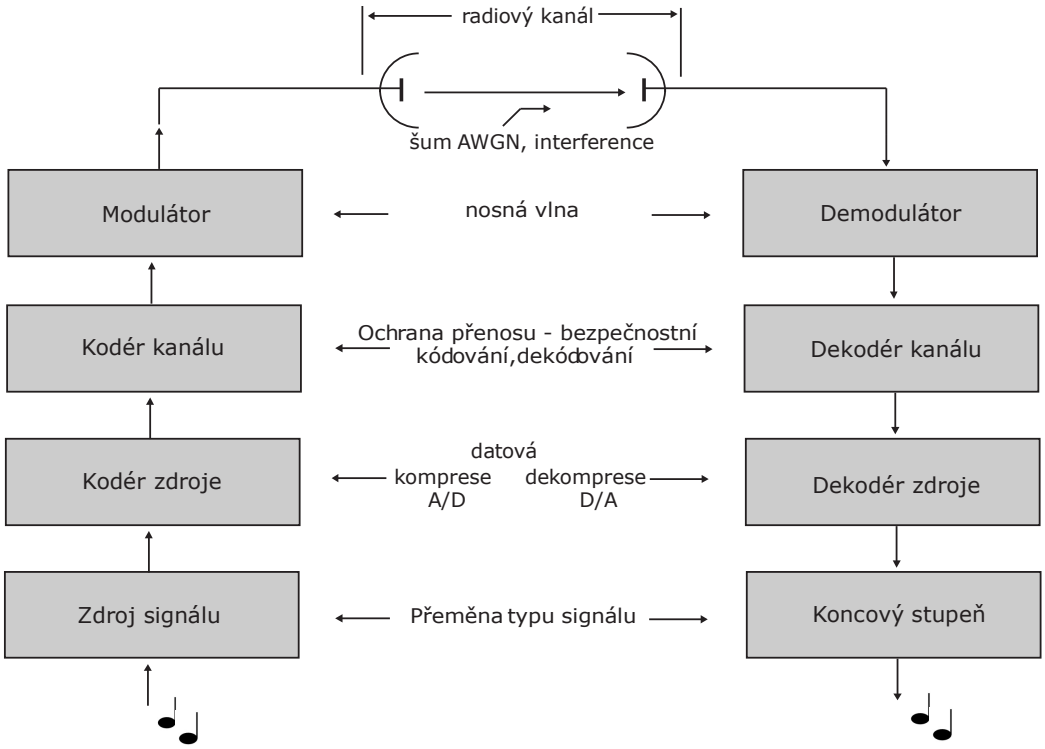
došlo k explozi výzkumu všech aspektů turbo-kódů, což vedlo k jejich značnému rozšíření a praktickému nasazení. Za všechny může být jmenováno např. využití v bezdrátové technologii pro přenos dat *WiMax*, začlenění do standardů pro třetí generaci mobilních přenosů, ve standardu televizního digitálního vysíláni *DVB*, či v komunikačním systému vesmírné sondy *Mars Reconnaissance Orbiter* a celé řady dalších kosmických zařízení.

I. TEORETICKÁ ČÁST

1 ZÁKLADNÍ POJMY

1.1 Komunikační systém

S komunikačním systémem se setkáváme všude, kde se pohybuje informace z jednoho místa na druhé, ať už v živém organismu nebo v umělém zařízení. Budeme-li uvažovat pouze systémy digitální, lze všechny komunikační systémy, které byly používány v minulosti i přítomnosti, zobecnit do tzv. *obecného komunikačního systému* (Obr. 1 [2]). Toto zobecnění koncipoval Shannon v 50. letech, a jedná se určitou idealizaci, neboť v praxi může jít o výrazně složitější strukturu s řadou dalších funkčních bloků.



Obr. 1. Obecné schéma rádiového komunikačního systému [2]

Úkolem *kodéru zdroje* je provést kódováni užitečného signálu, tedy redukci bitové rychlosti, resp. kompresi dat tak, aby byl použit co nejmenší počet znaků, které jsou vhodné pro přenos rádiovým kanálem. *Dekodér zdroje* na přijímací straně provádí inverzní operaci, na svém výstupu tedy poskytuje užitečný signál, který by se měl co nejvíc shodovat se signálem na vstupu kodéru zdroje. Následujícím blokem systému je *kodér kanálu*, jehož úkolem je zabezpečit spolehlivost přenosu tím, že doplňuje informační znaky o přídavné znaky podle určitého algoritmu bezpečnostního kódu. Na přijímací straně

odpovídá tomuto bloku *dekodér kanálu*, který detekuje či opravuje případné chyby vzniklé při přenosu a rekonstruuje signál tak, aby odpovídal výstupnímu signálu kodéru zdroje. Právě tyto dva bloky jsou hlavním předmětem zájmu předkládané práce. Do systému jsou dále zahrnuty ostatní transformace signálu vznikající při přenosu (modulátor, demodulátor, přenosové médium, působení rušení atd.).

1.2 Diskrétní kanál bez paměti

Diskrétní kanál představuje abeceda vstupního zdroje $X = (x_1, x_2, ...x_{Nx})$, výstupní abeceda $Y = (y_1, y_2, ...y_{Ny})$ a soubor podmíněných pravděpodobností $P(y_j \mid x_i)$, které reprezentují pravděpodobnost příjmu symbolu y_j za předpokladu, že byl vyslán symbol x_i. Pravděpodobnost toho, že *n* přijatých symbolů je určených *n*-ticí vyslaných symbolů, odpovídá u kanálu bez paměti součinu individuálních pravděpodobností těchto symbolů:

$$P(y_j \mid x_i) = \prod_{j,i=1}^{n} P(y_j \mid x_i) \tag{1}$$

Pro kvantitativní hodnocení přenosu informace byla zavedena veličina $I(X,Y)$ nazývaná vzájemná informace [3]:

$$I(X,Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X), \tag{2}$$

kde veličiny $H(X)$ a $H(Y)$ se nazývají *vstupní* a *výstupní entropie* a jsou definovány vztahy:

$$H(X) = -\sum_{i=1}^{Nx} P(x_i) \log P(x_i) \tag{3}$$

$$H(Y) = -\sum_{j=1}^{Ny} P(y_j) \log P(y_j) \tag{4}$$

Veličiny $H(Y \mid X)$ a $H(X \mid Y)$ se nazývají *podmíněné entropie* a jsou definovány vztahy:

$$H(Y \mid X) = -\sum_{i=1}^{Nx} \sum_{j=1}^{Ny} P(x_i, y_j) \log P(y_j \mid x_i) \tag{5}$$

$$H(X \mid Y) = -\sum_{i=1}^{Nx} \sum_{j=1}^{Ny} P(x_i, y_j) \log P(x_i \mid y_j), \tag{6}$$

jež udávají množství informace potřebné k určení výstupního, resp. vstupního symbolu, je-li znám vstupní, resp. výstupní symbol. Entropii $H(Y \mid X)$ lze tedy chápat jako množství přidané, rušivé informace a $H(X \mid Y)$ jako množství informace ztracené při přenosu.

Schopnost kanálu přenášet informaci je popisována veličinou nazývanou *kapacita kanálu*. Ta je určena jako maximum vzájemné informace přes všechna možná rozdělení pravděpodobností vstupních symbolů, tedy:

$$C = \max_{P(x)} I(X,Y), \tag{7}$$

kde *C* je kapacita kanálu [*Sh/symbol*]. Pro binární symetrický kanál je kapacita kanálu dána vztahem [3]:

$$C = 1 + (1 - P_{ch}) \log(1 - P_{ch}) + P_{ch} \log P_{ch}, \tag{8}$$

kde P_{ch} je pravděpodobnost chybného přenesení znaku.

1.2.1 Bayesův teorém

Na tomto místě bude uvedena krátká odbočka do teorie pravděpodobnosti, která je však stejně jako s mnoha jinými obory i s informační a komunikační teorií pevně spjata.

Uvažujme jednotlivé pravděpodobnosti vstupních symbolů $P(x_i)$ a pravděpodobnosti výstupních symbolů $P(y_j)$. Víme-li, že výsledkem náhodného pokusu je jev y_j, můžeme stanovit podmíněné pravděpodobnosti hypotéz x_i vzhledem k jevu y_j pomocí *Bayesova vzorce* [4]:

$$P(x_i \mid y_j) = \frac{P(y_j \mid x_i) \cdot P(x_i)}{P(y_j)}, \tag{9}$$

kde

$$P(y_j) = \sum_{i=1}^{Nx} P(y_j \mid x_i) \cdot P(x_i) \tag{10}$$

Bayesův vzorec umožňuje vypočítat pravděpodobnosti hypotéz x_i po provedení náhodného pokusu, jehož výsledkem bylo nastoupení jevu y_j. Tyto podmíněné pravděpodobnosti které, berou v úvahu výsledek náhodného pokusu, se nazývají pravděpodobnosti *a-posteriori* (*APP*) na rozdíl od pravděpodobností $P(x_i)$, které jsou známy před

provedením pokusu a nazývají se pravděpodobnosti *a-priori* [4]. Dalším důležitým vztahem, jenž vychází z (9), je vyjádření společné pravděpodobnosti $P(x_i, y_j)$ (možno označit také jako $P(x_i \cap y_j)$) pomocí podmíněné pravděpodobnosti:

$$P(x_i \cap y_j) = P(y_j \mid x_i) \cdot P(x_i) = P(x_i \mid y_j) \cdot P(y_j) \tag{11}$$

Vztah (9) a (11) bude využit v kapitole o dekódování turbo-kódů.

1.3 Kanál s AWGN

Kapacita kanálu s *aditivním bílým Gaussovským šumem* (*AWGN*) je dána vztahem [5]:

$$C = B \log_2 \left(1 + \frac{P}{N}\right), \tag{12}$$

kde B [*Hz*] je *šířka pásma*, P je *výkon signálu* a $N = N_0 B$ je *výkon šumu* [2] vyjádřen jako součin *spektrální výkonové hustoty šumu* N_0 a šířky pásma B. Kompletní odvození (12) uvádí např. [2], [5]. Označíme-li *přenosovou rychlost* jako R [*bit/s*] a *střední energii na jeden bit přenášené informace* jako E_b [*J*], je možná psát $P = E_b R$ a dosazením do rovnice (12) získáme vztah:

$$C = B \log_2 \left(1 + \frac{E_b}{N_0} \cdot \frac{R}{B}\right) \tag{13}$$

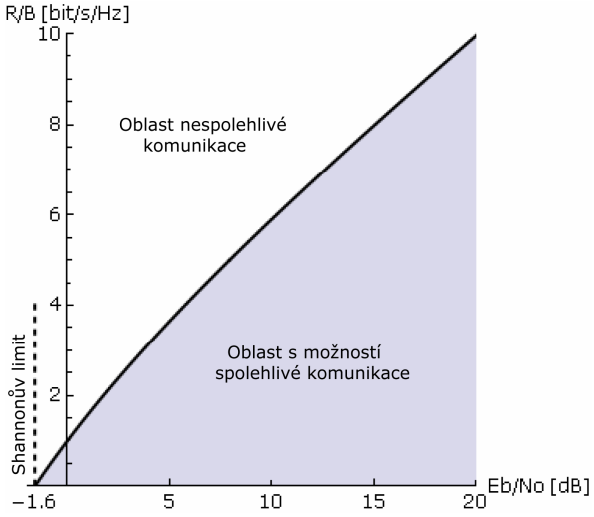
Z o*brácené věty Shannonova kódovacího teorému* vyplývá, že chceme-li komunikovat s libovolně nízkou pravděpodobností výskytu chyb, nesmí být přenosová rychlost větší než kapacita kanálu, tedy $R \leq C$. S přihlédnutím k této skutečnosti a vztahu (13), lze tuto nerovnost přepsat jako:

$$\frac{R}{B} \leq \frac{C}{B} = \log_2 \left(1 + \frac{E_b}{N_0} \cdot \frac{R}{B}\right) \tag{14}$$

Vyjádříme-li z rovnice (14) hodnotu E_b / N_0, můžeme psát [5]:

$$\frac{E_b}{N_0} > \frac{2^{\frac{R}{B}} - 1}{\frac{R}{B}} = \frac{2^{\eta} - 1}{\eta}, \tag{15}$$

kde poměr $\eta = R/B$ [*bit/s/Hz*] se nazývá *spektrální účinnost.* Ze vztahu (15) je možné určit limitní hodnotu poměru E_b / N_0 pro spektrální účinnost blížící se nule: $\lim_{\eta \to 0}\left(\frac{E_b}{N_0}\right) = \lim_{\eta \to 0}\left(\frac{2^\eta - 1}{\eta}\right) = \log_2(2) = 0{,}693 \quad tj. -1{,}59 \approx -1{,}6\,dB$. Tato minimální hodnota poměru $E_b / N_0 = -1{,}6\,dB$, se nazývá *Shannonův limit* [2] a říká, že v AWGN kanálu, pro poměr $E_b / N_0 < -1{,}6\,dB$, není uskutečnitelná spolehlivá komunikace, a to bez ohledu na to, jak kvalitní kódování je použito.



Obr. 2. Závislost maximální dosažitelné spektrální účinnosti na poměru E_b / N_0.

Graf na obrázku (Obr. 2) vyjadřuje základní problém digitální rádiové komunikace: má-li narůstat při dané šířce pásma *B* přenosová rychlost *R*, musí se zvětšovat poměr E_b / N_0. Toto dilema rádiové komunikace se však postupně odstraňuje, a to především díky zdokonalujícím se metodám kanálového kódování (objev turbo-kódů aj.), rozvoji kódových modulací apod. [2].

2 BEZPEČNOSTNÍ KÓDOVÁNÍ

Systémy bezpečnostního kódování lze rozdělit do dvou širokých kategorií:

1. Při kódování jsou k informační sekvenci přidávány paritní bity, které slouží k pouhé detekci, zda při přenosu nastala chyba. Takové kódování (kódy) nazýváme *detekční*. Této techniky využívá např. systém *automatického opakování přenosu na základě požadavku*, zkráceně *ARQ* (*Automatic Repeat Request*). Jsou-li na přijímací straně zjištěny chyby, je směrem ke zdroji zprávy vyslána žádost o opětovný přenos. Spojení mezi vysílačem a přijímačem tedy musí být obousměrné.
2. Druhý způsob zvýšení spolehlivosti přenosu je tzv. *dopředná korekce chyb*, zkráceně *FEC* (*Forvard Error Correction*). Dekodér se snaží v tomto případě chyby nejen odhalit, ale zároveň opravit. Právě tato strategie bude náplní následujících kapitol. Kódy využívající FEC můžeme rozdělit do dvou hlavních kategorií: *Blokové kódy* a *Konvoluční kódy*.

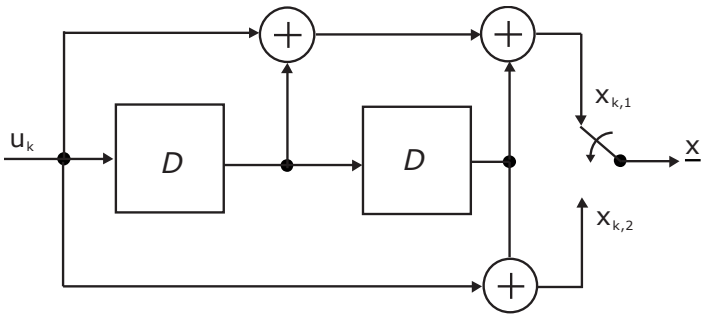
2.1 Lineární blokové kódy

Blokové kódy jsou charakteristické tím, že zobrazení vstupního, informačního slova do kódového slova probíhá pouze v rámci bloku dat s pevnou délkou. Kód je určen dvojicí *(n, κ)*, kde *n* je délka kódu a κ počet informačních znaků v kódové kombinaci. *Lineární blokové kódy* se vyznačují tím, že libovolná lineární kombinace kódových slov je opět kódovým slovem. Při použití tzv. *systematického kódu* lze rozlišit část, která je původním informačním slovem (informační bity) a část, která byla přidána z důvodu kontroly nebo zabezpečení informace (zabezpečovací nebo paritní bity). Matematické operace se symboly (bity) jsou tzv. *sčítání* a násobení *modulo-2* [6]. Podrobnější vysvětlení problematiky lineárních blokových kódů je např. v [3], [5], [7].

2.2 Konvoluční kódy

NSC (*Non-Systematic Convolutional*) kodér konvolučního kódu je stavový automat s konečným počtem 2^M stavů, který je možné popsat třemi základními parametry (n, κ, m), případně (n, κ, K). V časovém okamžiku *k* přijímá kodér κ informačních bitů, generuje *n* výstupních bitů a přechází do nového stavu, viz stavový diagram na obrázku

(Obr. 4). Tento proces je možné považovat za konvoluci impulsní odezvy kodéru a vstupního signálu, což se promítlo i do názvu těchto kódů.



Obr. 3. NSC$[7,5]_8$ *kodér (*$g_1 = 7$ *a* $g_2 = 5$*)*

Vstupující bity procházejí přes lineární posuvný registr a *n*-tice výstupních bitů je výsledkem operace sčítání modulo-2 (*XOR*) nad aktuálním vstupním bitem a výstupy z různých stupňů posuvného registru. Je-li *velikost paměti p*-tého vstupu označena jako m_p, pak *paměť kodéru* odpovídá hodnotě $m = \max(m_p)$ a pro *celkovou paměť kodéru* platí:

$$M = \sum_{p=1}^{\kappa} m_p \tag{16}$$

Konvoluční kód je lineární, časově invariantní mřížkový kód s konečnou *délkou kódového omezení K*, která říká, kolik κ-tic zdrojových symbolů ovlivňuje jednu *n*-tici kódových symbolů [7]. Kódový poměr je určen podílem:

$$R = \frac{\kappa}{n} \tag{17}$$

Délka kódového omezení *K* je dána vztahem:

$$K = 1 + m \tag{18}$$

Vyjádříme-li vstupní sekvenci jako $u = (u_0, u_1, u_2, ...)$ a uvažujeme-li kodér s $n = 2$, kde výstupní sekvence jsou $x_1 = (x_{0,1}, x_{1,1}, x_{2,1}, ...)$ a $x_2 = (x_{0,2}, x_{1,2}, x_{2,2}, ...)$, mohou impulsní odezvy trvat nejvýše *m*+1 časových jednotek a jsou dány jako $g_1 = (g_{0,1}, g_{1,1}, g_{2,1}, ..., g_{m,1})$ a $g_2 = (g_{0,2}, g_{1,2}, g_{2,2}, ..., g_{m,2})$. Pro kodér konvolučního kódu na obrázku (Obr. 3) s $m = 2$ je $g_1 = (1,1,1)$ a $g_2 = (1,0,1)$.

Tyto impulsní odezvy g_1 a g_2 se nazývají *generátory*. Kódovací rovnice pak může být zapsána jako:

$$x_1 = u * g_1 \tag{19}$$

$$x_2 = u * g_2, \tag{20}$$

kde operace * značí diskrétní konvoluci a všechny dílčí operace jsou modulo-2. Za předpokladu, že pro všechny kroky $k < i$ je $u_{k-i} = 0$, platí pro všechna $k \geq 0$:

$$x_{k,j} = \sum_{i=0}^{m} u_{k-i} g_{i,j} = u_k g_{0,j} + u_{k-1} g_{1,j} + ... + u_{k-m} g_{m,j}, \quad j = 1,2 \tag{21}$$

Pro kodér na obrázku (Obr. 3) je $x_{k,1} = u_k 1 + u_{k-1} 1 + u_{k-2} 1$ a $x_{k,2} = u_k 1 + u_{k-1} 0 + u_{k-2} 1$. Na těchto dvou kódových sekvencích je provedena operace multiplexování, čímž vzniká sekvence označena jako *kódové slovo* $\underline{x}$, která je vyslána přenosovým kanálem. Toto kódové slovo má tvar $\underline{x} = (x_{0,1}\, x_{0,2}, x_{1,1}\, x_{1,2}, x_{2,1}\, x_{2,2}, ...)$.[8]

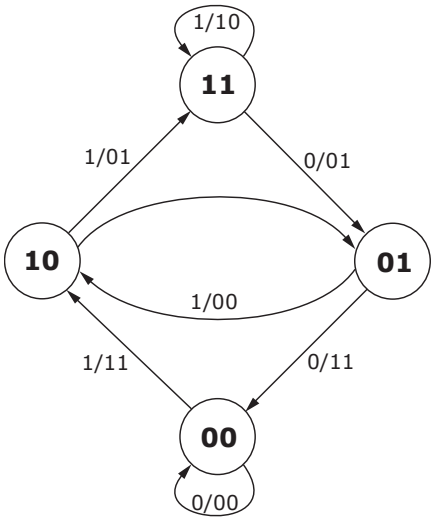
Další možností zápisu generátorů, které bude využíváno v následujících kapitolách, je zápis v osmičkové soustavě - pro kodér na obrázku (Obr. 3) je tedy $g_1 = 7$ a $g_2 = 5$. Takový kodér je pak možno označit jako $\mathrm{NSC}[7,5]_8$. Zavedeme-li proměnnou *D*, která reprezentuje jednotkové zpoždění, je možný zápis generátorů prostřednictvím *generujících polynomů* $g_1(D) = 1 + D + D^2$ a $g_2(D) = 1 + D^2$. Tyto polynomy lze přepsat do tzv. *generující matice*, která má pro tento případ tvar $G(D) = \left[g_1(D) \quad g_2(D)\right]$, a pro kódování pak platí:

$$\underline{x}(D) = u(D)G(D), \tag{22}$$

kde $u(D)$ je polynom popisující vstupní sekvenci a $\underline{x}(D) = \left[x_1(D) \quad x_2(D)\right]$ je vektor výstupních polynomů.

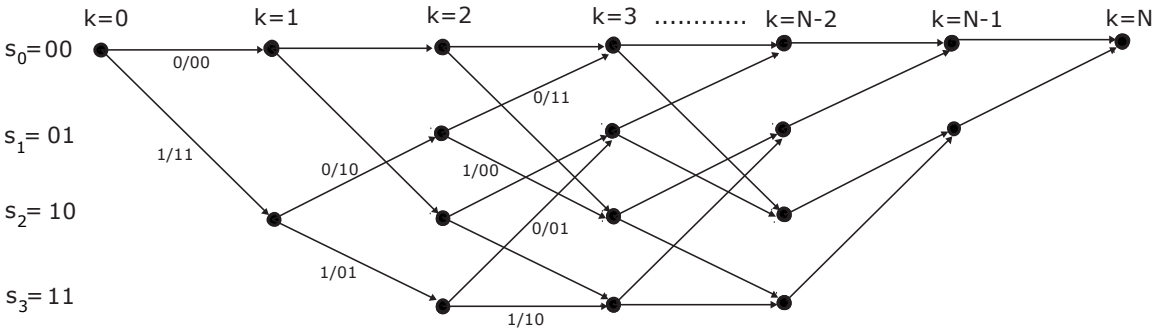
2.2.1 Stavový a trellis diagram

Na obrázku (Obr. 4) je znázorněn stavový digram pro $\mathrm{NSC}[7,5]_8$ kodér. Přechody mezi jednotlivými vnitřními stavy kodéru jsou znázorněny šipkami označenými symboly $u_k / x_{k,1} x_{k,2}$, které mají stejný význam, jaký byl uveden výše.



Obr. 4. Stavový digram $NSC[7,5]_8$ kodéru

Stavový diagram může být rozšířen na *trellis diagram*, který ukazuje jednotlivé časové kroky *k* a jim příslušející stavy $s_{i,k}$, kde *i* = 0,1,2,3. Ukázka trellis diagramu $NSC[7,5]_8$ kodéru je na obrázku (Obr. 5). V případě konvolučních kódů je každé kódové slovo asociováno s unikátní cestou skrz diagram. Tato cesta se nazývá *stavová sekvence*, a je vyjádřena jako $s = (s_{0,0}, ...s_{0,N})$, kde *N* je celková délka vstupní sekvence. [23]

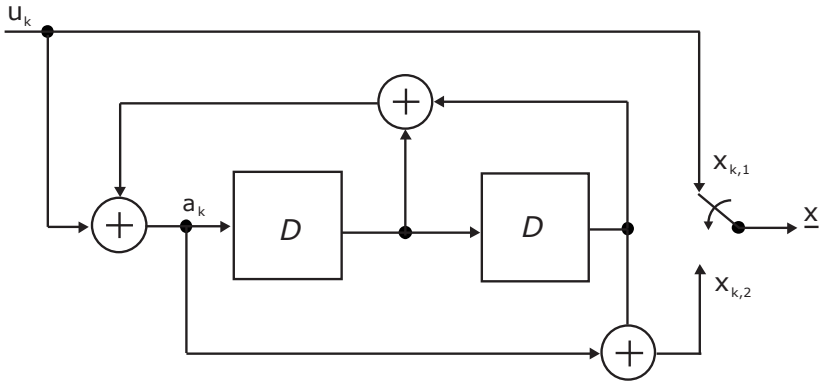


Obr. 5. Trellis digram $NSC[7,5]_8$ kodéru

2.3 Rekurzivní systematické konvoluční kódy

RSC (*Recursive Systematic Convolutional*) kodér konvolučního kódu kombinuje vlastnosti NSC a systematického kódu. Binární RSC kód je z kódu NSC získán přidáním zpětnovazební smyčky a nastavením jednoho ze dvou výstupů $x_{k,1}$ nebo $x_{k,2}$ přímo na hodnotu vstupního bitu u_k. Pro následující vztahy bude uvažováno $x_{k,1} = u_k$, jak zachycuje

RSC kodér na obrázku (Obr. 6). Bity na výstupu $x_{k,1}$ budou označovány jako systematické informační bity a bity na výstupu $x_{k,2}$ jako paritní bity.



Obr. 6. RSC$[7,5]_8$ *kodér*

Na vstupu posuvného registru se objevuje nová proměnná a_k, která je počítána rekurzivně podle vztahu:

$$a_k = u_k + \sum_{i=1}^{m} a_{k-1} g_{i,1}, \quad pro \quad k = 0, 1, ..., N-1, \tag{23}$$

kde hodnota $a_{-1} = 0$. Paritní bit, který se objeví na výstupu x_2 je počítán podle vtahu:

$$x_{k,2} = \sum_{i=0}^{m} a_{k-1} g_{i,2} \quad pro \quad k = 0, 1, ..., N-1 \tag{24}$$

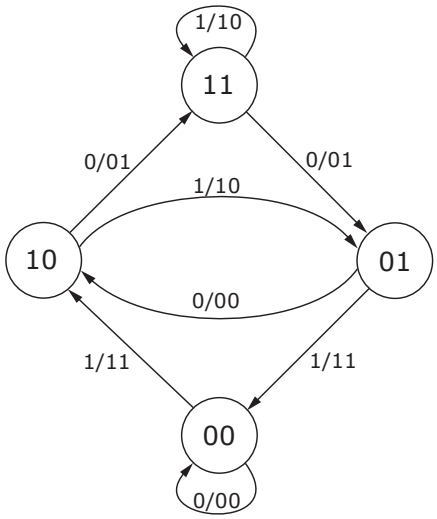
Generující matice RSC kodéru se nazývá *rekurzivní generující matice*, a má tvar $G_R(D) = \left[1 \quad \frac{g_2(D)}{g_1(D)}\right]$, kde generující polynomy $g_1(D)$ a $g_2(D)$ jsou shodné s polynomy NSC kodéru. Důležitou vlastností konvolučních kodérů je *impulsní odezva.* V případě NSC kodéru je tato odezva konečná (*FIR*), v případě RSC kodéru je nekonečná (*IIR*) a periodická. Pro periodu impulsní odezvy RSC platí:

$$p \leq 2^m - 1 \tag{25}$$

Tato vlastnost přináší výhodu zejména tehdy, chceme-li docílit co největší váhy výstupní sekvence při poměrně jednoduché konstrukci kodéru. Také z tohoto důvodu našly právě RSC kodéry využití v architektuře turbo-kódů, viz kapitola (kap. 3.2).

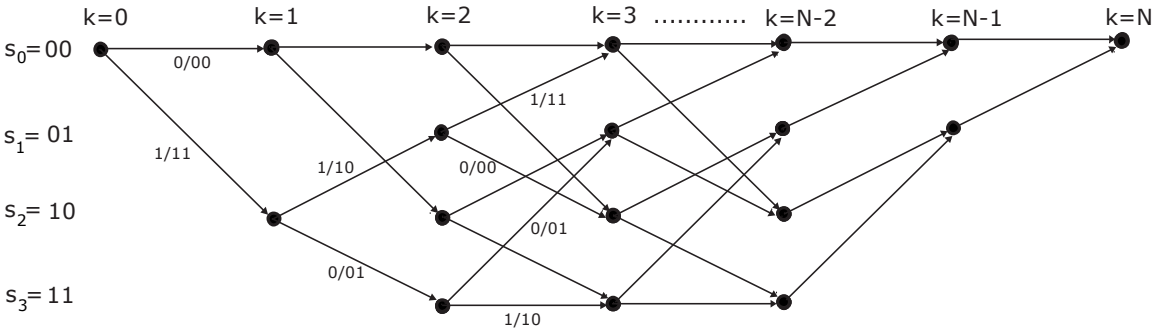
2.3.1 Stavový a trellis diagram

RSC kodér je rovněž konečný automat. Stavový diagram $\mathrm{RSC}[7,5]_8$ kodéru je uveden na obrázku (Obr. 7).



Obr. 7. Stavový digram RSC$[7,5]_8$ *kodéru*

Konstrukce trellis diagramu RSC kodéru je stejná jako u NSC, rozdílné jsou však symboly $u_k / x_{k,1}x_{k,2}$ uvedené u přechodů mezi jednotlivými stavy, což je způsobeno rozdílnou odezvou RSC kodéru na vstupní bity (zpětná vazba) a vlastností, že $u_k = x_{1,k}$.



Obr. 8. Trellis digram RSC$[7,5]_8$ *kodéru*

2.4 Vyprázdnění NSC a RSC kodéru

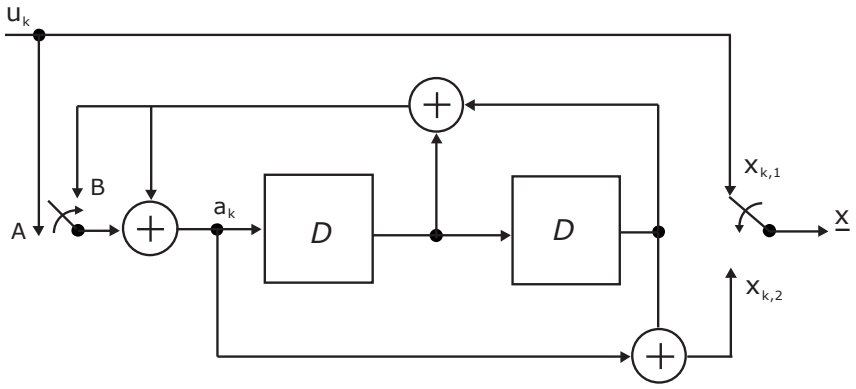
Na rozdíl od blokových kódů nemají obecné konvoluční kódy stanovenou fixní délku, pro mnoho aplikací (např. přítomnost prokladače u turbo-kódů, viz kapitola (kap. 3.4)) je však tato vlastnost nezbytná. V takovém případě je nutné uvést kodér zpět do nulového stavu, tedy provést jeho *vyprázdnění* (*termination*).

Situace u NSC kodéru je poměrně jednoduchá, kdy je konečného stavu $s_{0,N}$ docíleno doplněním vstupní sekvence o takové množství nulových bitů, jaká je velikost paměti kodéru *m*.

Jelikož RSC kodér má nekonečnou impulsní odezvu, tato situace zde neplatí. Pro vyprázdnění kodéru je nutné doplnit vstupní sekvenci o *m* bitů u'_k.

$$u'_k = \sum_{i=1}^{m} a_{k-i} g_{i,1}, \quad pro \quad N - m \leq k < N \tag{26}$$

Tento princip je zachycen rovněž na obrázku (Obr. 9), kdy po dobu $k < N - m$ je přepínač na vstupu ponechán v poloze A a v posledních m krocích je přepnut do polohy B.



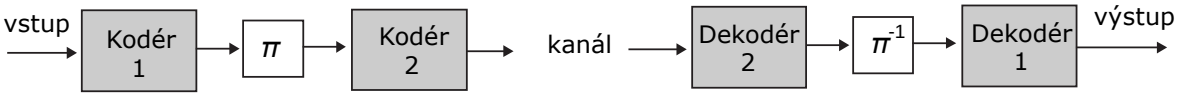
Obr. 9. Vyprázdnění RSC kodéru [9]

Pro oba typy kódů platí, že je nutné blok *N-m* datových bitů doplnit vhodným způsobem o *m* bitů, tzv. *konec zprávy* [7] (*tail of the message*). Z tohoto důvodu dochází k mírnému snížení kódového poměru z hodnoty $R = \frac{\kappa}{n}$ na hodnotu $R' = \frac{N-m}{nN}$, rozdíl $R - R'$ je však při větších hodnotách *N* zanedbatelný.

2.5 Zřetězené kódy

U složitějších přenosových systémů nejsme schopni docílit požadované úrovně ochrany přenosu prostřednictvím jednoho kanálového kódu. Z Shannonovy práce vyplývá, že kódy s velmi dobrou zabezpečovací schopností musí mít dobrý informační poměr. Této vlastnosti snáze dosáhneme, budou-li kódová slova co nejdelší. Naplnění tohoto požadavku při únosné míře složitosti kódovacích a dekódovacích procedur nabízejí tzv. *zřetězené kódy*, které jsou založené na spojení dvou či více jednoduchých kódů a tzv. *prokládání* (*interleaving*). Existují tři základní typy zřetězení, a sice sériové, paralelní a hybridní.

Příklad zjednodušeného uspořádání prvních dvou typů je zachycen na obrázku (Obr. 10) a (Obr. 11).

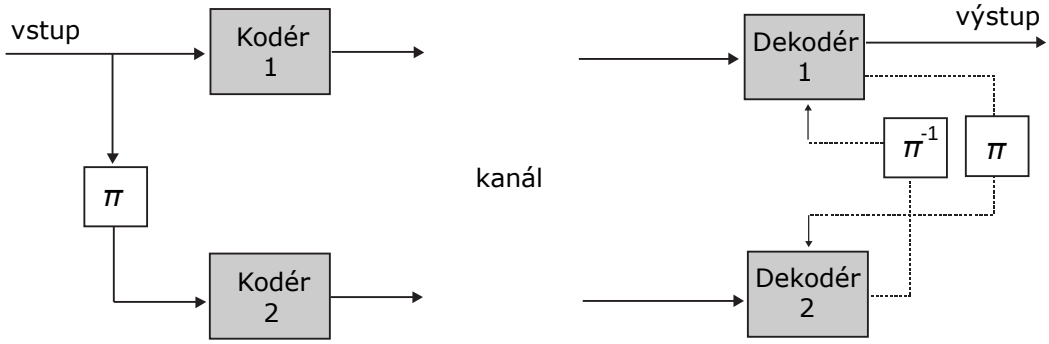


Obr. 10. Obecné schéma sériového zřetězení

Celkový kódový poměr při sériovém zřetězení dvou kódů je dán vztahem [22]:

$$R_c = \frac{\kappa_1 \kappa_2}{n_1 n_2}, \tag{27}$$

což odpovídá součinu dvou dílčích kódových poměrů.



Obr. 11. Obecné schéma paralelního zřetězení

Celkový kódový poměr při paralelním zřetězení dvou kódů je dán vztahem [22]:

$$R_c = \frac{\kappa}{n_1 + n_2} \tag{28}$$

Při paralelním i sériové zřetězení je často mezi kodéry vkládán tzv. *prokladač* nebo též *blok prokladu* (*interleaver*), označený symbolem π a *inverzní* (*zpětný) prokladač* (*deinterleaver*), označený symbolem π^{-1}. Účelem prokladače je záměrná změna pořadí bitů originální vstupní sekvence, která jednak zajišťuje velkou váhu výstupní sekvence a zároveň napomáhá ke korekci shluků chyb. Při dekódování se na výstupu inverzního prokladače objevují tyto shluky jako rozptýlené chyby, které jsou snáze opravitelné. Základní typy prokladačů jsou uvedeny v kapitole (kap. 3.4)

3 TURBO-KÓDY

3.1 Úvod

Nároky na vysokou míru zabezpečení byly před uvedením turbo-kódů řešeny prostřednictvím konvolučních kódů s velkou délkou kódového omezení. Takové konvoluční kódy však mají velkou vnitřní paměť, a následné dekódováni je komplikované. *Turbo-kódy,* které v roce 1993 představil Berrou, Glavieux a Thitimajshim [1], znamenaly průlom v teorii kódování. V originálu byly představeny na RSC kodérech s délkou kódového omezení $K = 5$ doplněných blokem prokladu s pamětí 65536 bitů. Výsledky prezentovaných simulací ukázaly, že chybovosti BER[1] 10^{-5} jsou tyto kódy schopny dosáhnout při poměru E_b / N_0 pohybujícím se kolem 0,7 dB. Tyto výsledky byly nejprve přijímány se značnou skepsí, jakmile však byly opakovaně ověřeny jinými výzkumníky, byla již vědecká komunita o mimořádných vlastnostech turbo-kódů přesvědčena.

Od roku 1993 se tyto kódy staly předmětem intenzivního výzkumu a vývoje, hned na úvod je tedy třeba poznamenat, že pod označením „turbo-kódy“ se dnes skrývá široká škála kódů, z nichž mohou být jmenovány např. *turbo product kódy* nebo turbo-kódy založené na sériovém zřetězení. Následující kapitoly budou věnovány „původním“ a zároveň nejpoužívanějším turbo-kódům, jejichž základ tvoří paralelní zřetězení dvou modulů. V procesu kódování představují tyto moduly RSC kodéry, na straně dekódování jsou to pak *SISO* (*Soft-Input Soft-Output*) dekodéry. Dekódovací algoritmus pracuje iterativním způsobem, během kterého se v cyklu zpracovávají přijatá data. Právě podobnost tohoto iterativního procesu s funkcí turbodmychadla spalovacích motorů byla inspirací pro název těchto kódů.

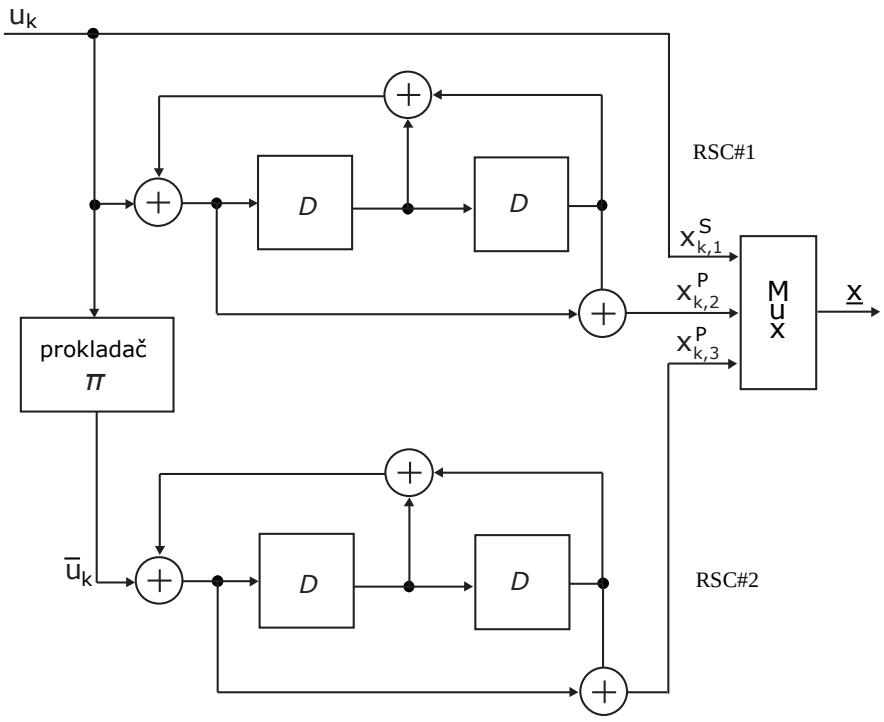
3.2 Kódování turbo-kódů

Základní struktura turbo-kodéru je znázorněna na obrázku (Obr. 12). Kodér tvoří dva paralelně zapojené RSC kodéry, které jsou obvykle identické. Oba kodéry přijímají stejná

[1] Bitová chybovost *BER* (*Bit Error Rate*) je definována poměrem chybně přijatých bitů ku celkovému počtu přijatých bitů za určitou dobu měření [26].

data, do druhého (spodního) kodéru však tyto data vstupují v obměněném pořadí, které určuje prokladač. Praktická realizace prokladače vyžaduje jeho pevně danou konstrukci, z tohoto důvodu pracuje prokladač nad pevně danými bloky dat a turbo-kódy se tak řadí mezi lineární blokové kódy. Valenti [23] se ve své práci odkazuje na třídu tzv. *stream oriented* turbo-kódů, které nepracují na blokovém principu, ty však nebudou předmětem zájmu tohoto textu.

Schéma kodéru na obrázku (Obr. 12) obsahuje dva dílčí $\mathrm{RSC}[7,5]_8$ kodéry (dále jen RSC#1 a RSC#2). Vstupní bity jsou označeny symbolem u_k, blok prokladu symbolem π a proložená data vstupující do druhého dekodéru symbolem $\overline{u}_k$. Systematické informační bity $x_{k,1}^S$ jsou vzaty pouze z horního kodéru RSC#1 a paritní bity $x_{k,2}^P$ a $x_{k,3}^P$ z výstupu obou kodérů.



Obr. 12. Příklad turbo-kodéru

Všechny tři výstupy jsou multiplexovány do výstupní sekvence $\underline{x} = (x_{0,1}^S, x_{0,2}^P, x_{0,3}^P, ..., x_{N-1,1}^S, x_{N-1,2}^P, x_{N-1,3}^P)$. Celkový kódový poměr kodéru R_c je určen vztahem:

$$\frac{1}{R_c} = \frac{1}{R_1} + \frac{1}{R_2} - 1, \tag{29}$$

V tomto konkrétním případě jsou kódové poměry dílčích kodérů $R_1 = R_2 = \frac{1}{2}$ a celkový kódový poměr je $R_c = \left(\frac{1}{0{,}5} + \frac{1}{0{,}5} - 1 \right)^{-1} = \frac{1}{3}$. U konvolučních kódů je často využívaná technika tzv. *punkturingu,* kterou lze zvýšit kódový poměr. Jakým způsobem toho lze dosáhnout u paralelně zřetězených turbo-kódů, bude stručně popsáno v kapitole (kap. 3.3).

3.2.1 Efektivní volná vzdálenost

Klíčovou vlastností, která ovlivňuje schopnost kódu detekovat a opravovat chyby, je *Hammingova kódová vzdálenost*. Kódová vzdálenost dvou slov je definována jako počet odlišných znaků v těchto slovech. Pro lineární kódy je *minimální kódová vzdálenost* mezi dvěma kódovými slovy rovna nejmenší *Hammingově váze w*, tedy počtu nenulových složek.

Výsledná váha výstupní sekvence turbo-kodéru je dána příspěvky obou RSC kodérů. Povaha turbo-kódů je taková, že výstup z druhého kodéru nemá v ideálním případě stejnou váhu, jako výstup kodéru prvního, bez znalosti tohoto výstupu tedy není praktické provádět výpočet Hammingovy vzdálenosti. Pro účely turbo-kódů je proto zavedena tzv. *efektivní volná vzdálenost* $d_{free,eff}$, která je definována jako nejmenší Hammingova váha nenulové kódové sekvence RSC kódu, která je vyvolaná vstupní sekvencí s váhou $w = 2$. Hodnotu efektivní volné vzdálenosti je možné pro turbo-kódy považovat za ekvivalentní hodnotě Hammingovy vzdálenosti.

Údaje v tabulce (Tab. 1), které jsou převzaty z [13], obsahují seznam optimálních typů RSC kodérů a jejich hodnot $d_{free,eff}$, které jsou vhodné pro návrh turbo-kódů s kódovým poměrem $R_c = \frac{1}{3}$.

Tab. 1. Optimální RSC kodéry pro turbo-kódy s $R_c = 1/3$

m	g_1	g_2	$d_{free,eff}$
2	7	5	10
3	15 (13)	17	14
4	37	21	10
5	43	55	30

3.3 Punkturované turbo-kódy

Punkturing je proces, během kterého jsou systematicky vypouštěny určité bity, které se objevují na výstupech kodéru. Trellis diagram konvolučních kódů s kódovým poměrem $R = \frac{\kappa}{n}$ má v každém uzlu 2^{κ} možných cest, ze kterých se během dekódování vybírá ta nejpravděpodobnější. Tato vlastnost vede k exponenciálnímu nárůstu složitosti dekódování s rostoucím parametrem κ. Technika punkturingu byla zavedena za účelem snížení počtu početních operací během dekódování (jednodušší uspořádání trellis diagramu) a splnění požadavků mnohých aplikací, které vyžadují vyšší kódový poměr.

Chceme-li zvýšit celkový kódový poměr původního turbo-kódu z $R_c = \frac{1}{3}$ na $R_c' = \frac{1}{2}$, můžeme z původní sekvence $\underline{x} = (x_{0,1}^S, x_{0,2}^P, x_{0,3}^P, x_{1,1}^S, x_{1,2}^P, x_{1,3}^P, x_{2,1}^S, x_{2,2}^P, x_{2,3}^P, ...)$ odstranit paritní bity kodéru RSC#1 v lichých časových okamžicích a paritní bity kodéru RSC#2 v sudých časových okamžicích (pozn.: uvažujeme počáteční čas *k* = 0). Vyslaná sekvence pak bude mít tvar $\underline{x} = (x_{0,1}^S, x_{0,2}^P, x_{1,1}^S, x_{1,3}^P, x_{2,1}^S, x_{2,2}^P, x_{3,1}^S, x_{3,3}^P ...)$. Operace vypuštění určitých bitů může být též popsána prostřednictvím tzv. *punkturovací matice P*. Tato matice má rozměry $n \times p_p$, kde p_p je perioda punkturování. Bity *l*-tého výstupu jsou přenášeny, pokud je hodnota v *l*-tém řádku matice „1“ a punkturovány pokud je „0“. Pro zmíněný příklad má punkturovací matice tvar:

$$P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{30}$$

3.4 Prokladače

Prokládání π je proces přeuspořádání datové sekvence, který napomáhá ke korekci shluků chyb. Shluky chyb, které se vyskytnou na proložené sekvenci, např. vlivem úniku signálu, jsou operací inverzního prokládání π^{-1} rozptýleny na jednoduché chyby, které jsou pak snáze opravitelné. V architektuře turbo-kódů plní prokladač několik dalších základních funkcí, které uvádí [24]:

1. Spojení dvou krátkých kódů s nižší mírou zabezpečení do jednoho dlouhého kódu s vyšší mírou zabezpečení.

2. Provedení dekorelace vstupních dat obou kodérů, čehož je využito při dekódování, které je založeno na výměně nekorelované informace mezi SISO dekodéry. Chyby, které nejsou opraveny v prvním dekodéru se přes prokladač mohou šířit tak, aby byly opraveny v druhém dekodéru.
3. Zajištění velké váhy kódových slov (výstupní sekvence) nebo snížení počtu kódových slov s malou váhou a zvýšení volné vzdálenosti kódových slov.

Uvažujme indexování pozic v prokladači od hodnoty 0 po hodnotu *N*-1. Obecně lze operaci prokládání označit jako vzájemně jednoznačné zobrazení sekvence přirozených čísel. Toto zobrazení je definováno tzv. mapovací funkcí [10]:

$$\pi : I \to I, \quad pro \quad I = \{0,1,2,...,N-1\}, \tag{31}$$

kde *N* reprezentuje délku prokládané datové sekvence. Na přijímací straně se provádí inverzní prokládání, které je definováno mapovací funkcí [10]:

$$\pi^{-1} : I \to I, \quad kde \quad \pi^{-1}(\pi(i)) = i, \quad pro \quad \forall i \in I = \{0,1,2,...,N-1\}. \tag{32}$$

Vzhledem k existenci velkého množství prokládacích technik budou v následujících podkapitolách krátce popsány pouze základní typy prokladačů.

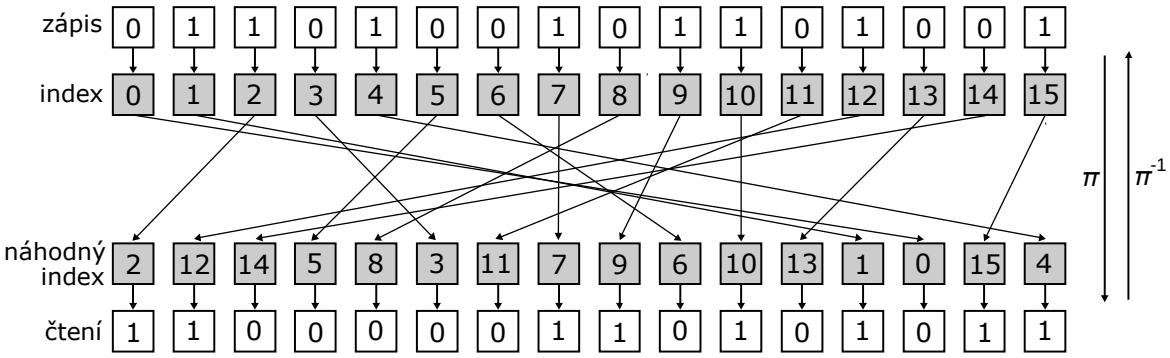
3.4.1 Uniformní náhodný prokladač

Blok vstupních dat o délce *N* je postupně zapsán do prokladače a čten je na základě pseudonáhodné sekvence indexů. Mapovací funkci prokladače $\pi(i)$ lze popsat v *N* krocích, které uvádí [13]:

krok 1. Výběr náhodného přirozeného čísla i_1 z množiny $I = \{0,1,2,...,N-1\}$ v rámci rovnoměrného rozdělení mezi hodnotou 0 a *N*-1 s pravděpodobností $P(i_1) = \frac{1}{N}$. Zvolená hodnota i_1 odpovídá $\pi(0)$.

krok *k* až *N*. Pro $k>1$ jsou s pravděpodobností $P(i_k) = \frac{1}{N-(k+1)}$ generována náhodná přirozená čísla i_k z podmnožiny $I_k = \{i \in I, i \neq i_1,...i_{k-1}\}$. Mapovací funkce prokladače $\pi(k)$ je nastavena na příslušné hodnoty i_k. V kroku $k=N$ je poslední zbývající hodnota i_N přidělena $\pi(N-1)$.

Na obrázku (Obr. 13) je příklad náhodného uniformního prokladače s délkou $N = 16$. Je možné si všimnout, že některé sousední vstupní bity jsou v proložené sekvenci umístěny opět vedle sebe. Pravděpodobnost výskytu tohoto nepříznivého jevu se snižuje s rostoucí velikostí *N*.



Obr. 13. Náhodný uniformní prokladač

3.4.2 Neuniformní (Berrou-Glavieux) náhodný prokladač

Neuniformní náhodný prokladač byl použit autory turbo-kódů v originálním návrhu kódovacího konceptu [1] a následně popsán v [14]. Data jsou zapsána po řádcích do čtvercové matice o rozměru $R \times R$ a čtena jsou z pseudonáhodně určených indexů s nerovnoměrným rozdělením pravděpodobnosti. Indexace probíhá podle vztahů [14]:

$$i_r = \left(\frac{R}{2} + 1\right) \cdot (i + j) \bmod R \tag{33}$$

$$\xi = (i + j) \bmod L \tag{34}$$

$$j_r = \left[P(\xi) \cdot (j + 1)\right] - 1 \bmod R \tag{35}$$

kde indexy *i* a *j* určují adresu řádku a sloupce pro zápis a indexy i_r a j_r odpovídají adrese řádku a sloupce pro čtení. Konstanta *L* je malé celé číslo, které je voleno v závislosti na velikosti *R*. Koeficient $\xi = 0, 1, ...L - 1$ a funkce $P(\xi)$ určují prvočíslo nesoudělné s hodnotou *R*. Pro typickou velikost prokladače *N* = 65536, kde *R* = 256 a *L* = 8, mohou být použity hodnoty $P(\xi)$ uvedené tabulce (Tab. 2).

Tab. 2. Hodnoty $P(\xi)$ pro neuniformní prokladač s L=8

$P(0) = 17$	$P(2) = 19$	$P(4) = 41$	$P(6) = 13$
$P(1) = 37$	$P(3) = 29$	$P(5) = 23$	$P(7) = 7$

3.4.3 Semi-random (polo-náhodný) prokladač

Semi-random, zkráceně *s-random*, nebo také *polo-náhodný prokladač*, je jakýmsi mezistupněm mezi náhodnými prokladači a prokladači s pevně danou strukturou prokládání. Zejména u vstupních sekvencí s malou délkou *N* nemůže náhodný prokladač zaručit, že vstupní sekvence bude „rozbita“ ideálním způsobem. Polo-náhodný prokladač popsaný v [12] se snaží optimálním způsobem změnit pořadí bitů vstupní sekvence tak, aby sousední vstupní bity byly od sebe v proložené sekvenci vzdáleny vždy o jistou garantovanou hodnotu >*S*. Algoritmus, podle kterého prokladač pracuje, je následující:

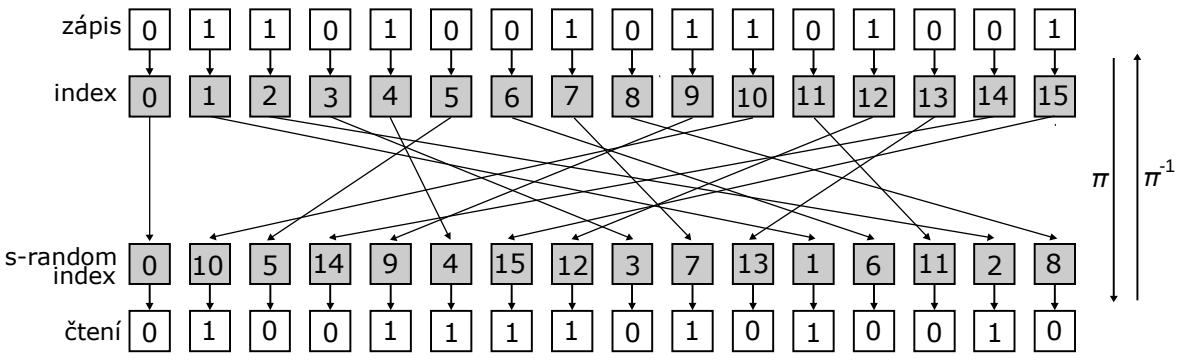
krok 1. Je zvoleno přirozené číslo $S < \sqrt{\frac{N}{2}}$.

krok 2. Je generován náhodný index $i_k \in I = \{0,1,2,...N-1\}$.

krok 3. Index i_k je porovnán s *S* předchozími vygenerovanými indexy $\{i_{k-1},...,i_{k-S}\}$. Pokud je vzdálenost i_k v rozmezí $\pm S$ od předchozích indexů, je zamítnut a následuje krok 2. V opačném případě je index i_k „vyjmut“ z *I* a použit.

krok 4. Kroky 2. a 3. jsou opakovány, dokud není využito všech *N* indexů.

Je zřejmé, že s rostoucím parametrem *S* roste i časová složitost algoritmu a není zajištěno, že bude úspěšně dokončen. Simulačně bylo ověřeno, že právě pro hodnoty $S < \sqrt{\frac{N}{2}}$ algoritmus konverguje ke správnému řešení, a to v přijatelném čase. Na obrázku (Obr. 14) je příklad polo-náhodného prokladače s $N = 16$ a $S = 2$. Je možné si všimnout, že *S* sousedních bitů vstupní sekvence je po proložení od sebe vzdáleno vždy o hodnotu >S.



Obr. 14. Semi-random prokladač

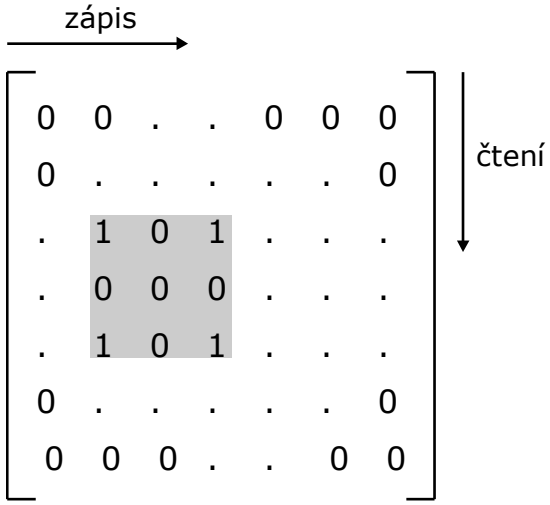
3.4.4 Blokové prokladače

Obdélníkový blokový prokladač

Blokový (*obdélníkový*) *prokladač* tvoří jednoduchá maticová struktura s rozměry $R \times C$. Vstupní data jsou zapisována po řádcích *R* a čtena jsou po sloupcích *C*, tak jak je znázorněno na obrázku (Obr. 15). Délka prokládané sekvence je $N = R \cdot C$ a pro mapovací funkci platí [1]

$$\pi(i) = (i \bmod R) \cdot C + \left\lfloor \frac{i}{R} \right\rfloor \qquad pro \quad \forall i \in I = \{0,1,2,...,R \cdot C - 1\} \tag{36}$$

Z obrázku (Obr. 15) je dále patrné, že tento typ prokladače není vhodný pro prokládání některých vstupních sekvencí s nízkou váhou. Briffa [11] uvádí příklad, kdy je vstupní sekvence s váhou *w* = 4 uložena tak, že hodnota 1 je zapsána v rozích submatice, která má rozměry odpovídající násobku periody *p* impulsní odezvy RSC kodéru. V takových případech má sekvence paritních bitů obou RSC kodérů nízkou váhu a je tím celkově zhoršen výkon turbo-kódu.



Obr. 15. Obdélníkový blokový prokladač

[1] Funkce $\lfloor x \rfloor$ určuje dolní celou část čísla *x* (nejbližší menší celé číslo). Anglicky – *Floor(x).*

Diagonální blokový prokladač

Diagonální prokladač je podobný obdélníkovému prokladači s rozměrem $R \times C$. Rozdíl je v tom, že vstupní data jsou zapisována po řádcích *R* a *C* symbolů je vždy čteno diagonálně zleva doprava. Délka prokládané sekvence je $N = R \cdot C$ a pro mapovací funkci platí:

$$\pi(i) = (i \cdot C) \bmod N + \left(\left\lfloor \frac{i}{R} \right\rfloor + i \bmod R \right) \bmod C \quad pro \quad \forall i \in I = \{0,1,2,...,R \cdot C - 1\} \quad (37)$$

Blokový prokladač - sudý-lichý

Tento blokový prokladač je navržen pro punkturované turbo-kódy s kódovým poměrem $R_c = \frac{1}{2}$. K mapovací funkci obdélníkového prokladače je přidána podmínka [13]:

$$(\pi(i) + i) \bmod 2 = 0 \quad pro \quad \forall i \in I = \{0,1,2,...,R \cdot C - 1\}, \quad (38)$$

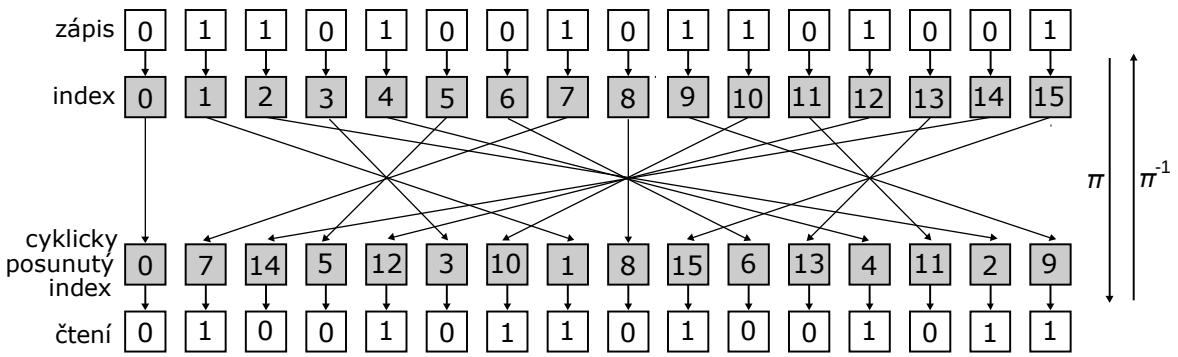
která zajistí, že informační bity na lichých pozicích jsou mapovány opět na liché pozice a bity na sudých pozicích jsou mapovány na sudé pozice. Aby byla podmínka (38) splněna, je nutné, aby oba rozměry matice, tedy *R* a *C*, měly lichou hodnotu.

3.4.5 Prokladač s cyklickým posuvem

Dalším typem, který stejně jako blokové prokladače nevyužívá náhodnosti, je prokladač s cyklickým posuvem. Pro jeho mapovací funkci platí vztah [12]:

$$\pi(i) = (a \cdot i + r) \bmod N \quad pro \quad \forall i \in I = \{0,1,2,...,N - 1\}, \quad (39)$$

kde $r < N$ je offset a proměnná $a < N$ je velikost kroku posuvu, jejíž hodnota musí být nesoudělná s délkou prokládané sekvence *N*. Na obrázku (Obr. 16) je prokladač s cyklickým posuvem, kde offset $r = 0$, $a = 7$ a $N = 16$.



Obr. 16. Prokladač s cyklickým posuvem

Z obrázku (Obr. 16) je patrné, že dva sousední indexy původní sekvence jsou v proložené sekvenci vzdáleny vždy o 7 nebo 9 pozic. [22] uvádí, že pro tento typ prokladače byly prokázány dobré výsledky zejména tehdy, má-li vstupní sekvence váhu $w = 2$, kdy proložená sekvence dokáže zajistit velkou váhu výstupní sekvence paritních bitů. Vzhledem k pravidelné vzdálenosti sousedních vstupních bitů o 9 nebo 7 pozic, však může být pro vstupní sekvence s vyšší váhou splnění těchto požadavků komplikovanější.

3.4.6 Jiné typy prokladačů

Hledání co nejefektivnější implementace prokladače je stále aktuálním tématem, a kromě uvedených typů byla navržena celá řada jiných. Může být jmenován např. *Code Matched Interleaver* [13], který je přizpůsoben stavbě RSC kodérů a výběr indexů provádí takovým způsobem, aby eliminoval všechny výstupní sekvence způsobující malou váhu výstupního slova, *Chaotic Interleaver*, *Interleaver Design with Simulated Annealing* [11], *Deterministic Interleaver* [15], *Correctly-Terminating Interleaver* [11] aj.

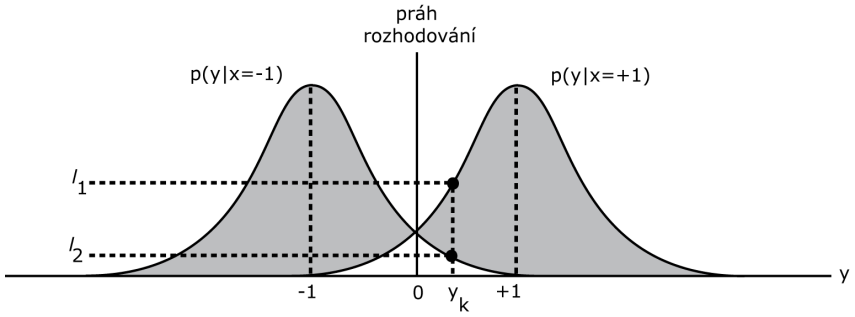
4 DEKÓDOVÁNÍ TURBO-KÓDŮ

4.1 Úvod

Budeme-li uvažovat AWGN model přenosového kanálu, je rozložení náhodných poruch dáno Gaussovou distribucí. Přijatý (demodulovaný) signál pak nemá podobu diskrétního binárního signálu, připomíná však spojitý náhodný signál. Tvrdé porovnání takového signálu s určitou prahovou úrovní může vést ke špatnému vyhodnocení logické úrovně signálu a následné dekódování nemusí být efektivní. Aby byly při procesu dekódování využity všechny informace obsažené v signálu (skutečná velikost, korelace mezi symboly), je třeba využívat tzv. měkkých rozhodnutí, tj. hodnot a-posteriorních pravděpodobností. Informace získané měkkým rozhodováním lze pak dodávat z výstupu jednoho dílčího dekodéru na vstup dekodéru druhého a tento proces iterativně opakovat, čímž je docíleno postupné korekce chyb. [2] uvádí dva základní algoritmy, které se v praxi používají k dekódování turbo-kódů. Prvním z nich je Viterbiho algoritmus *SOVA* (*Soft Output Viterbi Algorithm*). Druhým je iterativní algoritmus *MAP (Maximum A-Posteriori)*, který se označuje také jako dekodér s měkkým vstupem a výstupem *SISO (Soft-Input Soft-Output)*. Druhý jmenovaný algoritmus bude předmětem zájmu této části práce.

4.1.1 Rozhodování podle maximální věrohodnosti

Obrázek (Obr. 17) ukazuje dvě funkce hustoty pravděpodobnosti náhodné veličiny *y* podmíněné hodnotou veličiny *x* při přenosu přes AWGN kanál.



Obr. 17. Věrohodnostní funkce

Tyto funkce jsou označovány jako tzv. *věrohodnostní funkce* (*likelihood functions*). Veličina *y* odpovídá hodnotě užitečného signálu doplněné náhodným bílým Gaussovským šumem, symboly l_1 a l_2 jsou hodnoty věrohodnostních funkcí pro konkrétní přijatý symbol

y_k a proměnná *x* odpovídá hodnotě přenášeného bitu, o kterém je nutné rozhodnout.

Z důvodu lepšího vyjádření některých matematických vztahů je zavedena konvence, že logické hodnoty *x* = 1 a *x* = 0 jsou vyjádřeny prostřednictvím polárních hodnot *x* = +1 a *x* = -1. Rozhodovací pravidlo známé jako metoda *maximální věrohodnosti* (*ML*) volí hodnotu *x* = +1, pokud je $l_1 > l_2$ nebo hodnotu *x* = -1, pokud je $l_2 > l_1$.

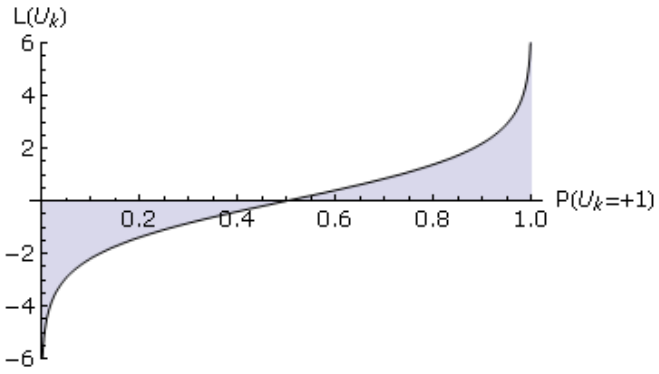
Na straně dekodéru se teorie kanálového kódování zajímá především o pravděpodobnost $P(u_k = \pm 1)$ datového bitu u_k, která je podmíněna přijatou sekvencí $\underline{y}$. Pro tento druhý případ existuje podobné rozhodovací pravidlo, které je popsáno v následující podkapitole.

4.1.2 Log-Věrohodnostní poměr

Logaritmický věrohodnostní poměr LLR (*Log-Likelihood Ratio*) datového bitu u_k je označen jako $L(u_k)$ a definován jako přirozený logaritmus poměru pravděpodobností datového bitu, který může nabývat dvou hodnot:

$$L(u_k) = \ln\left[\frac{P(u_k = +1)}{P(u_k = -1)}\right] \tag{40}$$

Obrázek (Obr. 18) ukazuje, jak se mění hodnota $L(u_k)$ v závislosti na pravděpodobnosti $P(u_k = +1)$. Znaménko $L(u_k)$ indikuje, zda bit u_k nabývá pravděpodobněji hodnoty +1 nebo -1. Hodnota $L(u_k)$ pak říká, s jakou jistotou odpovídá hodnota bitu u_k hodnotě udávané znaménkem $L(u_k)$. V případě, že $L(u_k) \approx 0$ je $P(u_k = +1) \approx P(u_k = -1) \approx 0{,}5$ a nemůže být rozhodnuto o hodnotě u_k. Analogicky, pokud je $L(u_k) >> 0$, je $P(u_k = +1) >> P(u_k = -1)$ a může být rozhodnout, že hodnota bitu $u_k = +1$. [16]



Obr. 18. LLR funkce

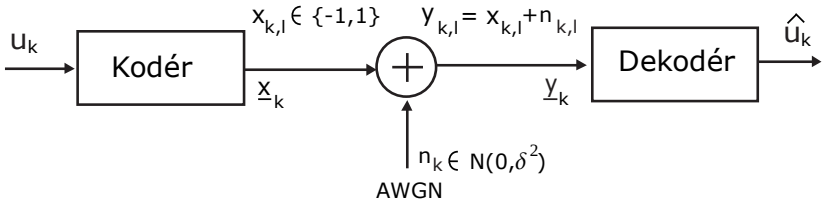
Stejně tak, jako je k výpočtu LLR $L(u_k)$ potřeba nepodmíněných pravděpodobností $P(u_k = \pm 1)$, je možné provádět výpočet LLR založený na podmíněných pravděpodobnostech. LLR $L(u_k \mid \underline{y})$, je definován jako:

$$L(u_k \mid \underline{y}) = \ln \left[\frac{P(u_k = +1 \mid \underline{y})}{P(u_k = -1 \mid \underline{y})} \right], \tag{41}$$

kde u_k je testovaný datový bit a $\underline{y}$ je přijatá sekvence. Podmíněná pravděpodobnost $P(u_k = \pm 1 \mid \underline{y})$ je známa jako a-posteriorní pravděpodobnost dekódovaného bitu u_k. Právě výpočet této hodnoty je hlavním cílem SISO dekodéru, který je využit při dekódování turbo-kódů.

4.1.3 Schéma komunikačního systému

Pro vysvětlení principu dekódovacího algoritmu a sjednocení symboliky, která bude využita v následujícím textu, je na obrázku (Obr. 19) uvedeno zjednodušené blokové schéma komunikačního systému. Je uvažováno, že hodnota log1 odpovídá napětí +1 V a log0 odpovídá napětí -1V (pozn.: skutečné mapování na symboly {+1,-1} je prováděno až v modulátoru, který není ve schématu zakreslen.).



Obr. 19. Zjednodušené blokové schéma komunikačního systému

Symbolem u_k je označen datový bit vstupní sekvence $u = (u_0, u_1, u_2, .., u_{N-1})$ vstupující do kodéru v čase *k*. Výstupní kódový blok složený z bitů $x_{k,l}$ je označena jako $\underline{x}_k = (x_{k,1}, x_{k,2}, ..., x_{k,n})$, kde $l = 1,2,...,n$ odpovídá počtu výstupů kodéru. Zakódovaná data vstupují do diskrétního Gaussovského kanálu bez paměti, na jehož výstupu je přijatá *n*-tice $\underline{y}_k = (y_{k,1}, y_{k,2}, ..., y_{k,n})$, jejíž jednotlivé symboly jsou dány součtem $y_{k,l} = x_{k,l} + n_{k,l}$. Symbol $n_{k,l}$ odpovídá hodnotě šumového signálu s rozptylem σ^2 a střední hodnotou 0.

4.2 Maximum A-Posteriori algoritmus

Jak uvádí [16], algoritmus známý jako *Maximum A-Posteriori* (*MAP*) navrhnul v roce 1974 Bahl, Cocke, Jelínek a Raviv. Byl navržen pro odhad a-posteriorní pravděpodobnosti stavů a přechodů pozorovaného *Markovova řetězce*[1]. Tento algoritmus se stal známý také jako *BCJR algoritmus*, pojmenovaný po jeho objevitelích, kteří ukázali, že může být s výhodou použit pro dekódování blokových a konvolučních kódů. Při použití u konvolučních kódů je optimální co do minimalizace dekódovacího procesu, kdy je minimalizována pravděpodobnost výběru nesprávné cesty skrz trellis diagram. I přes tuto skutečnost nenalezl širšího využití, neboť zkoumá každou možnou cestu skrz trellis diagram, což se podepsalo na jeho výkonnosti. Na výsluní se MAP algoritmus dostal až po objevu turbo-kódů. Výhodou tohoto algoritmu je, že poskytne nejen odhadovanou bitovou sekvenci, ale také pravděpodobnost s jakou byl každý bit dekódován správně. Tato vlastnost se stala velmi důležitou právě pro iterativní dekódování turbo-kódů, a proto našel MAP algoritmus uplatnění v klíčové práci Berrou a kol. [1]. S uplynulou dobou bylo vynaloženo mnoho úsilí pro redukci složitosti MAP algoritmu na rozumnou úroveň.

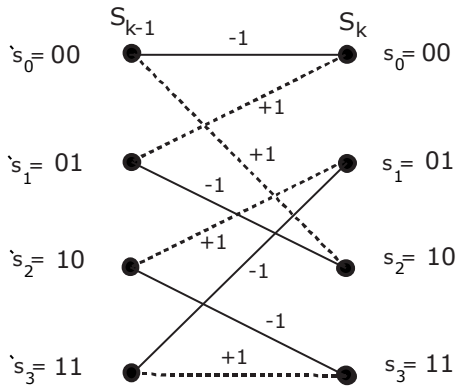
V této kapitole je popsán princip MAP algoritmu podle [16], který je využíván pro měkké rozhodování, jež bylo nastíněno v úvodu této kapitoly.

4.2.1 Odvození základního vztahu pro LLR

Na obrázku (Obr. 20) je zobrazen trellis diagram $\mathrm{RSC}[7,5]_8$ kodéru s K=3. Pro tento kodér existují čtyři stavy, v jakých se může nacházet, a v každém tomto stavu jsou možné dva přechody v závislosti na hodnotě vstupního bitu. Jeden z těchto přechodů je zobrazen plnou čarou s hodnotou vstupního bitu -1, druhý je vyznačen přerušovanou čarou a odpovídá hodnotě vstupního bitu +1. Z obrázku (Obr. 20) je dále patrné, že jsme schopni určit, jaká byla hodnota bitu, která způsobila přechod mezi dvěma různými stavy, za

[1] Markovův řetězec označuje stochastický (náhodný či pravděpodobnostní) proces, který má Markovovskou vlastnost. Ta říká, že v každém stavu procesu je pravděpodobnost navštívení dalších stavů nezávislá na dříve navštívených stavech. To znamená, že chování v Markovových řetězcích je „bezpaměťové“: V každém konkrétním stavu je možno zapomenout historii (posloupnost stavů předcházející stavu současnému). [25]

předpokladu, že známe předešlý stav $S_{k-1} = `s$ a současný stav $S_k = s$. Pravděpodobnost, s jakou nabývá bit $u_k$ hodnoty -1, pak odpovídá pravděpodobnosti, že přechod ze stavu $S_{k-1} = `s$ do stavu $S_k = s$ je jeden ze čtyř možných přechodů, které odpovídají hodnotě -1. Jelikož mohl nastat pouze jeden z těchto přechodů a vzájemně se tedy vylučují, je pravděpodobnost toho, že se některý z nich vyskytl, rovna součtu jejich individuálních pravděpodobností.



Obr. 20. Možné přechody v $RSC[7,5]_8$ kodéru

MAP algoritmus přiřazuje každému dekódovanému bitu u_k pravděpodobnost, že tento bit má hodnotu +1 nebo -1. Tento proces je totožný s hledáním LLR, který byl vysvětlen v kapitole (kap. 4.1.2). Vztah (41) je možné s využitím Bayesova teorému (11) upravit na tvar:

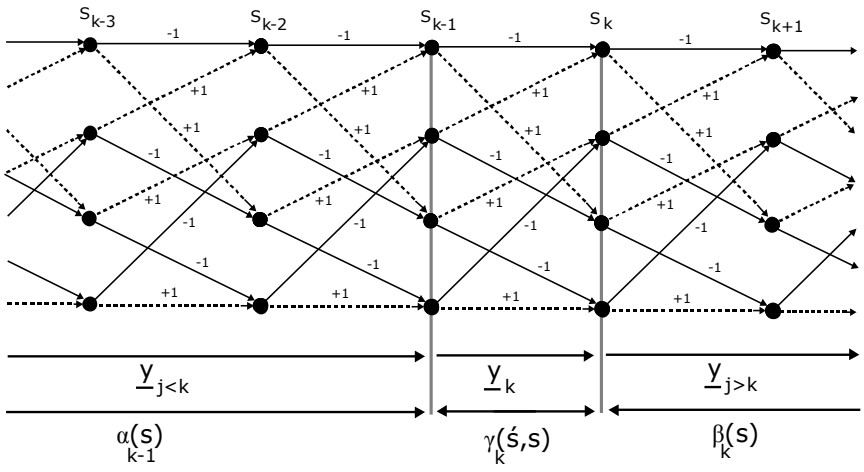
$$\begin{aligned} L(u_k \mid \underline{y}) &= \ln\left[\frac{P(u_k = +1 \cap \underline{y}) / P(\underline{y})}{P(u_k = -1 \cap \underline{y}) / P(\underline{y})}\right] \\ &= \ln\left[\frac{P(u_k = +1 \cap \underline{y})}{P(u_k = -1 \cap \underline{y})}\right] \end{aligned} \tag{42}$$

Další úpravou, kdy pravděpodobnosti $P(u_k = +1 \cap \underline{y})$ a $P(u_k = -1 \cap \underline{y})$ nahradíme součtem pravděpodobností výskytu daného přechodu, získáme vztah:

$$L(u_k \mid \underline{y}) = \ln\left[\frac{\sum\limits_{\substack{(`s,s) \Rightarrow \\ u_k = +1}} P(`s \cap s \cap \underline{y})}{\sum\limits_{\substack{(`s,s) \Rightarrow \\ u_k = -1}} P(`s \cap s \cap \underline{y})}\right], \tag{43}$$

kde $(\grave{s}, s) \Rightarrow u_k \pm 1$ jsou soubory přechodů z předchozího stavu $S_{k-1} = \grave{s}$ do nového stavu $S_k = s$, které mohou nastat v případě vstupního bitu $u_k = \pm 1$.

Přijatá sekvence bitů $\underline{y}$ může být rozdělena do tří oddělených sekcí: přijatá sekvence $\underline{y}_k$ odpovídající současnému přechodu, přijatá sekvence $\underline{y}_{j<k}$ před současným přechodem a přijatá sekvence $\underline{y}_{j>k}$ následující po současném přechodu. Toto rozdělení je zobrazeno na následujícím obrázku (Obr. 21).



Obr. 21. Trellis $[7,5]_8$ *kódu využívaný Map dekodérem*

Za předpokladu, že přenosový kanál je bez paměti, bude následující přijatá sekvence $\underline{y}_{j>k}$ záviset pouze na současném stavu s a nikoliv na předešlém stavu $\grave{s}$ nebo současné a předešlé přijaté sekvenci $\underline{y}_k$ a $\underline{y}_{j<k}$. Při použití Bayesova teorému (11) je možné hodnotu $P(\grave{s} \cap s \cap \underline{y})$ rozdělit na tři části:

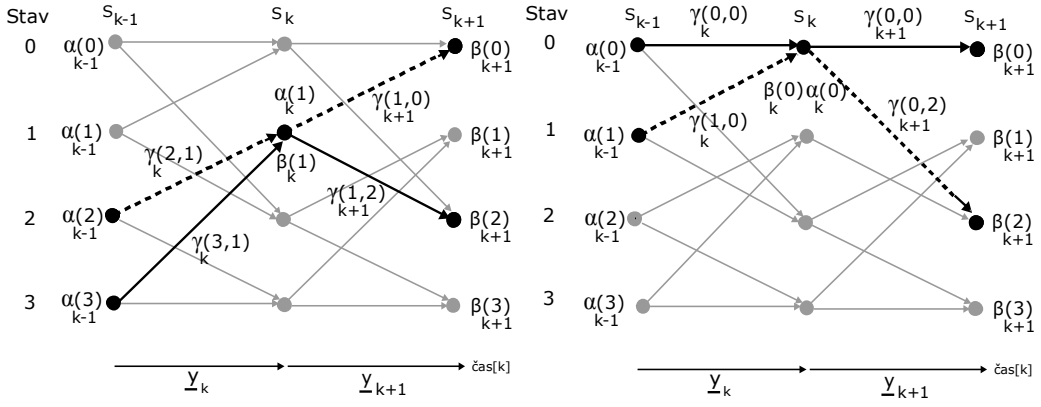
$$P(\grave{s} \cap s \cap \underline{y}) = \underbrace{P(\grave{s} \cap \underline{y}_{j<k})}_{\alpha_{k-1}(\grave{s})} \cdot \underbrace{P(\underline{y}_{j>k} \mid s)}_{\beta_k(s)} \cdot \underbrace{P(\{\underline{y}_k \cap s\} \mid \grave{s})}_{\gamma_k(\grave{s},s)}, \qquad (44)$$

Kompletní odvození vztahu (44) uvádí např. [5], [16], [17].

Hodnota $\alpha_{k-1}(\grave{s})$ označovaná jako *dopředná metrika*, je pravděpodobnost toho, že trellis diagram je ve stavu $\grave{s}$ a přijatá sekvence je do této chvíle $\underline{y}_{j<k}$. Hodnota $\beta_k(s)$ označovaná jako *zpětná metrika*, je pravděpodobnost toho, že diagram je v čase k ve stavu s a následná přijatá sekvence bude $\underline{y}_{j>k}$. Konečně hodnota $\gamma_k(\grave{s}, s)$ označovaná jako

metrika přechodu, je pravděpodobnost toho, že diagram, který je v čase *k*-1 ve stavu $\grave{s}$, přejde do stavu *s*, zatímco přijatá sekvence odpovídá $\underline{y}_k$.

Obrázek (Obr. 22) znázorňuje význam těchto tří pravděpodobností při přechodu ze stavu $S_{k-1} = \grave{s}$ do stavu $S_k = s$. Úkolem MAP algoritmu je najít všechny hodnoty $\alpha_k(s)$ a $\beta_k(s)$ pro všechny možné stavy *s* v celé mřížce a všechny hodnoty $\gamma_k(\grave{s}, s)$ pro přechody mezi stavy $S_{k-1} = \grave{s}$ a $S_k = s$ v čase $k = 0, 1, \ldots, N\text{-}1$.



Obr. 22. Výpočet hodnot $\alpha_k(s)$, $\beta_k(s)$, $\gamma_k(\grave{s}, s)$

Vypočítané hodnoty slouží pro hledání pravděpodobnosti $P(S_{k-1} = \grave{s} \cap S_k = s \cap \underline{y})$, která je potřebná pro výpočet LRR $L(u_k \mid \underline{y})$ jednotlivých bitů u_k.

4.2.2 Výpočet dopředné metriky

Z hodnoty $\alpha_{k-1}(\grave{s})$ ve vztahu (44), lze $\alpha_k(s)$ vyjádřit jako [16]:

$$\begin{aligned} \alpha_k(s) &= P(S_k = s \cap \underline{y}_{j<k+1}) \\ &= \sum_{All\ \grave{s}} P(s \cap \grave{s} \cap \underline{y}_{j<k} \cap \underline{y}_k) \\ &= \sum_{All\ \grave{s}} P(\{s \cap \underline{y}_k\} \mid \grave{s}) \cdot P(\grave{s} \cap \underline{y}_{j<k}), \\ &= \sum_{All\ \grave{s}} \gamma_k(\grave{s}, s) \cdot \alpha_{k-1}(\grave{s}) \end{aligned} \tag{45}$$

Hodnota $\gamma_k(\grave{s}, s)$ je známa (viz dále) a metrika $\alpha_k(s)$ je vypočítána rekurzivně z inicializačních hodnot: $\alpha_0(s) = \begin{cases} 1 & pro\ s = 0 \\ 0 & pro\ \forall s \neq 0 \end{cases}$.

Příklad výpočtu hodnot $\alpha_k(0)$ a $\alpha_k(1)$ je na obrázku (Obr. 22), kde:

$$\alpha_k(0) = \alpha_{k-1}(0) \cdot \gamma_k(0,0) + \alpha_{k-1}(1) \cdot \gamma_k(0,2)$$

$$\alpha_k(1) = \alpha_{k-1}(2) \cdot \gamma_k(2,1) + \alpha_{k-1}(3) \cdot \gamma_k(3,1)$$

[17] uvádí způsob normalizace hodnoty $\alpha_k(s)$, kterou je nutné provést v každém výpočetním cyklu. Tato normalizace je nutná z důvodu numerické stability výpočtu a je dána vztahem:

$$\alpha_k(s) = \frac{\alpha_k(s)}{\sum_{All\ s} \alpha_k(s)}. \tag{46}$$

4.2.3 Výpočet zpětné metriky

Z hodnoty $\beta_k(s)$ (44), lze $\beta_{k-1}(`s)$ vyjádřit jako [16]:

$$\begin{aligned} \beta_{k-1}(`s) &= P(\underline{y}_{j>k-1} \mid S_{k-1} = `s) \\ &= \sum_{All\ s} P(\{\underline{y}_{j>k-1} \cap s\} \mid `s) \\ &= \sum_{All\ s} P(\underline{y}_{j<k} \mid \{`s \cap s \cap \underline{y}_k\}) \cdot P(\{\underline{y}_k \cap s\} \mid `s)' \\ &= \sum_{All\ s} \beta_k(s) \cdot \gamma_k(`s, s) \end{aligned} \tag{47}$$

Hodnota $\gamma_k(`s, s)$ je známa a metrika $\beta_{k-1}(s)$ je vypočítána rekurzivně použitím hodnoty $\beta_k(s)$. V posledním kroku $k = N$-1, neexistuje žádná budoucí přijatá sekvence, proto je zde nastavení inicializačních hodnot poněkud složitější, než v případě metriky $\alpha_0(s)$. Berrou a kol. [1] uvádí ve své práci následující inicializační podmínku: $\beta_N(s) = \begin{cases} 1 & pro\ s = 0 \\ 0 & pro\ \forall s \neq 0 \end{cases}$. Jiné varianty jsou uvedeny např. v [16], [23].

Příklad výpočtu hodnot $\beta_k(0)$ a $\beta_k(1)$ je na obrázku (Obr. 22), kde:

$$\beta_k(0) = \beta_{k+1}(0) \cdot \gamma_k(0,0) + \beta_{k+1}(2) \cdot \gamma_k(0,2)$$

$$\beta_k(1) = \beta_{k+1}(0) \cdot \gamma_k(1,0) + \beta_{k+1}(2) \cdot \gamma_k(1,2)$$

Podobně, jako u hodnoty $\alpha_k(s)$, uvádí [17] normalizaci pro metriku $\beta_k(s)$:

$$\beta_k(s) = \frac{\beta_k(s)}{\sum_{All\ s} \beta_k(s)}. \tag{48}$$

4.2.4 Výpočet metriky přechodu

Následné odvození vztahu pro metriku $\gamma_k(`s, s)$ uvádí [16], [18]. A-prioriní pravděpodobnost přechodu ze stavu `s do *s* $P(`s \cap s)$ odpovídá hodnotě $P(u_k)$, kde $u_k$ je přijatý bit korespondující s přechodem ze stavu `s do *s*. Vztah pro $\gamma_k(`s, s)$ vyjádřený v (44), lze při využití Bayesova teorému (11) upravit na tvar:

$$\begin{aligned}\gamma_k(`s, s) &= P(\{\underline{y}_k \cap s\} \mid `s) \\ &= P(\underline{y}_k \mid \{`s \cap s\}) \cdot P(u_k)\end{aligned}. \tag{49}$$

Hodnotu $P(u_k)$ je možné vyjádřit z úvodní rovnice (40). Uvažujeme-li $P(u_k = -1) = 1 - P(u_k = +1)$, můžeme rovnici (40) přepsat do tvaru:

$$e^{L(u_k)} = \frac{P(u_k = +1)}{1 - P(u_k = +1)},$$

$$P(u_k = +1) = \frac{e^{L(u_k)}}{1 + e^{L(u_k)}} = \frac{1}{1 + e^{-L(u_k)}},$$

$$P(u_k = -1) = \frac{1}{1 + e^{+L(u_k)}} = \frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}}.$$

Z předchozích úprav tedy vyplývá:

$$\begin{aligned}P(u_k = \pm 1) &= \frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}} \cdot e^{u_k L(u_k)/2} \\ &= C_{1k} \cdot e^{(u_k L(u_k)/2)}\end{aligned}. \tag{50}$$

Zlomek $C_{1k} = e^{-L(u_k)/2} / 1 + e^{-L(u_k)}$ není závislý na tom, zda hodnota u_k je +1 nebo -1, ale pouze na hodnotě *LLR* $L(u_k)$. Druhý člen rovnice (49) $P(\{\underline{y}_k \mid \{`s \cap s\})$ je ekvivalentní s $P(\underline{y}_k \mid \underline{x}_k)$, kde hodnota $\underline{x}_k$ odpovídá vyslané bitové sekvenci spojené s přechodem z `s do *s*. Vyslanou a přijatou sekvenci $\underline{x}_k$ a $\underline{y}_k$ tvoří *n*-tice jednotlivých bitů $x_{k,l} = x_{k,1}, x_{k,2}, \ldots x_{k,n}$ a $y_{k,l} = y_{k,1}, y_{k,2}, \ldots y_{k,n}$ pro $l = 1,2,\ldots,n$.

V případě, že bity $x_{k,l}$ byly přenášeny přes AWGN kanál, při využití *BPKS*[1], lze psát:

$$P(\underline{y}_k \mid \underline{x}_k) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{E_b}{2\sigma^2} \left(y_{k,l} - a x_{k,l} \right)^2 \right], \tag{51}$$

kde E_b je energie signálu na 1 bit, σ^2 je rozptyl šumu a *a* je amplituda úniku, která pro AWGN kanál bez úniku nabývá hodnoty *a* = 1.

Pravděpodobnost $P(\underline{y}_k \mid \{`s \cap s\}) = P(\underline{y}_k \mid \underline{x}_k)$, že *n* přijatých bitů je určeno *n*-ticí vyslaných bitů, odpovídá součinu jejich individuálních pravděpodobností $P(y_{k,l} \mid x_{k,l})$. Druhý člen rovnice (49) můžeme být podle (1) vyjádřen jako:

$$P(\underline{y}_k \mid \{`s \cap s\}) = \prod_{l=1}^{n} P(y_{k,l} \mid x_{k,l}) . \tag{52}$$

Spojením (51) a (52) dostáváme:

$$\begin{aligned} P(\underline{y}_k \mid \{`s \cap s\}) &= \prod_{l=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{E_b}{2\sigma^2} \left(y_{k,l} - a x_{k,l} \right)^2 \right] \\ &= \left(\frac{1}{\sqrt{2\pi\sigma}} \right)^n \exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^{n} y_{k,l}^2 \right) \exp\left(-a^2 \frac{E_b}{2\sigma^2} \sum_{l=1}^{n} x_{k,l}^2 \right) \exp\left(2a \frac{E_b}{2\sigma^2} \sum_{l=1}^{n} y_{k,l} \cdot x_{k,l} \right) \\ &= C_{2k} \cdot \exp\left(2a \frac{E_b}{2\sigma^2} \sum_{l=1}^{n} y_{k,l} \cdot x_{k,l} \right) . \end{aligned} \tag{53}$$

Součin označený jako $C_{2k} = \left(\frac{1}{\sqrt{2\pi\sigma}} \right)^n \exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^{n} y_{k,l}^2 \right) \exp\left(-a^2 \frac{E_b}{2\sigma^2} \sum_{l=1}^{n} x_{k,l}^2 \right)$ nezávisí na hodnotě u_k, ani na znaménku, či kódovém slovu $\underline{x}_k$. První část součinu je závislá pouze na přenosovém kanálu, druhá část závisí na kanálu a sekvenci $\underline{y}_k$ a třetí část, kde $\sum x_{k,l}^2 = n$, závisí pouze na kanálu a amplitudě úniku. To znamená, že prvek C_{2k} bude mít stejnou hodnotu jak v čitateli, tak ve jmenovateli rovnice (43) a tím bude vyrušen.

---

[1] U modulací *PKS*, tj. u modulací s klíčováním fázovým posuvem resp. zdvihem, datový binární signál ovlivňuje fázi nosné vlny, přičemž její amplituda zůstává konstantní. U nejjednodušší dvojstavové modulace *BPKS* nabývá fáze dva diskrétní stavy, např. 0° a 180°. [2]

Vlastnosti kanálu lze popsat pomocí tzv. *spolehlivosti kanálu* L_C. Tato hodnota je závislá na odstupu signálu od šumu a amplitudě úniku. Výsledný vztah pro (49) lze spojením (50) a (53) přepsat na tvar:

$$\begin{aligned}\gamma_k(\grave{s},s) &= C_{2k}\exp\left(\frac{L_C}{2}\sum_{l=1}^{n} y_{k,l}\cdot x_{k,l}\right)\cdot C_{1k}\cdot e^{(u_k L(u_k)/2)} \\ &= C_k\cdot e^{(u_k L(u_k)/2)}\exp\left(\frac{L_C}{2}\sum_{l=1}^{n} y_{k,l}\cdot x_{k,l}\right)\end{aligned}, \tag{54}$$

kde hodnota $L_C = 4a\frac{E_b}{2\sigma^2}$ a $C_k = C_{1k}\cdot C_{2k}$. Hodnota C_k nezávisí na u_k, ani na znaménku či kódovém slovu $\underline{x}_k$, je tedy konstantní, a bude ve finálním vztahu pro *LLR* vyrušena.

Spojením rovnic (43) a (44) dostáváme vztah:

$$\begin{aligned}L(u_k\mid\underline{y}) &= \ln\left[\frac{\sum_{\substack{(\grave{s},s)\Rightarrow\\ u_k=+1}} P(S_{k-1}=\grave{s}\cap S_k=s\cap\underline{y})}{\sum_{\substack{(\grave{s},s)\Rightarrow\\ u_k=-1}} P(S_{k-1}=\grave{s}\cap S_k=s\cap\underline{y})}\right] \\ &= \ln\left[\frac{\sum_{\substack{(\grave{s},s)\Rightarrow\\ u_k=+1}} \beta_k(s)\cdot\gamma_k(\grave{s},s)\cdot\alpha_{k-1}(\grave{s})}{\sum_{\substack{(\grave{s},s)\Rightarrow\\ u_k=-1}} \beta_k(s)\cdot\gamma_k(\grave{s},s)\cdot\alpha_{k-1}(\grave{s})}\right]\end{aligned}. \tag{55}$$

Tento vztah, vyjadřující logaritmický věrohodnostní poměr LLR $L(u_k\mid\underline{y})$ je právě tím, co počítá a dodává MAP dekodér.

4.3 Princip Iterativního dekódování

Pro využití MAP algoritmu při iterativním dekódování zavádí Berrou a kol. [1] koncept tzv. *extrinsické* (vnější) *informace*. Následně je uvažován systematický kód, kde jeden z *n* přenášených bitů je systematický informační bit u_k. Uvažujeme-li, že tento systematický bit u_k je první v *n*-bitové sekvenci, je jeho vyslaná verze (spojená s daným přechodem z $\grave{s}$ do *s*) označena jako $x_{k,1}^S$ a přijatá verze jako $y_{k,1}^S$. Zbývající bity $x_{k,l}^P$ a $y_{k,l}^P$ v dané sekvenci jsou paritní. Seznam možných přechodů ze stavu $\grave{s}$ do *s*, a jim odpovídající paritní bity $x_{k,l}^P$ při daném vstupním bitu u_k resp. $x_{k,1}^S$ pro RSC$[7,5]_8$ kodér, je uveden v následující tabulce (Tab. 3).

Tab. 3. Možné přechody a odpovídající systematické a paritní bity pro $RSC[7,5]_8$

u_k	`s	s	$x_{k,1}^S$	$x_{k,l}^P$
-1	0	0	-1	-1
-1	1	2	-1	-1
-1	2	3	-1	+1
-1	3	1	-1	+1
+1	0	2	+1	+1
+1	1	0	+1	+1
+1	2	1	+1	-1
+1	3	3	+1	-1

[16] a [18] uvádí rozšíření vztahu (54) do tvaru:

$$\begin{aligned}\gamma_k(\grave{s},s) &= C_k \cdot e^{(u_k L(u_k)/2)} \exp\left(\frac{L_C}{2}\sum_{l=1}^{n} y_{k,l}\cdot x_{k,l}\right)\\ &= C_k \cdot e^{(u_k L(u_k)/2)} \exp\left[\frac{L_C}{2}\left(y_{k,1}^S \cdot x_{k,1}^S + \sum_{l=2}^{n} y_{k,l}^P \cdot x_{k,l}^P\right)\right],\\ &= C_k \cdot e^{\frac{u_k}{2}\left[L(u_k)+L_C y_{k,1}^S\right]} \cdot \chi_k(\grave{s},s)\end{aligned} \tag{56}$$

kde nová proměnná $\chi_k(\grave{s},s)$ zastupuje výraz:

$$\chi_k(\grave{s},s) = \exp\left(\frac{L_C}{2}\sum_{l=2}^{n} y_{k,l}^P \cdot x_{k,l}^P\right), \tag{57}$$

Vztah (55) lze nyní upravit na tvar:

$$\begin{aligned}L(u_k \mid \underline{y}) &= \ln\left[\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=+1}} \beta_k(s)\cdot\gamma_k(\grave{s},s)\cdot\alpha_{k-1}(\grave{s})}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=-1}} \beta_k(s)\cdot\gamma_k(\grave{s},s)\cdot\alpha_{k-1}(\grave{s})}\right]\\ &= \ln\left[\frac{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=+1}} C_k \cdot e^{\frac{u_k}{2}\left[L(u_k)+L_C y_{k1}^S\right]}\cdot\chi_k(\grave{s},s)\beta_k(s)\cdot\alpha_{k-1}(\grave{s})}{\sum\limits_{\substack{(\grave{s},s)\Rightarrow\\ u_k=-1}} C_k \cdot e^{\frac{u_k}{2}\left[L(u_k)+L_C y_{k1}^S\right]}\cdot\chi_k(\grave{s},s)\cdot\beta_k(s)\cdot\alpha_{k-1}(\grave{s})}\right],\end{aligned} \tag{58}$$

kde bit u_k nabývá v čitateli vždy hodnoty +1 a ve jmenovateli -1.

Matematickou úpravou lze výraz (58) zjednodušit do tvaru:

$$\begin{aligned} L(u_k \mid \underline{y}) &= \ln\left[e^{L(u_k)} e^{L_C y_{k,1}^S} \cdot \frac{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \chi_k(\grave{s},s)\beta_k(s)\cdot\alpha_{k-1}(\grave{s})}{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \chi_k(\grave{s},s)\cdot\beta_k(s)\cdot\alpha_{k-1}(\grave{s})} \right] \\ &= L(u_k) + L_C y_{k,1}^S + \ln\left[\frac{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \chi_k(\grave{s},s)\beta_k(s)\cdot\alpha_{k-1}(\grave{s})}{\sum_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \chi_k(\grave{s},s)\cdot\beta_k(s)\cdot\alpha_{k-1}(\grave{s})} \right], \end{aligned} \tag{59}$$

Po zastoupení extrinsické informace, která je obsažena v logaritmu, ve třetí části výrazu (59), proměnou $L_e(u_k)$, je možné psát:

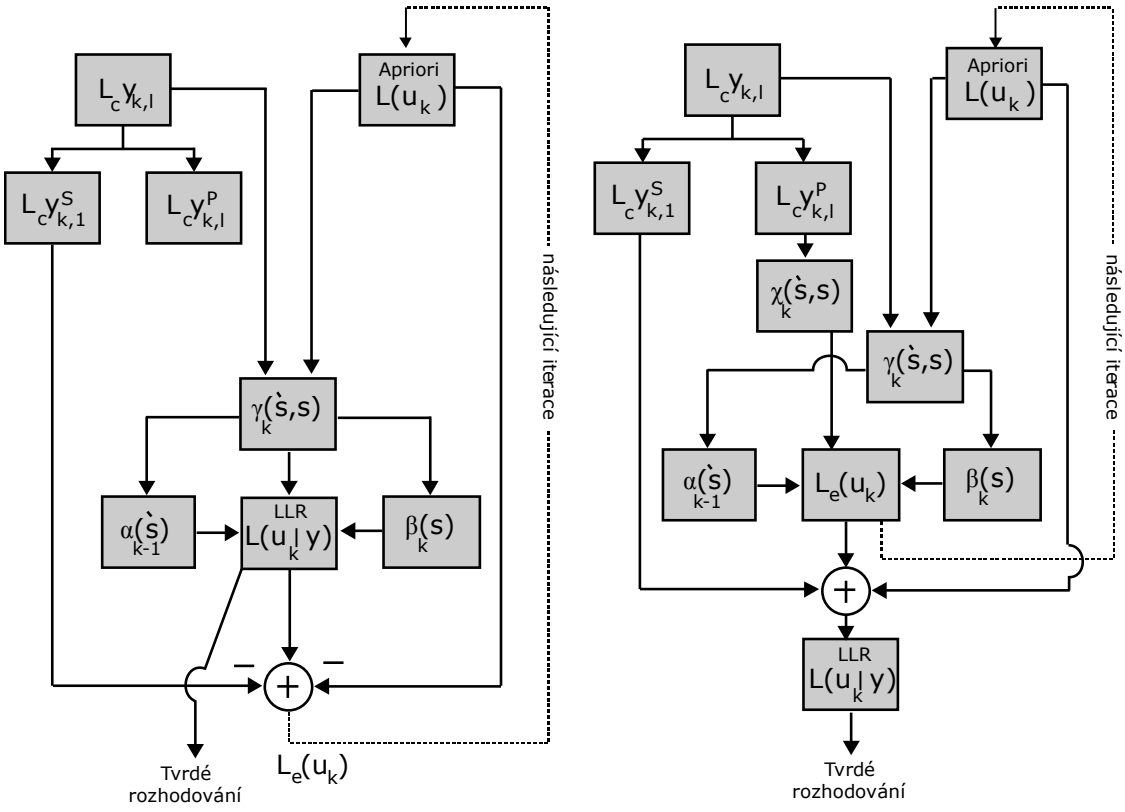
$$L(u_k \mid \underline{y}) = L(u_k) + L_C y_{k1}^S + L_e(u_k), \tag{60}$$

$$L_e(u_k) = L(u_k \mid \underline{y}) - L(u_k) - L_C y_{k1}^S, \tag{61}$$

Souhrn:

- $L(u_k)$ je a-priorní LLR informačních bitů u_k.
- $L(u_k \mid \underline{y})$ je a-posteriorní LLR informačních bitů u_k podmíněný přijatou sekvencí $\underline{y}$.
- $L_C y_{k,1}^S$ odpovídá měkkému výstupu kanálu pro systematický bit $y_{k,1}^S$. Vliv $L_C y_{k,1}^S$ na výpočet $L(u_k \mid \underline{y})$ je přímo úměrný spolehlivosti kanálu L_C.
- $L_e(u_k)$ je extrinsická informace, kterou zprostředkovává a využívá dekodér pro každou následující iteraci. Extrinsická informace je zpětnou vazbou na vstup druhého MAP dekodéru, kde bude sloužit jako zpřesněná a-priori pravděpodobnost pro další iteraci. Tato proměnná tedy obsahuje znalosti získané v dekódovacím procesu a z MAP dekodéru je získána odečtením a-priorní informace $L(u_k)$ a měkkého výstupu kanálu pro daný, přenesený systematický bit $L_C y_{k1}$ od a-posteriorní LLR $L(u_k \mid \underline{y})$. Alternativní možnost výpočtu vychází ze znalosti $\chi_k(\grave{s},s)$, $\beta_k(s)$ a $\alpha_{k-1}(\grave{s})$ tak, jak je uvedeno ve třetí části rovnice (59).

Následující diagramy (Obr. 23) zjednodušeně zachycují dvě možnosti výpočtu extrinsické informace $L_e(u_k)$, a to prozatím bez zavedení kooperace dvou dílčích MAP dekodérů a s pouhým nástinem iterativního procesu. Konstrukce a princip kompletního dekodéru turbo-kódu bude vysvětlena v následující kapitole (kap. 4.4).



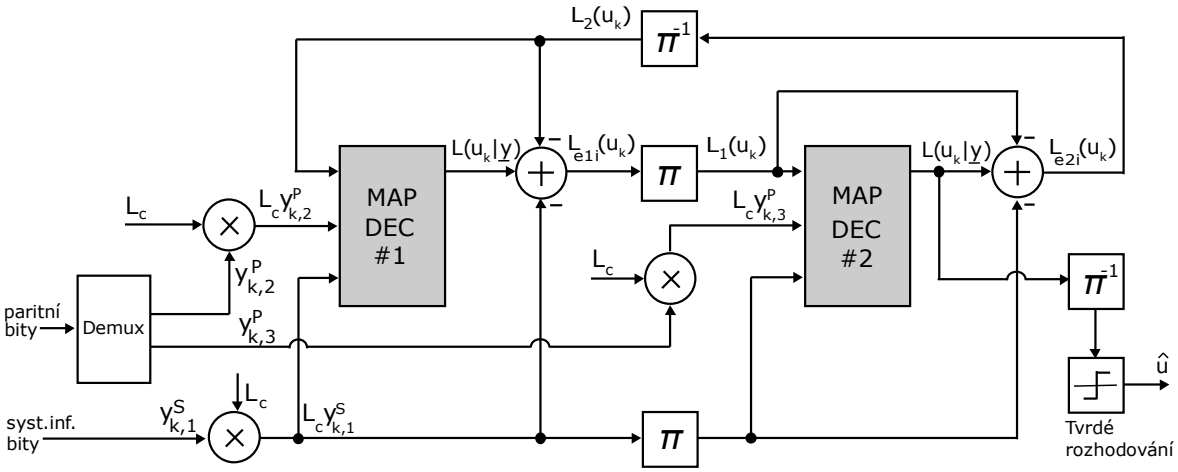
Obr. 23. Vývojové diagramy pro výpočet $L(u_k \mid \underline{y})$ *a* $L_e(u_k)$

(Obr. 23a) - Metriky $\gamma_k(\acute{s},s)$, $\alpha_{k-1}(\acute{s})$ a $\beta_k(s)$ jsou vypočítány podle vztahů (54), (47) a (45). Rozdílem uvedeným ve vztahu (61) je vypočítána extrinsická informace $L_e(u_k)$. Hodnota $L(u_k \mid \underline{y})$ je počítána podle vztahu (55) při každé iteraci - tento způsob bude také podrobně rozepsán v následující kapitole.

(Obr. 23b) - Nejprve je vypočítána hodnota $\chi_k(\acute{s},s)$ podle vztahu (57). Metriky $\gamma_k(\acute{s},s)$, $\alpha_{k-1}(\acute{s})$ a $\beta_k(s)$ jsou vypočítány podle vztahů (54), (47) a (45). Extrinsická informace $L_e(u_k)$ je vypočítána podle logaritmu (třetí části) ve vztahu (59) a finální hodnota $L(u_k \mid \underline{y})$ je počítána jen jednou, a to po skončení iteračního procesu podle vztahu (60).

4.4 Dekodér turbo-kódu

Architektura dekodéru, kterou uvádí [23], je zachycena na obrázku (Obr. 24). Přítomnost zpětné vazby naznačuje, že dekodér funguje iterativním způsobem. Jedna celá iterace se skládá ze dvou polovičních iterací, kdy je každá určena pro zpracování příslušné složky zakódované sekvence. Načasování je takové, že první dekodér (dále jen DEC#1) je činný v první polovině iterace a DEC#2 působí během druhé poloviny. Jak říká vztah (54), pro výpočet hodnoty $\gamma_k(`s, s)$ jsou využívány hodnoty $x_{k,1}^S$ a $x_{k,l}^P$, dílčí dekodéry musí mít proto plnou znalost o struktuře (trellis diagramu) příslušného kodéru a v tabulkách si udržovat obecné informace o hodnotách $x_{k,l}^S$ a jim příslušných $x_{k,l}^P$ pro všechny možné přechody $`s, s$, viz tabulka (Tab. 3). Po následujícím stručném popisu jedné iterace je uveden pseudokód dekódovacího algoritmu.



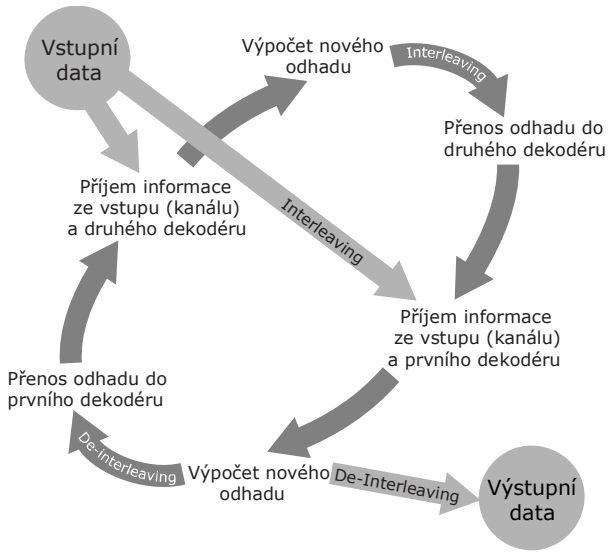
Obr. 24. Schéma turbo-dekodéru

Do dekodéru vstupuje přijatá kanálová sekvence $L_C \underline{y}$, která je rozdělena na systematické informační bity $L_C y_{k,1}^S$, paritní bity $L_C y_{k,2}^P$ z RSC#1 a paritní bity $L_C y_{k,3}^P$ z RSC#2. V případě, že paritní bity byly punkturovány, turbo-dekodér musí nastavit hodnotu 0 pro příslušné vypuštěné paritní bity $L_C y_{kl}^P$. Výstupem MAP dekodérů je aposteriorní LLR $L(u_k \mid \underline{y})$, ze kterého je vypočítána extrinsická informace $L_{edi}(u_k)$, kde index d = 1,2 označuje dekodér (DEC#1 nebo DEC#2), který tuto informaci vypočítal a index i=0,1,2,…,I je číslo aktuální iterace. Informace $L_{edi}(u_k)$ vypočítána jedním dekodérem je po operaci přímého nebo inverzního prokládání, použita jako a-priorní pravděpodobnost $L_d(u_k)$ na vstupu druhého dekodéru.

V průběhu první iterace vstupuje do DEC#1 $L_C y_{k,1}^S$ a $L_C y_{k,2}^P$, tento dekodér však nemá k dispozici hodnotu $L_{e20}(u_k)$, ta je proto nastavena na 0, což odpovídá a-priorní pravděpodobnosti 0,5, viz obrázek (Obr. 18). Výstupem DEC#1 je příslušný LLR, ze kterého je podle vztahu (61) extrahována informace $L_{e11}(u_k)$.

Po dokončení první poloviny iterace vstupuje do DEC#2 prokládaná verze informačních bitů $L_C y_{k,1}^S$ a paritní bity $L_C y_{k,3}^P$ z kodéru RSC#2. V tuto chvíli použije DEC#2 proloženou hodnotu $L_{e11}(u_k)$, resp. $L_1(u_k)$, která byla poskytnuta z DEC#1. Výstupem DEC#2 je nová hodnota a-posteriorního LLR, ze kterého je extrahována informace $L_{e21}(u_k)$. První iterace je dokončena inverzním proložením $L_{e21}(u_k)$ a odesláním této hodnoty na DEC#1.

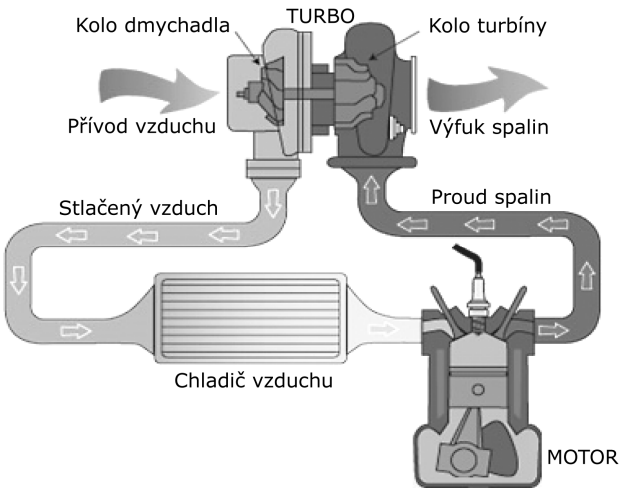
Po doběhnutí poslední iterace je výstupní hodnota a-posteriorního LLR $L(u_k \mid \underline{y})$ z DEC#2 inverzně proložena a pro jednotlivé bity tvrdě porovnána s prahovou úrovní 0,0, na základě čehož je rozhodnuto o hodnotě bitu u_k = -1 nebo +1, resp. log0 nebo log1. Celou situaci tohoto dekódovacího procesu zjednodušeně zachycuje obrázek (Obr. 25).



Obr. 25. Základní princip turbo-dekodéru

S rostoucím počtem iterací v průměru klesá chybovost BER dekódovaných bitů. Velikost tohoto poklesu se postupně snižuje, proto je výhodné stanovit pevný maximální počet iterací, obvykle I<10, kdy již další iterační kroky nepřináší výrazné navýšení výkonu. Možná je také varianta, kdy pevná hodnota I není nastavena, a iterační proces je ukončen jiným vhodným kritériem na základě vnitřních hodnot dekodéru.

Na tomto místě je pro zajímavost uveden obrázek (Obr. 26), který může posloužit pro přiblížení analogie mezi turbo-dekodérem a spalovacím motorem s turbodmychadlem, jež posloužil autorům turbo-kódů jako inspirace pro název těchto kódů.



Obr. 26. Motor s turbodmychadlem

4.4.1 Pseudokód iterativního MAP dekódování

$L_{e20}(u_k) = 0 \quad pro \quad k = 0, 1, \ldots, N\text{-}1.$

Pro stanovený počet iterací $i = 1: I$ proveď následující

Pro oba dekodéry DEC#1 a DEC#2 proveď kroky **1-5**

1. **Metrika přechodu**

   Pro všechny bity $k = 0 : N\text{-}1$

   Pro všechny přechody z `s do s

   If DEC#1 $\gamma_k(\grave{s}, s) = \exp\left[\frac{1}{2} x_{k1}^S L_2(u_k) + \frac{L_C}{2}\left(y_{k,1}^S \cdot x_{k,1}^S + y_{k,2}^P \cdot x_{k,2}^P\right)\right],$

   kde hodnota $L_2(u_k)$ odpovídá $L_{e2i-1}(u_k)$ od DEC#2 po operaci inverzního prokládání.

   If DEC#2 operace prokládání na hodnotě $y_{k,1}^S$.

   $$\gamma_k(\grave{s}, s) = \exp\left[\frac{1}{2} x_{k1}^S L_1(u_k) + \frac{L_C}{2}\left(y_{k,1}^S \cdot x_{k,1}^S + y_{k,3}^P \cdot x_{k,3}^P\right)\right],$$

   kde hodnota $L_1(u_k)$ odpovídá $L_{e1i}(u_k)$ od DEC#1 po operaci prokládání.

2. **Dopředná metrika**

$$\alpha_0(s) = \begin{cases} 1 & pro\ s = 0 \\ 0 & pro\ \forall s \neq 0 \end{cases}$$

Pro všechny bity $k = 1 : N$

Pro všechny přechody z $\grave{s}$ do s

$$\alpha_k(s) = \sum_{All\ \grave{s}} \gamma_k(\grave{s}, s) \cdot \alpha_{k-1}(\grave{s})$$

Normování hodnoty $\alpha_k(s)$

3. Zpětná metrika

If DEC#1 $\quad \beta_N(s) = \begin{cases} 1 & pro\ s = 0 \\ 0 & pro\ \forall s \neq 0 \end{cases}$

If DEC#2 $\quad \beta_N(s) = \dfrac{1}{2^M} \quad pro\ \forall s,$

kde M odpovídá velikosti paměti RSC kodéru, viz vztah (16).

Pro všechny bity $k = N : 1$

Pro všechny přechody z $\grave{s}$ do s

$$\beta_{k-1}(\grave{s}) = \sum_{All\ s} \beta_k(s) \cdot \gamma_k(\grave{s}, s)$$

Normování hodnoty $\beta_{k-1}(\grave{s})$.

4. Výpočet LLR

Pro všechny bity $k = 0 : N\text{-}1$

Pro všechny přechody z $\grave{s}$ do s

$$L(u_k \mid \underline{y}) = \ln \left[\frac{\sum_{(\grave{s},s) \Rightarrow u_k = +1} \beta_k(s) \cdot \gamma_k(\grave{s}, s) \cdot \alpha_{k-1}(\grave{s})}{\sum_{(\grave{s},s) \Rightarrow u_k = -1} \beta_k(s) \cdot \gamma_k(\grave{s}, s) \cdot \alpha_{k-1}(\grave{s})} \right],$$

kde $(\grave{s}, s) \Rightarrow u_k = +1$ je sada přechodů při vstupním bitu $u_k = 1$ a $(\grave{s}, s) \Rightarrow u_k = -1$ je sada přechodů při vstupním bitu $u_k = -1$.

5. Extrakce extrinsické informace z LLR

Pro všechny bity $k = 0 : N\text{-}1$

If DEC#1 $\quad L_{e1i}(u_k) = L(u_k \mid \underline{y}) - L_2(u_k) - L_C y_{k,1}^S$

If DEC#2 $\quad L_{e2i}(u_k) = L(u_k \mid \underline{y}) - L_1(u_k) - L_C y_{k,1}^S$

6. Tvrdé rozhodování

Operace inverzního prokládání na výsledné hodnotě $L(u_k \mid \underline{y})$

Pro všechny bity $k = 0 : N\text{-}1$

If $L(u_k \mid \underline{y}) > 0.0 \quad u_k = 1$

else $\quad u_k = 0$

4.5 Log-Map a Max-Log-MAP algoritmus

MAP algoritmus je z velké části založen na jednoduchých operacích násobení a sčítání. Tato vlastnost přináší nevýhodu při návrhu hardwarové implementace, neboť je potřeba začlenění složitých obvodů pro násobení reálných čísel.

Log-MAP algoritmus [18] je transformace MAP algoritmu, která vykazuje odpovídající výkonnost, aniž by nastaly problémy v praktickém provedení. Pracuje v logaritmické oblasti, kde je násobení převedeno na sčítání. Vyjádření jednotlivých metrik α_k^{LM}, β_k^{LM}, γ_k^{LM} Log-Map algoritmu je následující:

$$\alpha_k^{LM}(s) = \ln\left(\sum_{All\ `s} \gamma_k(`s,s)\cdot\alpha_{k-1}(`s)\right), \tag{62}$$

$$\beta_{k-1}^{LM}(`s) = \ln\left(\sum_{All\ s} \beta_k(s)\cdot\gamma_k(`s,s)\right), \tag{63}$$

$$\gamma_k^{LM}(`s,s) = \ln C_k + \frac{1}{2}u_k\cdot L(u_k) + \frac{L_C}{2}\sum_{l=1}^{n} y_{k,l}\cdot x_{k,l}\ . \tag{64}$$

A-posteriorní LLR $L(u_k \mid \underline{y})$ má tvar:

$$L(u_k \mid \underline{y}) = \ln\left(\frac{\sum_{\substack{(`s,s)\Rightarrow\\ u_k=+1}} \exp\left[\beta_k^{LM}(s)+\gamma_k^{LM}(`s,s)+\alpha_{k-1}^{LM}(`s)\right]}{\sum_{\substack{(`s,s)\Rightarrow\\ u_k=-1}} \exp\left[\beta_k^{LM}(s)+\gamma_k^{LM}(`s,s)+\alpha_{k-1}^{LM}(`s)\right]}\right). \tag{65}$$

Jak uvádí [19], rovnice (65) může být upravena využitím *Jacobian logaritmu* $\ln(e^x+e^y) = \max(x,y) + \ln\left[1+\exp(-|y-x|)\right]$ a za pomoci tabulek pro hodnocení korekční funkce $\ln\left[1+\exp(-|y-x|)\right]$. Touto transformací vznikne tzv. *Max-Log-MAP algoritmus (MLM)*, jehož složitost může být dále snížena omezením pouze na aproximaci $\ln(e^x+e^y) = \max(x,y)$. Toto zjednodušení má za následek zkreslení měkkého výstupu a degraduje výkon dekodéru, přesto je však Max-Log-MAP algoritmus, pro již zmíněné výhody, preferován pro hardwarovou realizaci dekodéru. Při použití Max-Log-MAP algoritmu je LLR pro jednotlivé informační systematické bity vyjádřen následujícím rozdílem:

$$L(u_k \mid \underline{y}) \approx \max_{\substack{(\grave{s},s)\Rightarrow \\ u_k=+1}} \left(\beta_k^{MLM}(s) + \gamma_k^{MLM}(\grave{s},s) + \alpha_{k-1}^{MLM}(\grave{s})\right) - \max_{\substack{(\grave{s},s)\Rightarrow \\ u_k=-1}} \left(\beta_k^{MLM}(s) + \gamma_k^{MLM}(\grave{s},s) + \alpha_{k-1}^{MLM}(\grave{s})\right). \tag{66}$$

Metriky α_k^{MLM} , β_k^{MLM} , γ_k^{MLM} jsou vypočítány podle rovnic:

$$\alpha_k^{MLM}(s) = \ln\left(\sum_{All\ \grave{s}} \exp\left[\gamma_k^{MLM}(\grave{s},s) + \alpha_{k-1}^{MLM}(\grave{s})\right]\right), \approx \max_{All\ \grave{s}} \left(\gamma_k^{MLM}(\grave{s},s) + \alpha_{k-1}^{MLM}(\grave{s})\right) \tag{67}$$

$$\beta_{k-1}^{MLM}(\grave{s}) = \ln\left(\sum_{All\ s} \exp\left[\gamma_k^{MLM}(\grave{s},s) + \beta_k^{MLM}(s)\right]\right), \approx \max_{All\ \grave{s}} \left(\gamma_k^{MLM}(\grave{s},s) + \beta_k^{MLM}(s)\right) \tag{68}$$

$$\gamma_k^{MLM}(\grave{s},s) = \ln\left[C_k \cdot e^{(u_k L(u_k)/2)} \exp\left(\frac{L_C}{2}\sum_{l=1}^{n} y_{k,l} \cdot x_{k,l}\right)\right]. = \ln C_k + \frac{1}{2}u_k \cdot L(u_k) + \frac{L_C}{2}\sum_{l=1}^{n} y_{k,l} \cdot x_{k,l} \tag{69}$$

Hodnota $\ln C_k$ není závislá na u_k ani na $x_{k,l}$, je proto možné ji považovat za konstantní a při samotném výpočtu ji vypustit.

Inicializace jednotlivých metrik se řídí podmínkou: $\alpha_0^{MLM}(s) = \begin{cases} 0 & pro\ s = 0 \\ -\infty & pro\ \forall s \neq 0 \end{cases}$ pro dopřednou metriku a $\beta_N^{MLM}(s) = \begin{cases} 0 & pro\ s = 0 \\ -\infty & pro\ \forall s \neq 0 \end{cases}$ pro zpětnou metriku.

4.5.1 Výpočetní složitost MAP a Max-Log-MAP algoritmu

V tabulce (Tab. 4) je uvedeno srovnání výpočetní složitosti dekódovacích algoritmů MAP a Max-Log-MAP při použití (*n*, κ, *m*) konvolučního kodéru s κ vstupy a pamětí *m*. Hodnoty v druhém a čtvrtém sloupci jsou převzaty z [13] a odpovídají počtu jednotlivých výpočetních operací za jednotku času. Ve třetím a pátém sloupci jsou uvedeny konkrétní hodnoty pro kód s kódovým poměrem $R_c = \frac{1}{3}$ a pamětí *m* = 2.

Tab. 4. Odhady složitosti MAP a Max-Log-MAP algoritmu

	MAP		Max-Log-MAP	
Operace	-	$\kappa = 1, m = 2$	-	$\kappa = 1, m = 2$
Sčítání	$2 \cdot 2^{\kappa} \cdot 2^{m} + 6$	22	$4 \cdot 2^{\kappa} \cdot 2^{m} + 8$	40
Násobení	$5 \cdot 2^{\kappa} \cdot 2^{m} + 8$	48	$2 \cdot 2^{\kappa} \cdot 2^{m}$	16
fce. Max.	-	-	$4 \cdot 2^{m} - 2$	14
fce. Exp.	$2 \cdot 2^{\kappa} \cdot 2^{m}$	16	-	-
Celkem	-	86	-	70

4.6 Nevýhody turbo-kódů

Mezi hlavní nevýhody turbo-kódů patří poměrně vysoká složitost dekódování, a procesní doba, která roste s počtem iterací. [2] uvádí, že z tohoto důvodu jsou turbo-kódy vhodné zejména pro aplikace nevyžadující zpracování signálu ve věrném reálném čase. Další nevýhodou je časové zpoždění, jež vzniká v důsledku prokládání, a které bude přímo úměrné velikosti a složitosti prokladače.

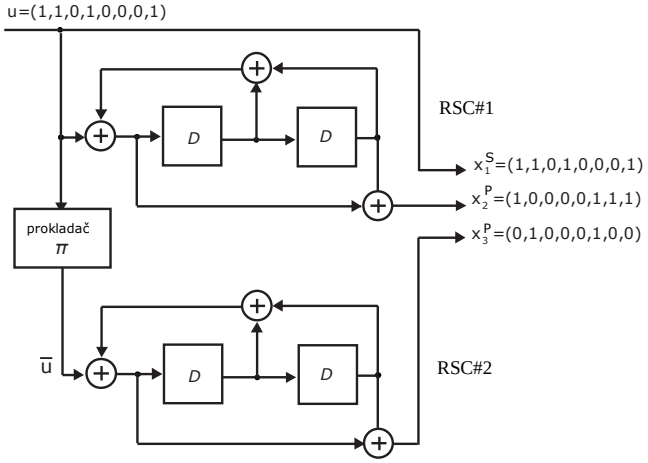
5 PŘÍKLAD KÓDOVÁNÍ A DEKÓDOVÁNÍ

5.1 Úvod

V této kapitole je na jednoduchém příkladu ilustrována funkce turbo-kodéru a dekodéru. Pro kódování je využit turbo-kodér s kódovým poměrem $R_c = 1/3$, který je složený ze dvou 4-stavových RSC$[7,5]_8$ kodérů s kódovým omezením K = 3. Vstupní datovou sekvenci tvoří 6 bitů, které jsou doplněny o 2-bitový konec zprávy, viz kapitola (kap. 2.4). Z důvodu velkého množství výpočtů v rámci MAP algoritmu, je detailně rozepsána pouze 1. iterace z celkových šesti. Pozn.: pro přiblížení se Shannonovu limitu vyžaduje praktická implementace výrazně delší vstupní datové sekvence.

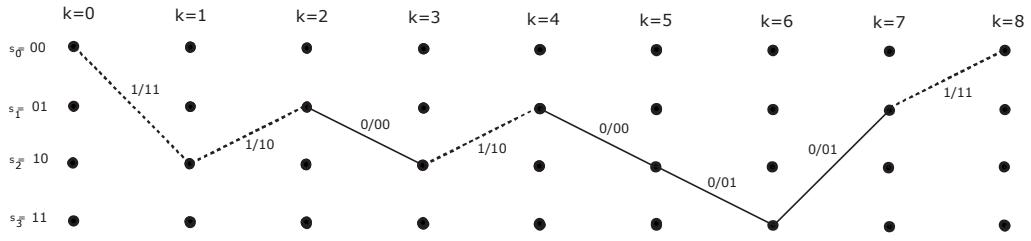
5.2 Kódování

Do kodéru znázorněném na obrázku (Obr. 27) vstupuje datová sekvence: $u = (u_0, u_1, u_2, u_3, u_4, u_5) = (1, 1, 0, 1, 0, 0)$ doplněna o konec zprávy: $u' = (u_6, u_7) = (0, 1)$.



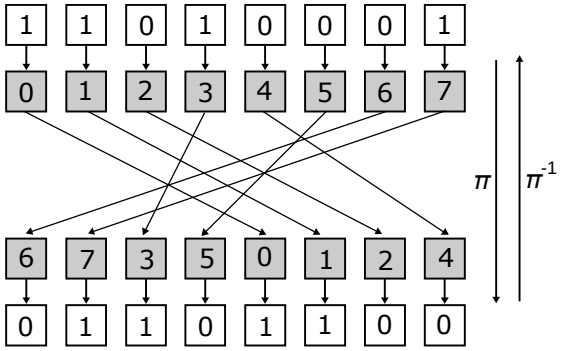
Obr. 27. Kodér turbo-kódu - příklad

Na obrázku (Obr. 28) je vyznačena cesta skrz trellis diagram, která odpovídá průchodu vstupní sekvence kodérem RSC#1.



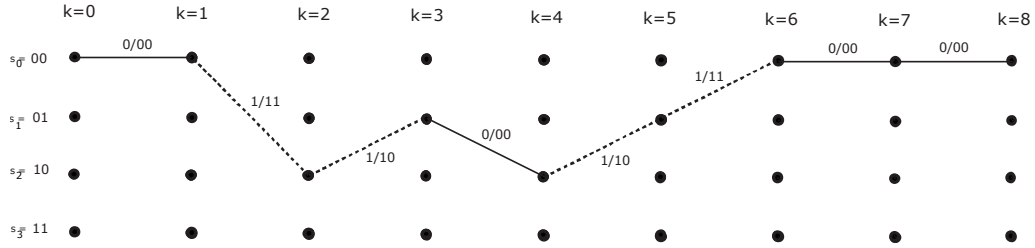
Obr. 28. Cesta trellis diagramem pro RSC#1 - příklad

Před vstupem do RSC#2 je vstupní sekvence proložena prostřednictvím uniformního pseudonáhodného prokladače znázorněného na obrázku (Obr. 29).



Obr. 29. Uniformní prokladač - příklad

Na obrázku (Obr. 30) je vyznačena cesta skrz trellis diagram, která odpovídá průchodu proložené vstupní sekvence $\underline{u} = (0, 1, 1, 0, 1, 1, 0, 0)$ přes kodér RSC#2.



Obr. 30. Cesta trellis diagramem pro RSC#2 - příklad

Výstupní systematické bity a paritní bity jsou v modulátoru mapovány na symboly {+1,-1}, které jsou odeslány přes AWGN kanál. Výstupní kódové trojice a jejich mapování na symboly je uvedeno v tabulce (Tab. 5).

Tab. 5. Mapování informačních a paritních bitů na symboly

k	$u_k = x_{k,1}^S$	$x_{k,2}^P$	$x_{k,3}^P$		$x_{k,1}^S$	$x_{k,2}^P$	$x_{k,3}^P$
0	1	1	0		+1	+1	-1
1	1	0	1	$\rightarrow$	+1	-1	+1
2	0	0	0		-1	-1	-1
3	1	0	0	mapování	+1	-1	-1
4	0	0	0	na	-1	-1	-1
5	0	1	1	symboly	-1	+1	+1
6	0	1	0		-1	+1	-1
7	1	1	0		+1	+1	-1

5.3 Přenosový AWGN kanál

Zakódovaná sekvence je přenášena přes AWGN kanál, ve kterém na jednotlivé bity působí bílý šumový signál $n_{k,l}$ s normálním rozdělením $N(\mu, \sigma^2) = N(0,1)$, viz tabulka (Tab. 6). Ve třetí části tabulky jsou uvedeny hodnoty přijatých symbolů, kde černě označená pole odpovídají chybám, které by se vyskytly při přímé aplikaci tvrdého rozhodování.

Tab. 6. Hodnoty přijatých symbolů

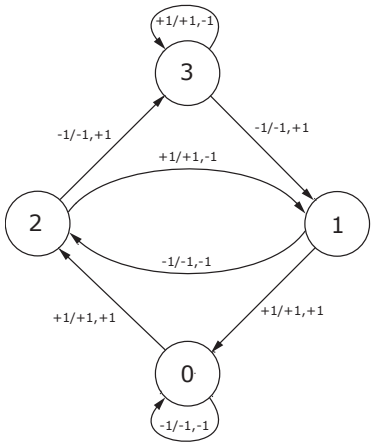
k	$x^S_{k,1}$	$x^P_{k,2}$	$x^P_{k,3}$		$n_{k,1}$	$n_{k,2}$	$n_{k,3}$		$y^S_{k,1}$	$y^P_{k,2}$	$y^P_{k,3}$
0	+1	+1	-1		0,014	1,016	0,306		1,014	2,016	-0,694
1	+1	-1	+1		-0,677	1,288	0,231		0,323	**0,288**	1,231
2	-1	-1	-1		-0,303	1,103	-0,395		-1,303	**0,103**	-1,395
3	+1	-1	-1	+	-1,163	-0,29	-0,495	=	**-0,163**	-1,29	-1,495
4	-1	-1	-1		0,764	0,952	-0,783		-0,236	-0,048	-1,783
5	-1	+1	+1		-0,092	-0,723	-1,68		-1,092	0,277	**-0,68**
6	-1	+1	-1		-0,033	-1,012	0,633		-1,033	**-0,012**	-0,367
7	+1	+1	-1		-2,54	-1,555	0,188		**-1,54**	**-0,555**	-0,812

5.4 Dekódování

5.4.1 1. Iterace DEC#1

V průběhu první poloviny 1. iterace vstupují do DEC#1 $L_C y^S_{k,1}$ a $L_C y^P_{k,2}$, tento dekodér však nemá k dispozici a-priorní LLR $L_2(u_k)$ resp. inverzně proloženou extrinsickou informaci $L_{e20}(u_k)$, ta je proto nastavena na 0. Hodnota spolehlivosti kanálu je $L_C = 4a\frac{E_b}{2\sigma^2} = \frac{4}{2 \cdot 1^2} = 2$.

Metrika přechodu je pro všechny přechody z `s do *s* vypočítána podle vztahu $\gamma_k(\grave{s}, s) = \exp\left[\frac{x^S_{k,1} \cdot L_2(u_k)}{2} + \frac{L_C}{2}\left(y^S_{k,1} \cdot x^S_{k,1} + y^P_{k,2} \cdot x^P_{k,2}\right)\right]$. Na obrázku (Obr. 31) je uveden stavový diagram příslušného RSC kodéru. Hodnoty přiřazené k hranám mají význam vstupního bitu, výstupního systematického bitu a paritního bitu, tedy $u_k / x^S_{k,1}, x^P_{k,l}$, kde $u_k = x^S_{k,1}$.



Obr. 31. Stavový diagram

RSC kodéru - příklad

Př. výpočtu hodnoty $\gamma_0(\grave{s}, s)$:

$x_{0,1}^{S} = -1$

$$\gamma_0(0,0) = \exp\left[0{,}5 \cdot \left((-1) \cdot 0\right) + 0{,}5 \cdot 2 \cdot \left(1{,}014 \cdot (-1) + 2{,}016 \cdot (-1)\right)\right] = 0{,}048$$
$$\gamma_0(1,2) = \exp\left[0{,}5 \cdot \left((-1) \cdot 0\right) + 0{,}5 \cdot 2 \cdot \left(1{,}014 \cdot (-1) + 2{,}016 \cdot (-1)\right)\right] = 0{,}048$$
$$\gamma_0(2,3) = \exp\left[0{,}5 \cdot \left((-1) \cdot 0\right) + 0{,}5 \cdot 2 \cdot \left(1{,}014 \cdot (-1) + 2{,}016 \cdot 1\right)\right] = 2{,}724$$
$$\gamma_0(3,1) = \exp\left[0{,}5 \cdot \left((-1) \cdot 0\right) + 0{,}5 \cdot 2 \cdot \left(1{,}014 \cdot (-1) + 2{,}016 \cdot 1\right)\right] = 2{,}724$$

$x_{0,1}^{S} = +1$

$$\gamma_0(0,2) = \exp\left[0{,}5 \cdot \left(1 \cdot 0\right) + 0{,}5 \cdot 2 \cdot \left(1{,}014 \cdot 1 + 2{,}016 \cdot 1\right)\right] = 20{,}697$$
$$\gamma_0(1,0) = \exp\left[0{,}5 \cdot \left(1 \cdot 0\right) + 0{,}5 \cdot 2 \cdot \left(1{,}014 \cdot 1 + 2{,}016 \cdot 1\right)\right] = 20{,}697$$
$$\gamma_0(2,1) = \exp\left[0{,}5 \cdot \left(1 \cdot 0\right) + 0{,}5 \cdot 2 \cdot \left(1{,}014 \cdot 1 + 2{,}016 \cdot (-1)\right)\right] = 0{,}367$$
$$\gamma_0(3,3) = \exp\left[0{,}5 \cdot \left(1 \cdot 0\right) + 0{,}5 \cdot 2 \cdot \left(1{,}014 \cdot 1 + 2{,}016 \cdot (-1)\right)\right] = 0{,}367$$

Vypočítané hodnoty metriky $\gamma_k(\grave{s}, s)$ jsou uvedeny v tabulce (Tab. 7).

Tab. 7. Hodnoty metriky $\gamma_k(\grave{s}, s)$ vypočítané v DEC#1 v rámci 1. iterace

$\grave{s}$, s	$\gamma_0(\grave{s}, s)$	$\gamma_1(\grave{s}, s)$	$\gamma_2(\grave{s}, s)$	$\gamma_3(\grave{s}, s)$	$\gamma_4(\grave{s}, s)$	$\gamma_5(\grave{s}, s)$	$\gamma_6(\grave{s}, s)$	$\gamma_7(\grave{s}, s)$
0,0	0,048	0,543	3,32	4,274	1,329	2,258	2,845	8,128
1,2	0,048	0,543	3,32	4,274	1,329	2,258	2,845	8,128
2,3	2,724	0,966	4,082	0,324	1,207	3,929	2,775	2,68
3,1	2,724	0,966	4,082	0,324	1,207	3,929	2,775	2,68
0,2	20,697	1,841	0,301	0,234	0,753	0,443	0,351	0,123
1,0	20,697	1,841	0,301	0,234	0,753	0,443	0,351	0,123
2,1	0,367	1,035	0,245	3,087	0,829	0,254	0,36	0,373
3,3	0,367	1,035	0,245	3,087	0,829	0,254	0,36	0,373

Dopředná metrika je vypočítána podle vztahu $\alpha_k(s) = \sum_{All\ `s} \gamma_k(`s, s) \cdot \alpha_{k-1}(`s)$ z $\gamma_k(`s, s)$ a inicializačních hodnot $\alpha_0(s) = \begin{cases} 1 & pro\ s = 0 \\ 0 & pro\ \forall s \neq 0 \end{cases}$. Hodnoty metriky $\alpha_k(s)$ jsou uvedeny v tabulce (Tab. 8).

Tab. 8. Hodnoty metriky $\alpha_k(s)$ vypočítané v DEC#1 v rámci 1. iterace

s	$\alpha_0(s)$	$\alpha_1(s)$	$\alpha_2(s)$	$\alpha_3(s)$	$\alpha_4(s)$	$\alpha_5(s)$	$\alpha_6(s)$	$\alpha_7(s)$	$\alpha_8(s)$
0	1	0,041	0,002	0,159	0,288	0,347	0,495	0,532	1,395
1	0	0,000	1,033	1,965	1,343	0,514	1,386	0,877	0,649
2	0	20,705	0,004	1,712	2,127	0,501	0,638	1,156	2,274
3	0	0,000	0,963	0,127	0,240	0,692	1,044	0,605	1,046
Sum.	1	20,746	2,002	3,963	3,998	2,054	3,562	3,171	5,364

V tabulce (Tab. 9) jsou uvedeny normalizované hodnoty $\alpha_k(s)$, které jsou vypočítány podle vztahu $\alpha_k(s) = \alpha_k(s) / \sum_{All\ s} \alpha_k(s)$, kde $\sum_{All\ s} \alpha_k(s) = Sum$.

Tab. 9. Normalizované hodnoty metriky $\alpha_k(s)$ vypočítané v DEC#1 v rámci 1. iterace

s	$\alpha_0(s)$	$\alpha_1(s)$	$\alpha_2(s)$	$\alpha_3(s)$	$\alpha_4(s)$	$\alpha_5(s)$	$\alpha_6(s)$	$\alpha_7(s)$	$\alpha_8(s)$
0	1	0,041	0,002	0,159	0,288	0,347	0,495	0,532	1,395
1	0	0,000	1,033	1,965	1,343	0,514	1,386	0,877	0,649
2	0	20,705	0,004	1,712	2,127	0,501	0,638	1,156	2,274
3	0	0,000	0,963	0,127	0,240	0,692	1,044	0,605	1,046

Zpětná metrika je vypočítána podle vztahu $\beta_{k-1}(`s) = \sum_{All\ s} \beta_k(s) \cdot \gamma_k(`s, s)$ z $\gamma_k(`s, s)$ a inicializačních hodnot $\beta_N(s) = \begin{cases} 1 & pro\ s = 0 \\ 0 & pro\ \forall s \neq 0 \end{cases}$. Hodnoty metriky $\beta_k(s)$ jsou uvedeny v tabulce (Tab. 10).

Tab. 10. Hodnoty metriky $\beta_k(s)$ vypočítané v DEC#1 v rámci 1. iterace

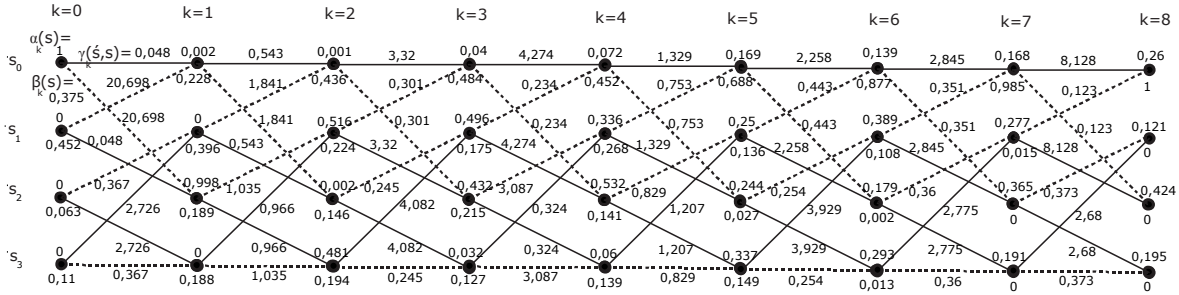
s	$\beta_8(s)$	$\beta_7(s)$	$\beta_6(s)$	$\beta_5(s)$	$\beta_4(s)$	$\beta_3(s)$	$\beta_2(s)$	$\beta_1(s)$	$\beta_0(s)$
0	1	8,127	2,803	1,982	0,935	1,966	1,672	0,507	3,914
1	0	0,124	0,345	0,392	0,554	0,711	0,859	0,881	4,718
2	0	0,000	0,006	0,078	0,292	0,873	0,560	0,420	0,658
3	0	0,000	0,042	0,429	0,287	0,516	0,744	0,418	1,148
Sum.	1	8,251	3,196	2,881	2,068	4,065	3,835	2,226	10,438

V tabulce (Tab. 11) jsou uvedeny normalizované hodnoty $\beta_k(s)$, které jsou vypočítány podle vztahu $\beta_k(s) = \beta_k(s) / \sum_{All\ s} \beta_k(s)$, kde $\sum_{All\ s} \beta_k(s) = Sum$.

Tab. 11. Normalizované hodnoty metriky $\beta_k(s)$ vypočítané v DEC#1 v rámci 1. iterace

s	$\beta_8(s)$	$\beta_7(s)$	$\beta_6(s)$	$\beta_5(s)$	$\beta_4(s)$	$\beta_3(s)$	$\beta_2(s)$	$\beta_1(s)$	$\beta_0(s)$
0	1	0,985	0,877	0,688	0,452	0,484	0,436	0,228	0,375
1	0	0,015	0,108	0,136	0,268	0,175	0,224	0,396	0,452
2	0	0,000	0,002	0,027	0,141	0,215	0,146	0,189	0,063
3	0	0,000	0,013	0,149	0,139	0,127	0,194	0,188	0,110

Shrnutí vypočítaných hodnot v DEC#1 během 1. iterace je uvedeno v trellis diagramu na obrázku (Obr. 32).



Obr. 32. Trellis diagram s vypočítanými hodnotami metrik

Při znalosti hodnot jednotlivých metrik $\gamma_k(\grave{s},s)$, $\alpha_k(s)$ a $\beta_k(s)$ je možné podle vztahu

$$L(u_k \mid \underline{y}) = \ln\left[\frac{\sum_{(\grave{s},s)\Rightarrow u_k=+1} \beta_{k+1}(s)\cdot\gamma_k(\grave{s},s)\cdot\alpha_k(\grave{s})}{\sum_{(\grave{s},s)\Rightarrow u_k=-1} \beta_{k+1}(s)\cdot\gamma_k(\grave{s},s)\cdot\alpha_k(\grave{s})}\right]$$

vypočítat a-posteriorní LLR pro jednotlivé bity u_k. Z hodnot $L(u_k \mid \underline{y})$ je poté extrahována hodnota extrinsické informace $L_{e11}(u_k) = L(u_k \mid \underline{y}) - L_2(u_k) - L_C y_{k,1}^S$.

Př. výpočtu hodnoty $L(u_0 \mid \underline{y})$ a $L_{e11}(u_0)$:

$$L(u_0 \mid \underline{y}) = \ln\left[\frac{\beta_1(2)\cdot\gamma_0(0,2)\cdot\alpha_0(0)+\beta_1(0)\cdot\gamma_0(1,0)\cdot\alpha_0(1)+\beta_1(1)\cdot\gamma_0(2,1)\cdot\alpha_0(2)+\beta_1(3)\cdot\gamma_0(3,3)\cdot\alpha_0(3)}{\beta_1(0)\cdot\gamma_0(0,0)\cdot\alpha_0(0)+\beta_1(2)\cdot\gamma_0(1,2)\cdot\alpha_0(1)+\beta_1(3)\cdot\gamma_0(2,3)\cdot\alpha_0(2)+\beta_1(1)\cdot\gamma_0(3,1)\cdot\alpha_0(3)}\right]$$
$$= \ln\left[\frac{0{,}189\cdot 20{,}697\cdot 1+0+0+0}{0{,}228\cdot 0{,}048\cdot 1+0+0+0}\right] = 5{,}878$$

$$L_{e11}(u_0) = 5{,}878 - 0 - 2\cdot 1{,}014 = 3{,}85$$

Hodnoty $L(u_k \mid \underline{y})$ vypočítané v první polovině 1. iterace a jim odpovídající extrinsické informace jsou uvedeny v tabulce (Tab. 12).

Tab. 12. Hodnoty $L(u_k \mid \underline{y})$ vypočítané v DEC#1

k	$L(u_k \mid \underline{y})$	$L_{e11}(u_k)$
0	5,878	3,85
1	0,212	-0,434
2	-2,066	0,54
3	0,063	0,389
4	0,284	0,756
5	-1,546	0,638
6	-1,082	0,984
7	-3,687	-0,607

Jestliže by v tuto chvíli bylo na základě měkkého rozhodnutí $L(u_k \mid \underline{y})$ tvrdě rozhodnuto o hodnotách bitů u_k, odpovídal by výstup dekodéru sekvenci $\hat{u} = \{1, 1, 0, 1, \mathbf{1}, 0, 0, \mathbf{0}\}$. Dvě chyby (označeny černě) by tedy zůstaly neodhaleny.

5.4.2 1. Iterace DEC#2

V průběhu druhé poloviny 1. iterace vstupuje do DEC#2 hodnota $L_C y^P_{k,3}$ a proložené hodnota $L_C y^S_{k,1}$. A-priorní LLR $L_1(u_k)$ odpovídá proložené extrinsické informaci $L_{e11}(u_k)$.

Metrika přechodu je pro všechny přechody z $`s$ do $s$ vypočítána podle vztahu $\gamma_k(`s, s) = \exp\left[\frac{x^S_{k,1} \cdot L_1(u_k)}{2} + \frac{L_C}{2}\left(y^S_{k,1} \cdot x^S_{k,1} + y^P_{k,3} \cdot x^P_{k,3}\right)\right]$.

Př. výpočtu hodnoty $\gamma_0(`s, s)$:

$x^S_{0,1} = -1$

$$\gamma_0(0,0) = \exp[0{,}5 \cdot ((-1) \cdot 0{,}984) + 0{,}5 \cdot 2 \cdot ((-1{,}033) \cdot (-1) + (-0{,}694) \cdot (-1))] = 3{,}439$$
$$\gamma_0(1,2) = \exp[0{,}5 \cdot ((-1) \cdot 0{,}984) + 0{,}5 \cdot 2 \cdot ((-1{,}033) \cdot (-1) + (-0{,}694) \cdot (-1))] = 3{,}439$$
$$\gamma_0(2,3) = \exp[0{,}5 \cdot ((-1) \cdot 0{,}984) + 0{,}5 \cdot 2 \cdot ((-1{,}033) \cdot (-1) + (-0{,}694) \cdot 1)] = 0{,}858$$
$$\gamma_0(3,1) = \exp[0{,}5 \cdot ((-1) \cdot 0{,}984) + 0{,}5 \cdot 2 \cdot ((-1{,}033) \cdot (-1) + (-0{,}694) \cdot 1)] = 0{,}858$$

$x^S_{0,1} = +1$

$$\gamma_0(0,2) = \exp[0{,}5 \cdot (1 \cdot 0{,}984) + 0{,}5 \cdot 2 \cdot ((-1{,}033) \cdot 1 + (-0{,}694) \cdot 1)] = 0{,}291$$

$$\gamma_0(1,0) = \exp[0{,}5 \cdot (1 \cdot 0{,}984) + 0{,}5 \cdot 2 \cdot ((-1{,}033) \cdot 1 + (-0{,}694) \cdot 1)] = 0{,}291$$
$$\gamma_0(2,1) = \exp[0{,}5 \cdot (1 \cdot 0{,}984) + 0{,}5 \cdot 2 \cdot ((-1{,}033) \cdot 1 + (-0{,}694) \cdot (-1))] = 1{,}165$$
$$\gamma_0(3,3) = \exp[0{,}5 \cdot (1 \cdot 0{,}984) + 0{,}5 \cdot 2 \cdot ((-1{,}033) \cdot 1 + (-0{,}694) \cdot (-1))] = 1{,}165$$

Vypočítané hodnoty metriky $\gamma_k(`s, s)$ jsou uvedeny v tabulce (Tab. 13).

Tab. 13. Hodnoty metriky $\gamma_k(`s, s)$ vypočítané v DEC#2 v rámci 1. iterace

`s , s	$\gamma_0(`s,s)$	$\gamma_1(`s,s)$	$\gamma_2(`s,s)$	$\gamma_3(`s,s)$	$\gamma_4(`s,s)$	$\gamma_5(`s,s)$	$\gamma_6(`s,s)$	$\gamma_7(`s,s)$
0,0	3,439	1,845	3,911	9,659	0,316	1,776	4,056	1,956
1,2	3,439	1,845	3,911	9,659	0,316	1,776	4,056	1,956
2,3	0,858	21,649	0,24	0,486	0,009	0,456	1,948	0,385
3,1	0,858	21,649	0,24	0,486	0,009	0,456	1,948	0,385
0,2	0,291	0,542	0,256	0,104	3,168	0,563	0,247	0,511
1,0	0,291	0,542	0,256	0,104	3,168	0,563	0,247	0,511
2,1	1,165	0,046	4,167	2,059	112,008	2,193	0,513	2,597
3,3	1,165	0,046	4,167	2,059	112,008	2,193	0,513	2,597

Dopředná metrika je vypočítána podle vztahu $\alpha_k(s) = \sum_{All\ `s} \gamma_k(`s, s) \cdot \alpha_{k-1}(`s)$. Hodnoty metriky $\alpha_k(s)$ jsou uvedeny v tabulce (Tab. 15).

Tab. 14. Hodnoty metriky $\alpha_k(s)$ vypočítané v DEC#2 v rámci 1. iterace

s	$\alpha_0(s)$	$\alpha_1(s)$	$\alpha_2(s)$	$\alpha_3(s)$	$\alpha_4(s)$	$\alpha_5(s)$	$\alpha_6(s)$	$\alpha_7(s)$	$\alpha_8(s)$
0	1	3,439	1,701	1,712	3,853	0,302	0,345	0,594	0,551
1	0	0,000	0,004	0,641	0,259	25,481	0,259	0,857	0,545
2	0	0,291	0,498	0,116	1,479	1,897	1,054	0,460	0,679
3	0	0,000	1,689	1,837	0,895	15,435	0,803	1,002	0,954
Sum.	1	3,730	3,893	4,306	6,486	43,115	2,461	2,914	2,728

V tabulce (Tab. 9) jsou uvedeny normalizované hodnoty $\alpha_k(s)$, které jsou vypočítány podle vztahu $\alpha_k(s) = \alpha_k(s) / \sum_{All\ s} \alpha_k(s)$, kde $\sum_{All\ s} \alpha_k(s) = Sum$.

Tab. 15. Normalizované hodnoty metriky $\alpha_k(s)$ vypočítané v DEC#2 v rámci 1. iterace

s	$\alpha_0(s)$	$\alpha_1(s)$	$\alpha_2(s)$	$\alpha_3(s)$	$\alpha_4(s)$	$\alpha_5(s)$	$\alpha_6(s)$	$\alpha_7(s)$	$\alpha_8(s)$
0	1	0,922	0,437	0,398	0,594	0,007	0,14	0,204	0,202
1	0	0	0,001	0,149	0,04	0,591	0,105	0,294	0,2
2	0	0,078	0,128	0,027	0,228	0,044	0,428	0,158	0,249
3	0	0	0,434	0,427	0,138	0,358	0,326	0,344	0,35

Zpětná metrika je vypočítána podle vztahu $\beta_{k-1}(\grave{s}) = \sum_{All\ s} \beta_k(s) \cdot \gamma_k(\grave{s}, s)$. Inicializační hodnota je v tomto případě $\beta_N(s) = \frac{1}{2^M} = \frac{1}{2^2} = 0{,}25 \quad pro\ \forall s$. Hodnoty metriky $\beta_k(s)$ jsou uvedeny v tabulce (Tab. 16).

Tab. 16. Hodnoty metriky $\beta_k(s)$ vypočítané v DEC#2 v rámci 1. iterace

s	$\beta_8(s)$	$\beta_7(s)$	$\beta_6(s)$	$\beta_5(s)$	$\beta_4(s)$	$\beta_3(s)$	$\beta_2(s)$	$\beta_1(s)$	$\beta_0(s)$
0	0,25	0,346	0,223	0,107	2,188	0,140	1,096	2,065	0,719
1	0,25	1,820	0,639	0,107	36,024	0,109	0,886	2,449	0,719
2	0,25	0,286	2,070	1,782	2,236	2,831	1,465	1,363	0,871
3	0,25	0,272	0,450	0,502	8,167	2,995	0,938	1,221	0,871
Sum.	1	2,724	3,382	2,499	48,616	6,074	4,385	7,097	3,180

V tabulce (Tab. 17) jsou uvedeny normalizované hodnoty $\beta_k(s)$, které jsou vypočítány podle vztahu $\beta_k(s) = \beta_k(s) / \sum_{All\ s} \beta_k(s)$, kde $\sum_{All\ s} \beta_k(s) = Sum$.

Tab. 17. Normalizované hodnoty metriky $\beta_k(s)$ vypočítané v DEC#2 v rámci 1. iterace

s	$\beta_8(s)$	$\beta_7(s)$	$\beta_6(s)$	$\beta_5(s)$	$\beta_4(s)$	$\beta_3(s)$	$\beta_2(s)$	$\beta_1(s)$	$\beta_0(s)$
0	0,25	0,127	0,066	0,043	0,045	0,023	0,25	0,291	0,226
1	0,25	0,668	0,189	0,043	0,741	0,018	0,202	0,345	0,226
2	0,25	0,105	0,612	0,713	0,046	0,466	0,334	0,192	0,274
3	0,25	0,1	0,133	0,201	0,168	0,493	0,214	0,172	0,274

A-posteriorního LLR pro jednotlivé bity u_k je opět vypočítán podle vztahu

$$L(u_k \mid \underline{y}) = \ln\left[\frac{\sum_{(\grave{s},s)\Rightarrow u_k=+1} \beta_{k+1}(s) \cdot \gamma_k(\grave{s}, s) \cdot \alpha_k(\grave{s})}{\sum_{(\grave{s},s)\Rightarrow u_k=-1} \beta_{k+1}(s) \cdot \gamma_k(\grave{s}, s) \cdot \alpha_k(\grave{s})}\right].$$

Extrinsická informace je dána rozdílem $L_{e21}(u_k) = L(u_k \mid \underline{y}) - L_1(u_k) - L_C y_{k,1}^S$.

Hodnoty $L(u_k \mid \underline{y})$ vypočítané v druhé polovině 1. iterace a jim odpovídající extrinsické informace jsou uvedeny v tabulce (Tab. 18).

Tab. 18. Hodnoty $L(u_k \mid \underline{y})$ vypočítané v DEC#2

k	$L(u_k \mid \underline{y}) \rightarrow \pi^{-1}$	$L_{e21}(u_k) \rightarrow \pi^{-1}$
0	5,167	-0,705
1	0,003	-0,208
2	-1,718	0,35
3	1,487	1,423
4	0,289	0,006
5	-0,526	1,019
6	-0,237	0,845
7	-0,144	3,544

Tvrdé rozhodnutí odpovídá sekvenci $\hat{u} = \{1, 1, 0, 1, \mathbf{1}, 0, 0, \mathbf{0}\}$. Stejně jako u DEC#1 zůstaly dvě chyby neodhaleny, je však patrná změna odpovídajících měkkých rozhodnutí, které konvergují ke správné hodnotě.

5.4.3 Další iterace

Tabulka (Tab. 19) obsahuje seznam vypočítaných hodnot a-posteriorního LLR $L(u_k \mid \underline{y})$ v DEC#2 pro všech 6 iterací. Z tabulky je patrné, že tvrdé rozhodnutí by již po 2. iteraci odpovídalo odeslané datové sekvenci, a všechny chyby by byly opraveny. Po 6. iteraci je výstupní dekódovaná sekvence $\hat{u} = \{1, 1, 0, 1, 0, 0, 0, 1\}$.

Tab. 19. Vypočítané hodnoty $L(u_k \mid \underline{y})$ v DEC#2 v průběhu šesti iterací

Iterace	k	0	1	2	3	4	5	6	7
1.	$L(u_k \mid \underline{y})$	5,167	0,003	-1,718	1,487	0,289	-0,526	-0,237	-0,144
	rozhodnutí	1	1	0	1	**1**	0	0	**0**
2.	$L(u_k \mid \underline{y})$	4,627	0,816	-2,603	1,164	-0,504	-2,16	-1,823	0,723
	rozhodnutí	1	1	0	1	0	0	0	1
3.	$L(u_k \mid \underline{y})$	6,063	1,589	-3,277	2,184	-1,393	-2,801	-2,625	1,662
	rozhodnutí	1	1	0	1	0	0	0	1
4.	$L(u_k \mid \underline{y})$	7,921	2,543	-4,353	3,564	-2,394	-4,347	-4,248	3,007
	rozhodnutí	1	1	0	1	0	0	0	1
5.	$L(u_k \mid \underline{y})$	10,489	3,895	-6,25	5,805	-3,627	-7,129	-7,145	4,626
	rozhodnutí	1	1	0	1	0	0	0	1
6.	$L(u_k \mid \underline{y})$	12,337	4,894	-8,328	8,298	-4,566	-10,465	-10,664	5,483
	rozhodnutí	1	1	0	1	0	0	0	1

6 APLIKACE TURBO-KÓDŮ

6.1 Úvod

Díky výbornému výkonu, kterého turbo-kódy dosahují ve srovnání s jinými kódy, byl tento koncept bezpečnostního kanálového kódování začleněn do mnoha komunikačních systémů a stal se součástí různých průmyslových standardů. V současné době turbo-kódy pokrývají širokou škálu aplikací od systémů pro ukládání dat, přes klasické drátové a bezdrátové komunikační systémy, až po systémy pro komunikaci se vzdálenými objekty ve vesmíru. Tabulka (Tab. 20) převzatá z [29] shrnuje hlavní normalizované či patentované aplikace turbo-kódů.

Tab. 20. Aplikace turbo-kódů [29]

Application	Turbo-code	termination	polynomials	rates
CCSDS (deep space)	binary, 16-state	tail bits	23,33,25,37	1/6,1/4,1/3,1/2
3GPP (UMTS)	binary, 8-state	tail bits	13,15,17	1/4,1/3,1/2
3GPP2 (CDMA2000)	binary, 8-state	tail bits	13,15,17	1/4,1/3,1/2
3G PP LTE (Long Term Evolution)	binary, 8-state	tail bits	13,15,17	1/4,1/3,1/2
DVB-RCS (Return Channel over Satellite)	double-binary, 8-state	circular	15,13	1/3 up to 6/7
DVB-RCT (Return Channel over Terrestrial)	double-binary, 8-state	circular	15,13	1/2,3/4
DVB-SSP (satellite service to portable)	double-binary, 8-state	tail bits	13,15	
IEEE 802.16 (WiMAX)	double-binary, 8-state	circular	15,13	
IEEE 802.16e (Mobile WiMAX)	double-binary, 8-state	circular	15,13	
Inmarsat (Aero-H)	binary, 16-state	no	23,35	
Eutelsat (Skyplex)	double-binary, 8-state	circular	15,13	4/5,6/7
Broadcom (Echostar)				
Qualcomm (MediaFLO)				
Homeplug	Turbo convolutional + product code			
Mitsubishi (optical communication)	Turbo product code			

6.2 Komunikace se vzdálenými objekty ve vesmíru

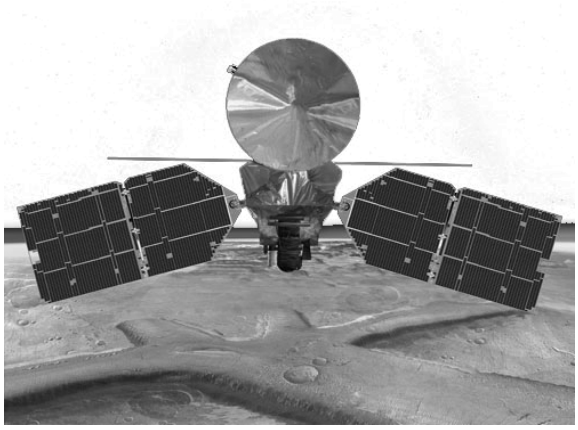
Jak uvádí [21], *Konzultační výbor pro systémy zpracovávající data z oblasti vesmíru* (*Consultative Committee for Space Data Systems, CCSDS)* vydává doporučení pro kódové zabezpečení telemetrických systémů, které používají vesmírné agentury z celého světa (*CNES, DLR, ESA, JAXA, NASA, RSA*, atd.). V roce 1999, tedy pouhých 6 let od jejich oficiální prezentace pro mezinárodní společenství, byly turbo-kódy zvoleny jako nová možnost pro CCSDS doporučení. Důvodem byla výrazně vyšší energetická účinnost, než které dosahovaly doposud využívané konvoluční a *Reed-Solomonovy kódy*.

Do tohoto doporučení patří turbo-kódy s 16-ti stavovými rekurzivními konvolučními kodéry, volitelným kódovým poměrem v hodnotách: $R = 1/2, 1/3, 1/4$, a $1/6$ a s prokladači od velikosti 1784 bitů do 16384 bitů. Turbo-kódy byly začleněny do komunikačních systémů mnohých zařízení, z nichž může být jmenována např. mezinárodní síť antén *DSN* (*Deep Space Network*) agentury NASA nebo vesmírné sondy:

Smart-1 - první evropská kosmická sonda organizace ESA vypuštěna v září 2003, která byla určena k průzkumu Měsíce. Mise byla ukončena v září 2006.

Rosetta - kosmická sonda organizace ESA vypuštěna v březnu 2004, která je určena k průzkumu jádra komety 67P/Churyumov-Gerasimenko. Té dosáhne po desetileté misi v roce 2014.

Mars Reconnaissance Orbiter (*MRO*), viz obrázek (Obr. 33 [30]) - planetární sonda organizace NASA vypuštěna v srpnu 2005, která je určená k průzkumu Marsu z oběžné dráhy. V březnu 2006 dosáhla Marsu a stala se jeho třetí aktivní družicí.



Obr. 33. Mars Reconnaissance Orbiter[30]

6.3 Třetí generace mobilních sítí

Třetí generace (*3G*) mobilních telekomunikačních standardů byla vytvořena *Mezinárodní telekomunikační unií* (*ITU*) v rámci projektu *IMT-2000* (*International Mobile Telecommunications by the year 2000*). Jeho předpokladem byla základna nabízející pevné, mobilní, hlasové, datové, multimediální služby a internetové připojení. Existují dva hlavní orgány vykonávání práce v rámci projektu IMT-2000: *3GPP* (*3rd Generation Partnership Project*) - partnerský projekt třetí generace, jehož specifikace založeny na rozvinutých *GSM* specifikacích, jsou známé jako *UMTS* (*Universal Mobile Telecommunication System*) a partnerský projekt *3GPP2*, který specifikuje 3G sítě založeny na *CDMA* (*Code Division Multiple Access*), známé jako *CDMA2000*.

Do specifikace kódového zabezpečení obou těchto projektů jsou začleněny turbo-kódy. V [17] je uvedeno, že systém UMTS používá kodér složený z 8-mi stavových RSC kodérů s kódovým poměrem $R = 1/3$ a délkou kódového omezení $K = 4$. Standard CDMA2000 využívá tři typy konvolučních kódů s kódovými poměry $R = 1/4$, 1/3 a 1/2 a omezující délkou $K = 9$ a turbo-kódy, které tvoří rovněž 8-mi stavové RSC kodéry s kódovým poměrem $R = 1/2$, 1/3 ,1/4 nebo 1/5.

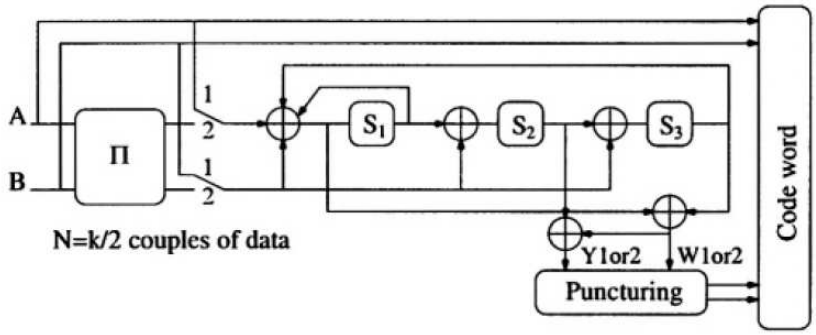
Nastupující *3GPP/LTE* (*Long Term Evolution*), je technologií sítí čtvrté generace, která vychází ze standardů 3GPP a nadále je rozvíjí. V závislosti na použití *MIMO* je pro downlink podporována rychlost více než 300 Mbit/s nebo 80 Mbit/s. Stejně tak, jako její předchůdci, využívá LTE kódové zabezpečení prostřednictvím turbo-kódů s podobnými parametry.

6.4 Digitální satelitní komunikace

DVB (*Digital Video Broadcasting*), je označení pro digitální televizní vysílání. *DVB-RCS (Digital Video Broadcast-Return Channel via Satellite)* je součástí evropské normy *ETSI*. DVB-RCS nabízí asymetrickou širokopásmovou komunikaci: po dopředném kanálu (k uživateli) se rychlosti pohybují v jednotkách až desítkách Mbit/s, po zpětném kanálu pak až 2 Mbit/s. Dopředný *DVB-S* kanál používá zapouzdření *IP* do formátu *MPEG-2* jako digitální televize, zpětný *RCS* kanál funguje jako multifrekvenční časový multiplex *MF-TDMA (Multi-Frequency Time Division Multiple Access*) [28].

Vzhledem k malému prostoru pro uplink signálu směrem k satelitu, je žádoucí kvalitní dopředná korekce chyb. Z tohoto důvodu byly do DVB-RCS standardu začleněny právě turbo-kódy. Ve [21] je uvedeno, že turbo-kódovací systém DVB-RCS je optimalizován pro krátké vstupní rámce o délkách 12 až 216 bytů a kódové poměry, které se pohybují v rozmezí $R = 1/3$ až $R = 6/7$. Pro kódování jsou využity tzv. kruhové rekurzivní systematické konvoluční kodéry (*Circular-RSC*), viz obrázek (Obr. 34 [5]), které využívají speciální techniku vyprázdnění kodéru. Na rozdíl od kódů definovaných nad *GF*(2), které byly popsány v předchozích kapitolách, využívá DVB-RCS tzv. *duo-binární kódy* definované nad *GF*(4). V každém kroku vstupují do CRSC kodérů dva datové bity a na výstupu se objevují dva paritní bity, viz obrázek (Obr. 34 [5]).

Přijetím turbo-kódů do specifikace DVB-RCS dosáhl satelitní přenos výrazně vyšší efektivity využití pásma a umožňuje dosažení velkých přenosových rychlostí. Díky těmto vlastnostem a vysokému pokrytí se stal satelitní internet vážným konkurentem DSL služeb.



Obr. 34. Duo-binární CRSC kodér používaný v DVB-RCS [5]

6.5 IEEE 802.16

WiMAX (*Worldwide Interoperability for Microwave Access*) je bezdrátová technologie definovaná v řadě norem IEEE 802.16, kterou lze zařadit do skupiny metropolitních sítí *MAN* (*Metropolitan Area Network*). IEEE 802.16e je rozšíření základní normy, které podporuje mobilitu uživatelů v metropolitním měřítku. Norma IEEE 802.16e je také označována jako *Mobile WiMAX*.

Pro zvýšení spolehlivosti přenosu v rámci FEC kódování uvádí [21] využití tzv. *Blokových turbo-kódů BTC* (Block *Turbo Code*) a tzv. *konvolučních turbo-kódů CTC* (*Convolutional Turbo Code*), které jsou podobně jako u DVB-RCS duo-binární.

6.6 Další aplikace Turbo-kódů

V současné době jsou různé typy turbo-kódů součástí mnohých aplikací v komerční i nekomerční sféře. Kromě již zmíněných aplikací uvádí [21] např. jejich nasazení satelitními operátory *INTELSAT* či *EUTELSAT* nebo v družicovém komunikačním systému *INMARSAT*. V oblasti optických komunikací jsou turbo-kódy použity jako jedna z možností pro snížení různých typů šumu a disperze [21]. Turbo-kódy bývají rovněž využívány technologií *Homeplug*, která umožňuje přenos dat po rozvodech elektrické sítě. V neposlední řadě je nutné jmenovat také oblast magnetických a optických médií pro záznam dat, kde poptávka po vyšší kapacitě, přenosové rychlosti a hustotě záznamu je rovněž důvodem výzkumu a aplikace turbo-kódů.

II. PRAKTICKÁ ČÁST

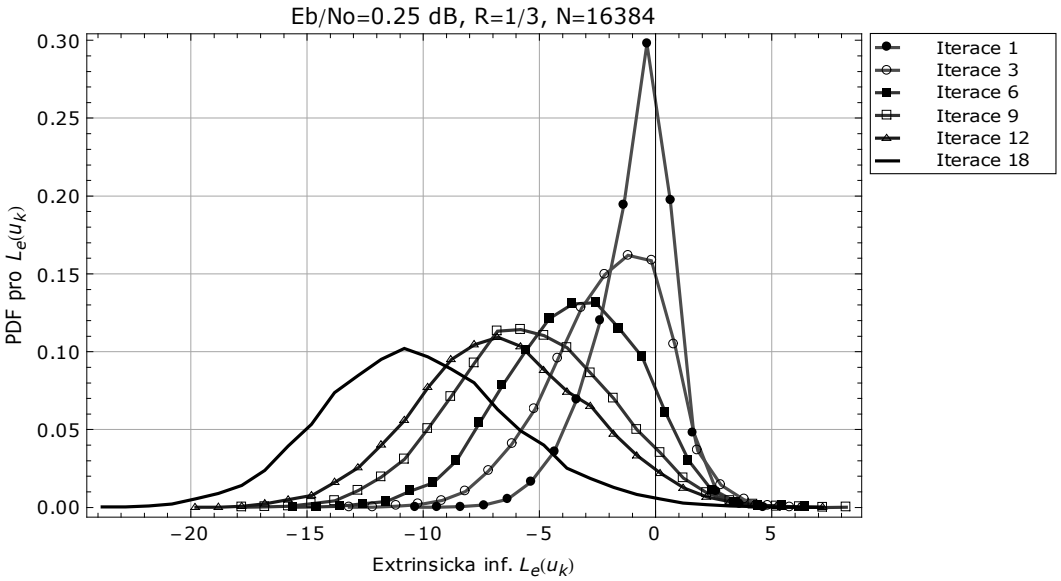
7 VÝSLEDKY SIMULACE TURBO-KODEKU

7.1 Úvod

V předchozí části této práce byla uvedena teoretická analýza a matematické základy, z nichž vychází činnost turbo-kódů. V rámci praktické části byly algoritmy, podle kterých pracuje turbo-kodér a dekodér, programově implementovány v jazyku C. Základní informace o tomto programu jsou v kapitole (kap. 8). V této kapitole jsou uvedeny výsledky simulací, které byly zpracovány z výstupů naprogramovaného kodeku. Jedné se o statistické zobrazení hodnot extrinsické informace a a-posteriorního LLR a testy chybovosti pro různá nastavení parametrů turbo-kódů, při využití dekódovacího algoritmu MAP nebo Max-Log-MAP.

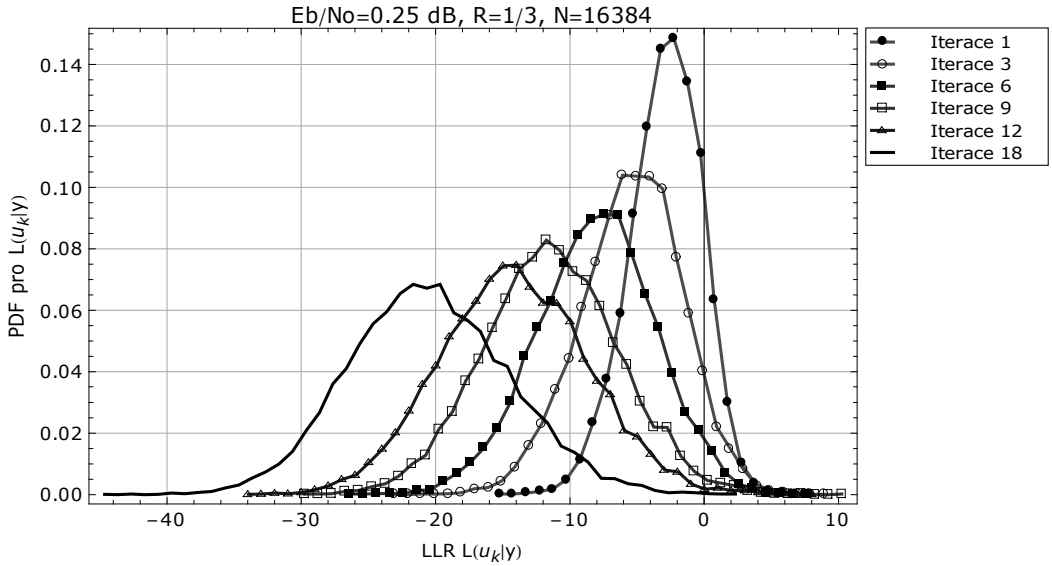
7.2 Relevance extrinsické informace a LLR

Na obrázku (Obr. 35) a (Obr. 36) jsou zobrazeny funkce, které reprezentují rozdělení pravděpodobnosti výskytu daných hodnot extrinsické informace $L_e(u_k)$ a a-posteriorního LLR $L(u_k \mid \underline{y})$ při různém počtu iterací pro blok N = 16384 bitů. Všechny přenášené bity odpovídají hodnotě log0 resp. -1. Z grafů je patrné, jak při rostoucím počtu iterací dochází k posunu $L_e(u_k)$ a $L(u_k \mid \underline{y})$ do oblasti záporných hodnot, a roste tak jejich relevance pro odhad hodnoty přenášených bitů.



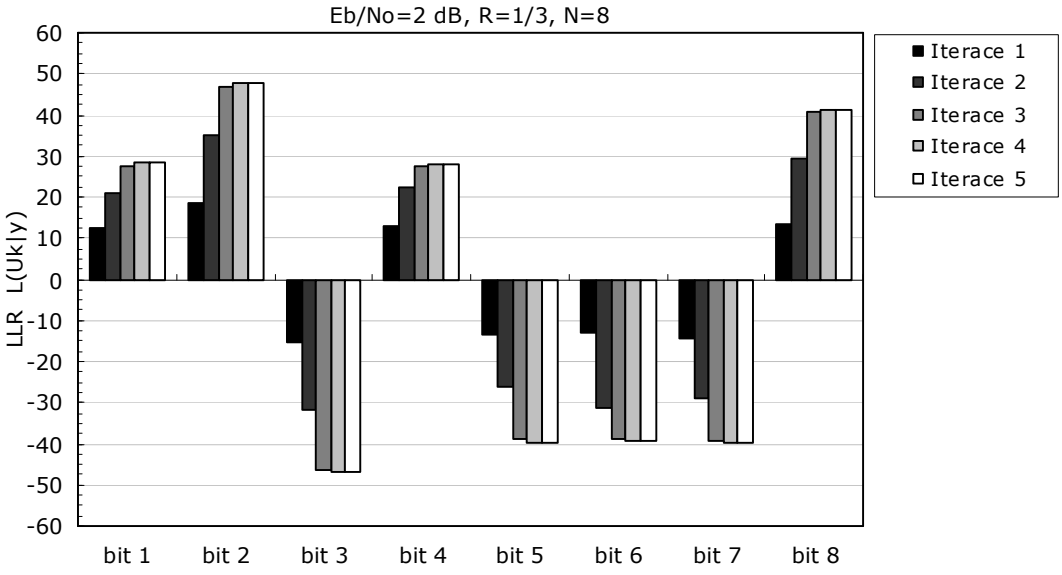
Obr. 35. Relevance extrinsické informace

Plocha, kterou jednotlivé funkce hustoty pravděpodobnosti $L(u_k \mid \underline{y})$ vymezují vpravo od bodu 0 na ose x, odpovídá chybovosti BER po aktuální iteraci, neboť leží v oblasti tvrdého rozhodnutí log1.



Obr. 36. Relevance a-posteriorního LLR

Podobné srovnání je znázorněno v grafu (Obr. 37). Ten ukazuje, jak se s rostoucím počtem iterací mění ohodnocení bitů hodnotou $L(u_k \mid \underline{y})$. Z grafu je patrné, že pro každý z osmi testovaných bitů $L(u_k \mid \underline{y})$ postupně konverguje k určité pevné hodnotě, na kterou již další zvyšovaní počtu iteraci nemá vliv.



Obr. 37. Konvergence hodnot $L(u_k \mid \underline{y})$

7.3 Test výkonnosti turbo-kódu

Možným způsobem zjištění výkonnosti turbo-kódů je testování chybovosti BER[1] v závislosti na hodnotě poměru E_b / N_0 [dB][2]. Základním parametrem, který v našem případě ovlivňuje vlastnosti přenosového AWGN kanálu je rozptyl šumového signálu σ^2. Odvození závislosti hodnoty σ^2 na poměru E_b / N_0 je uvedeno v [27]. Při kódovém poměru $R_c = \frac{1}{3}$ a využití BPKS modulace, je výsledný vztah dán rovnicí:

$$\sigma = \left(\sqrt{1 \cdot \frac{1}{3} \cdot 10^{\frac{G_{dB}}{10}}} \right)^{-1}, \tag{70}$$

kde $G_{dB} = 10 \cdot \log\left(\frac{E_b}{N_0} \right)$ udává hodnotu poměru E_b / N_0 v jednotkách decibel. Pomocí vztahu (70), lze pro různé hodnoty poměru E_b / N_0 testovat chybovost BER.

7.3.1 Vliv počtu iterací na výkonnost turbo-kódů

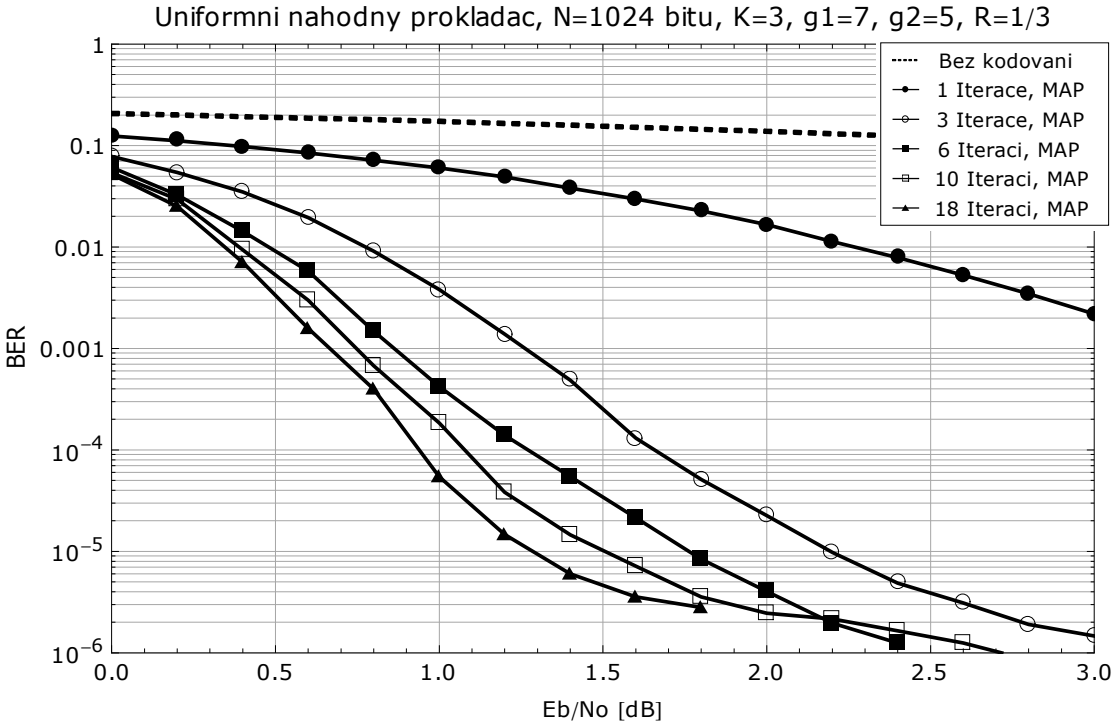
Obrázek (Obr. 38) ukazuje výkonnost turbo-kódu při použitém dekódovacím algoritmu MAP v závislosti na počtu provedených iterací. V simulaci je využit turbo-kodér s kódovým poměrem $R_c = \frac{1}{3}$ tvořený dvěma RSC$[7,5]_8$ kodéry a uniformním náhodným prokladačem. Z grafu je patrné, že se zlepšujícím se odstupem signálu od šumu se snižuje počet chyb a s rostoucím počtem iterací je potlačení chyb výrazně vyšší. Po aplikaci více než 6-ti iterací lze pozorovat, že další navyšování iteračních kroků již přináší poměrně malá zlepšení – pro BER 10^{-4} již asi jen o 0,1 dB.

Podobných výsledků bylo dosaženo i při použití Max-Log-MAP algoritmu, viz obrázek (Obr. 39). Srovnání výkonu obou dekódovacích algoritmů je na obrázku (Obr. 40).

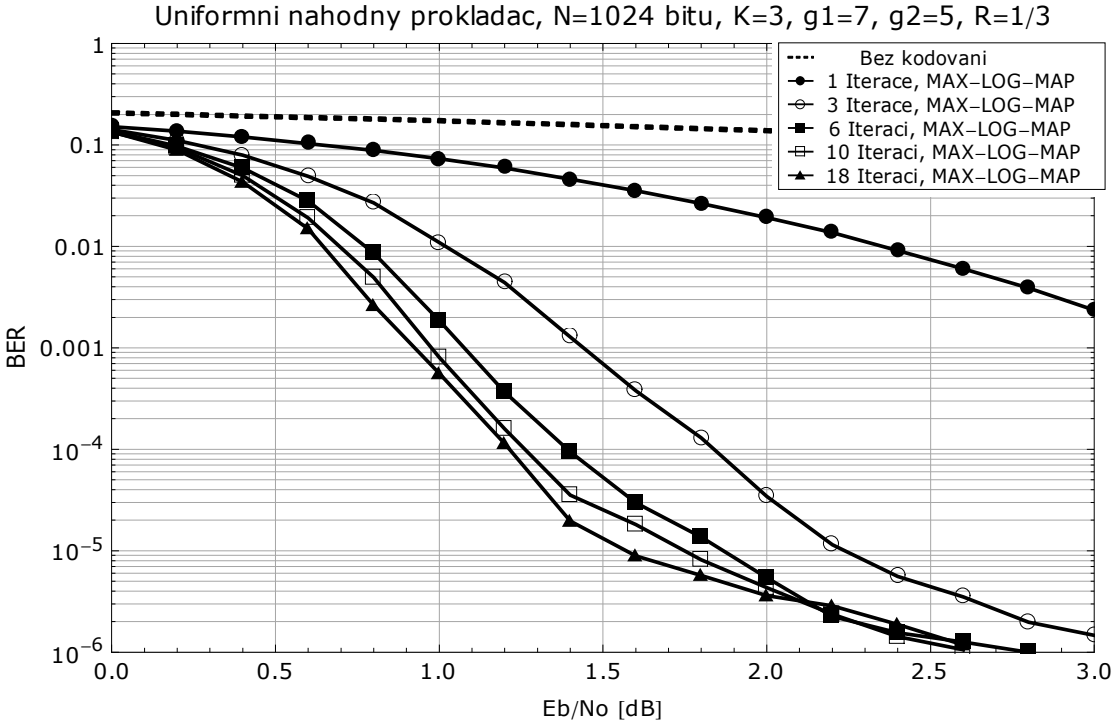
---

[1] Bitová chybovost BER (Bit Error Rate) je definována poměrem chybně přijatých bitů ku celkovému počtu přijatých bitů za určitou dobu měření [26].

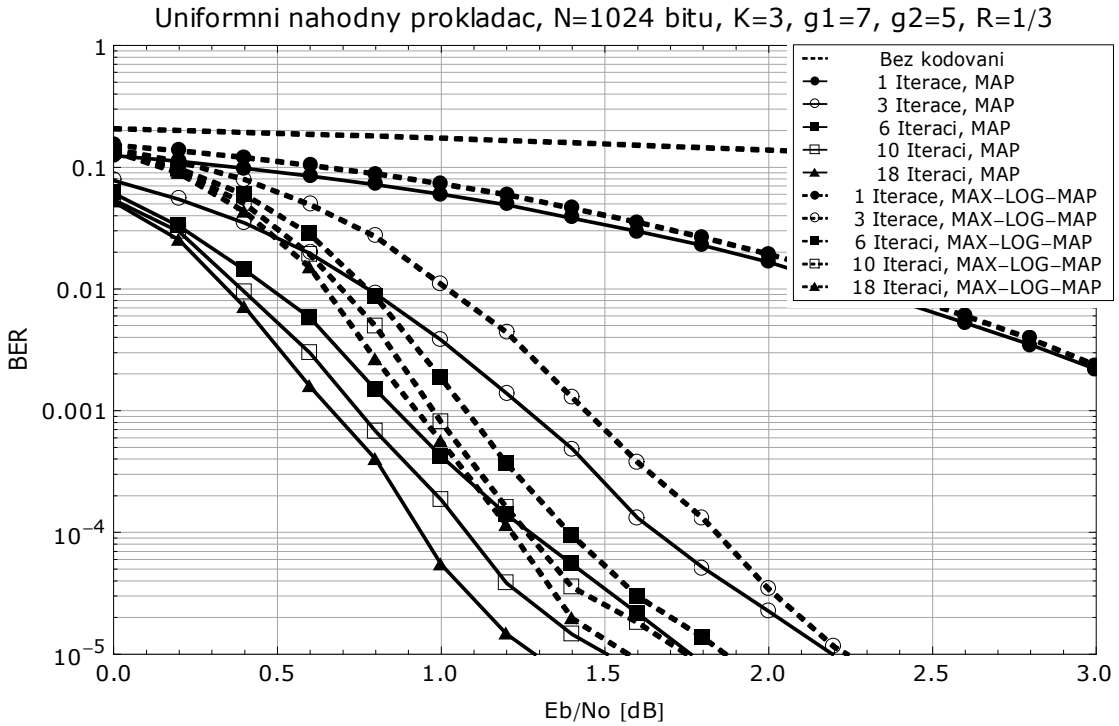
[2] N_0 - spektrální výkonová hustota šumu, E_b - střední energie na jeden bit přenášené informace.



Obr. 38. Závislost BER na počtu iterací při použití MAP algoritmu



Obr. 39. Závislost BER na počtu iterací při použití Max-Log-MAP algoritmu



Obr. 40. Srovnání Map a Max-Log-MAP algoritmu při různém počtu iterací

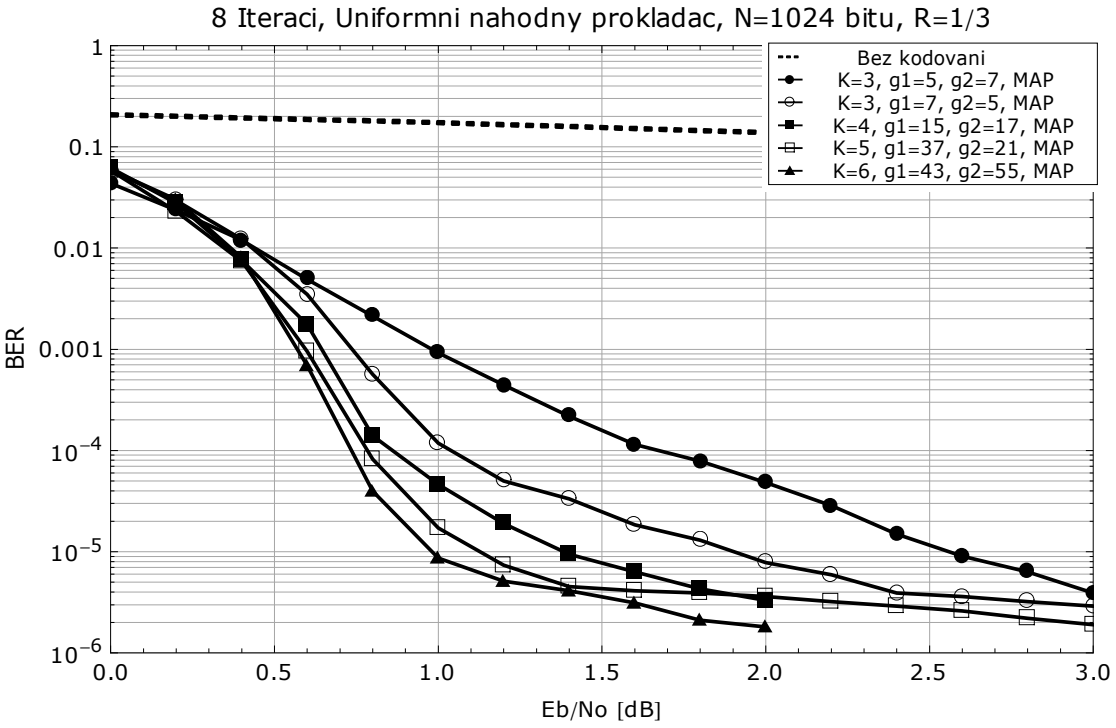
7.3.2 Vliv typu RSC kodéru na výkonnost turbo-kódů

Jak bylo zmíněno v kapitole (kap. 3.2.1), významným parametrem, který ovlivňuje minimální volnou kódovou vzdálenost turbo-kódů, je typ použitého RSC kodéru tvořeného generátory g_1 a g_2. Pro praktické ověření vlivu tohoto parametru na výkonnost turbo-kódu byly při testech využity kodéry uvedené v tabulce (Tab. 1) a navíc RSC kodér s generátory $g_1 = 5$ a $g_2 = 7$, který je využíván u klasických konvolučních kódů. Na obrázku (Obr. 41) je uvedeno srovnání výkonnosti při využití optimálních $\mathrm{RSC}[7,5]_8$ a kodérů, kde jsou hodnoty generátorů prohozeny. Je patrné výrazné zhoršení výkonnosti při využití $\mathrm{RSC}[5,7]_8$, které pro BER 10^{-4} činí asi 0,7 dB.

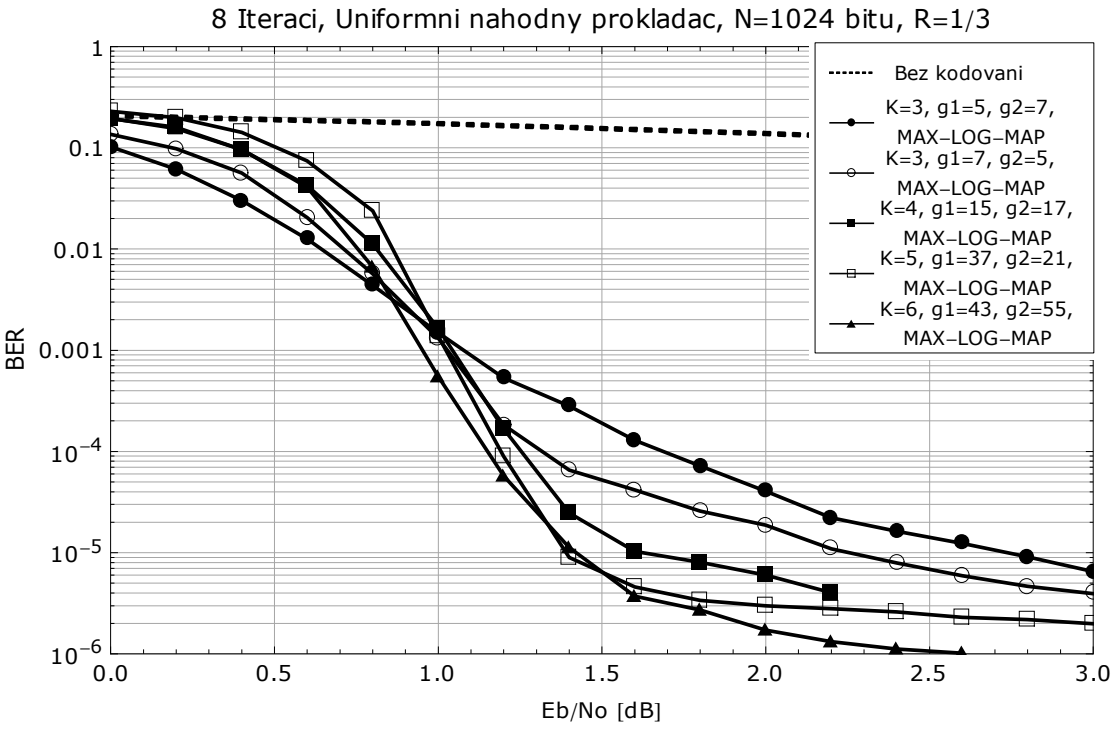
Na stejném obrázku jsou zachyceny výsledky simulace při kódovém omezení $K = 4$ a využití $\mathrm{RSC}[15,17]_8$ kodérů. Výsledný výkon byl v tomto případě zlepšen asi o 0,2 dB při BER 10^{-4}. Toto zlepšení je poskytnuto za cenu přibližně dvojnásobné dekódovací složitosti.

Při kódovém omezení $K = 5$ s kodéry $\mathrm{RSC}[37,21]_8$, které ve své práci [1] využívá i Berrou a kol., vykazuje kód zlepšení asi o 0,3 dB proti kódu s $K = 3$, přičemž složitost dekódování je čtyřnásobná.

Poslední test byl proveden pro kód s omezením $K = 6$ a $\mathrm{RSC}[43,55]_8$. Při osminásobné složitosti dekódování proti kódu s $K = 3$, je zlepšení výkonnosti asi o 0,37 dB. Ve srovnání s předchozím $K = 4$ a $K = 5$ kódem je toto zlepšení již velmi malé.

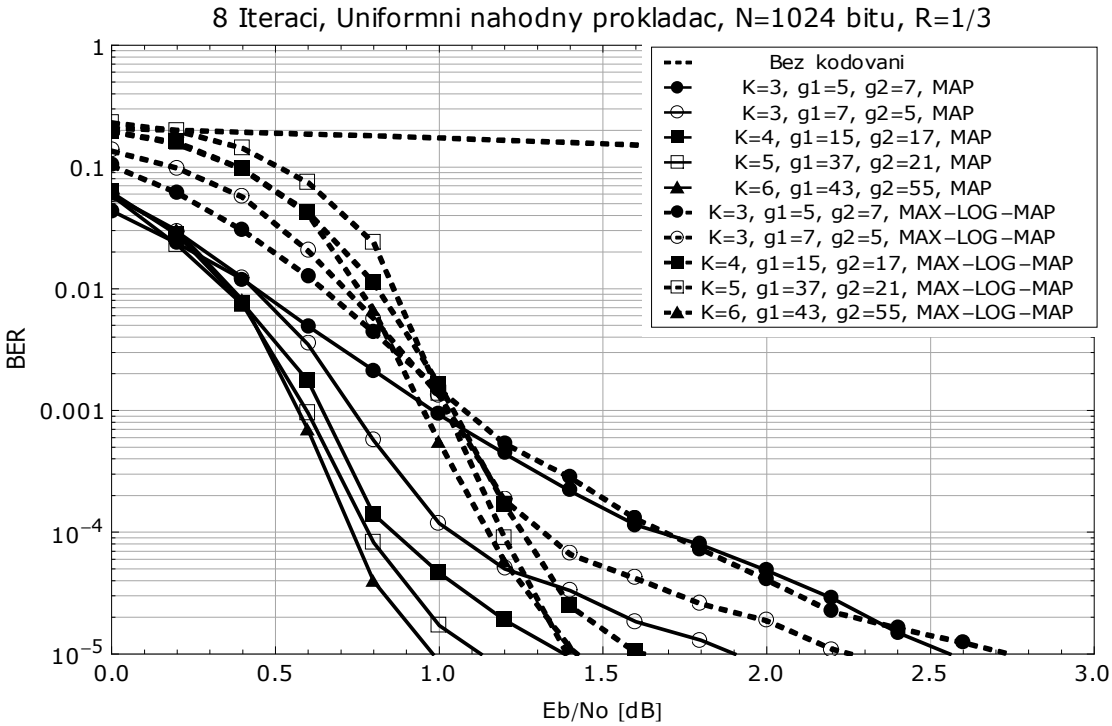


Obr. 41. Závislost BER na typu RSC při použití Map algoritmu



Obr. 42. Závislost BER na typu RSC při použití Max-Log-MAP algoritmu

Na obrázku (Obr. 42) jsou uvedeny výsledky testů chybovosti při využití Max-Log-MAP algoritmu. Pro stejné typy RSC, jaké byly využity výše, bylo na úrovni chybovosti 10^{-4} dosaženo podobných výsledků. Ze srovnání obou algoritmů, které je uvedeno na obrázku (Obr. 43), je patrné že, MAP algoritmus vykazuje vyšší výkonnost v řádu desetin decibelů.

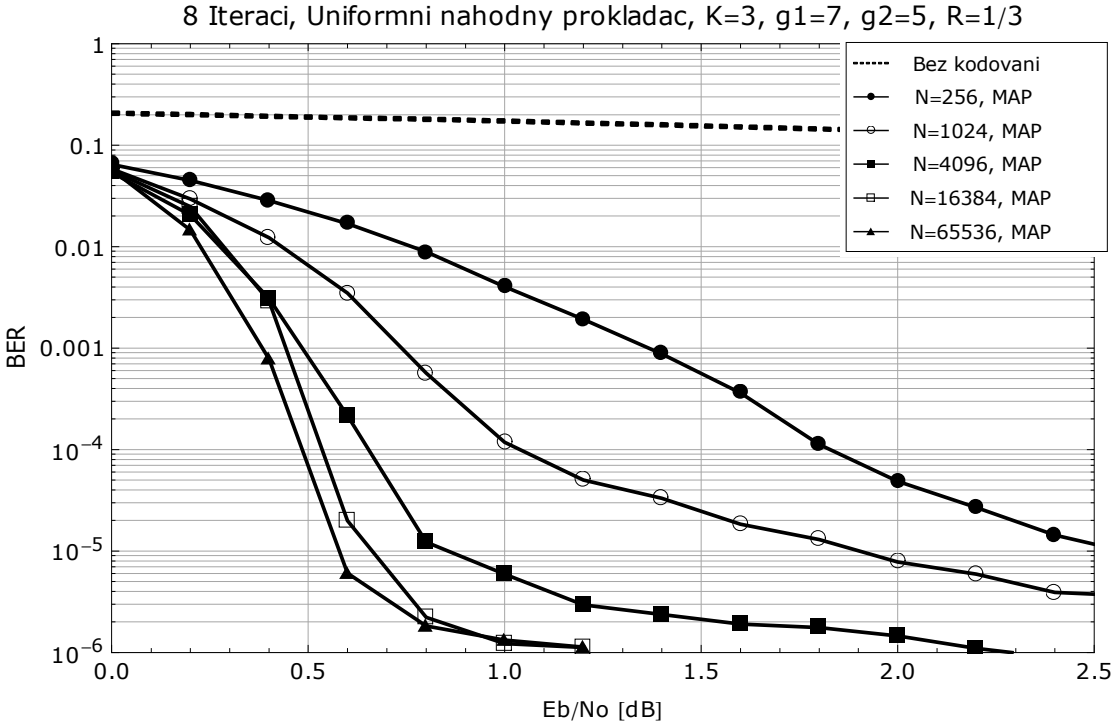


Obr. 43. Srovnání Map a Max-Log-MAP algoritmu pro různé typy RSC

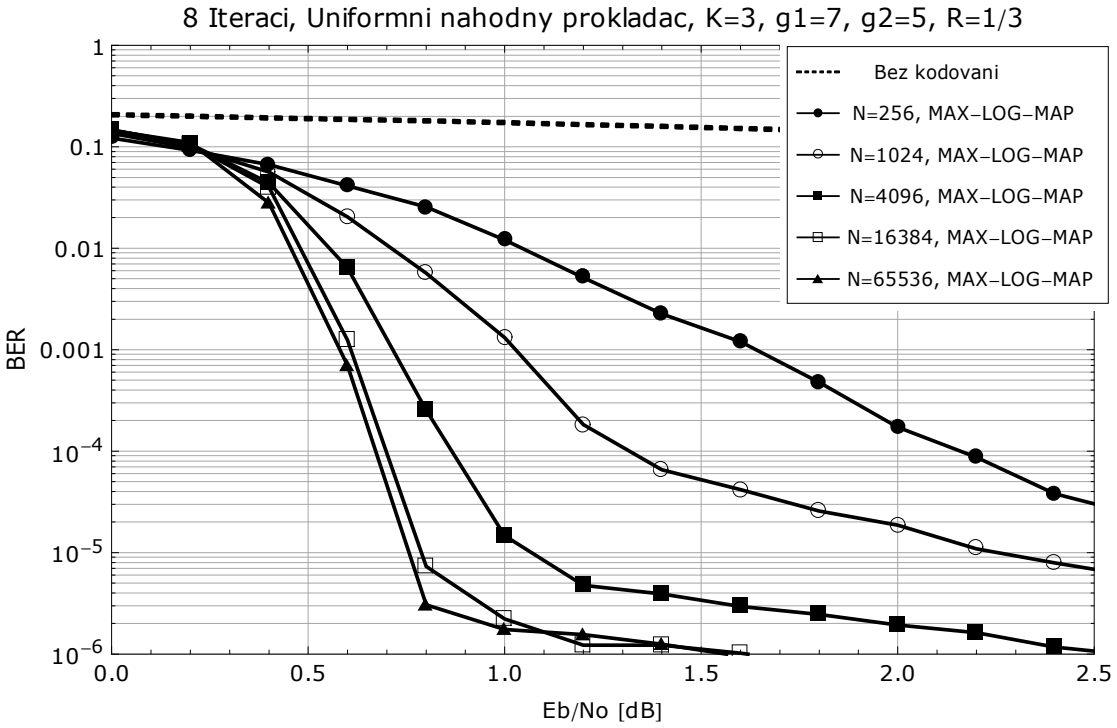
7.3.3 Vliv délky prokládané sekvence na výkonnost turbo-kódů

Pro potvrzení teoretických předpokladů vlivu délky prokládané sekvence na celkovou výkonnost turbo-kódu, byly provedeny testy pro různé hodnoty *N*. V souladu s těmito předpoklady a výsledky, které uvádí např. Berrou a kol. [1], byl ověřen vliv zvýšení délky sekvence na celkové zlepšení výkonnosti. Pro aplikace, které nevyžadují zpracování signálu v reálném čase, je výhodné používat dlouhé vstupní sekvence. Na obrázku (Obr. 44) jsou patrné velmi dobré výsledky pro N = 65536 bitů, kdy je při chybovosti 10^{-4} dosaženo hodnoty 0,5 dB. Dobré výkonnosti bylo dosaženo i pro hodnoty N = 16384 a N = 4096, kdy došlo k poklesu výkonnosti postupně o 0,05 a 0,15 dB. Pro délku N = 1024 bylo při chybovosti 10^{-4} dosaženo výkonnosti asi 1 dB. S klesající délkou prokládané sekvence výkon poměrně rapidně klesá. Některé aplikace, jako např. přenos řeči, vyžadují kratší vstupní sekvence. Při délce N = 256 bitů a chybovosti 10^{-4} bylo dosaženo hodnoty

asi 1,8 dB. Ve srovnání s jinými prakticky využívanými technikami kódování jsou tedy velmi dobré i výsledky pro menší délky vstupních sekvencí.

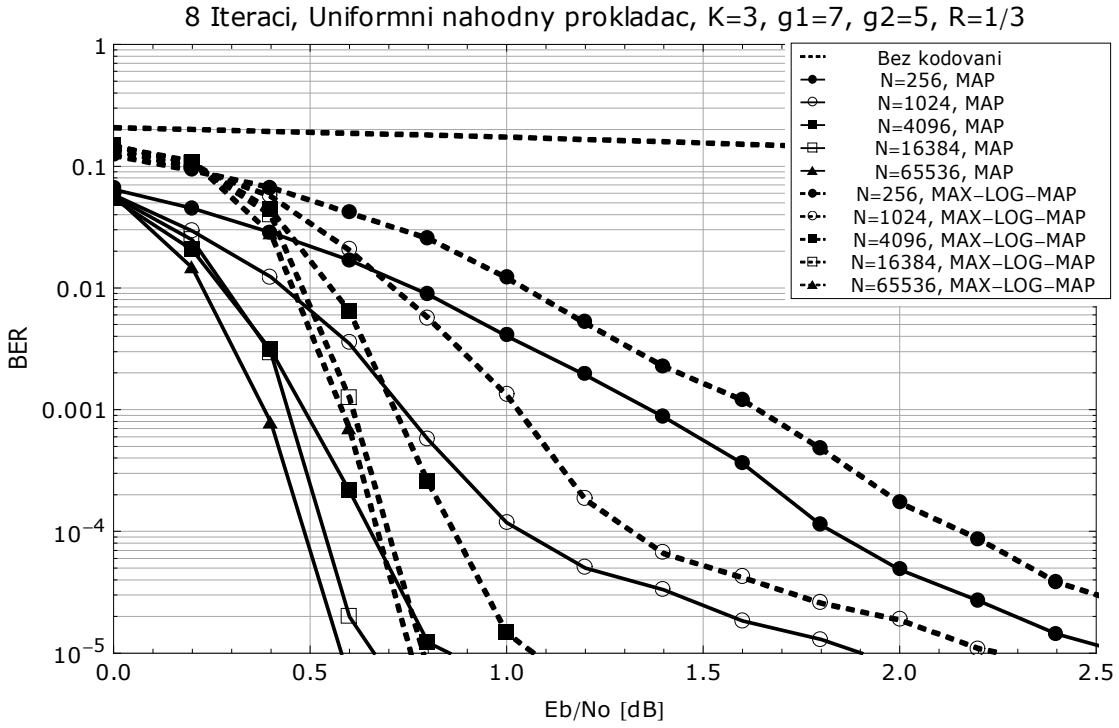


Obr. 44. Závislost BER na délce prokládané sekvence při použití Map algoritmu



Obr. 45. Závislost BER na délce prokládané sekvence při použití Max-Log-MAP algoritmu

Podobných výsledků bylo dosaženo i při použití Max-Log-MAP algoritmu, viz obrázek (Obr. 45). Na obrázku (Obr. 46) je srovnání výkonnosti obou použitých dekódovacích algoritmů.



Obr. 46. Srovnání Map a Max-Log-MAP algoritmu pro různé délky prokládané sekvence

7.3.4 Vliv typu prokladače na výkonnost turbo-kódů

Pro základní typy prokladačů, které jsou popsány v kapitole (kap. 3.4), byly provedeny testy výkonnosti. Jedné se o uniformní náhodný prokladač, semi-random prokladač, prokladač s cyklickým posuvem, neuniformní náhodný prokladač, čtvercový blokový prokladač a blokový diagonální prokladač.

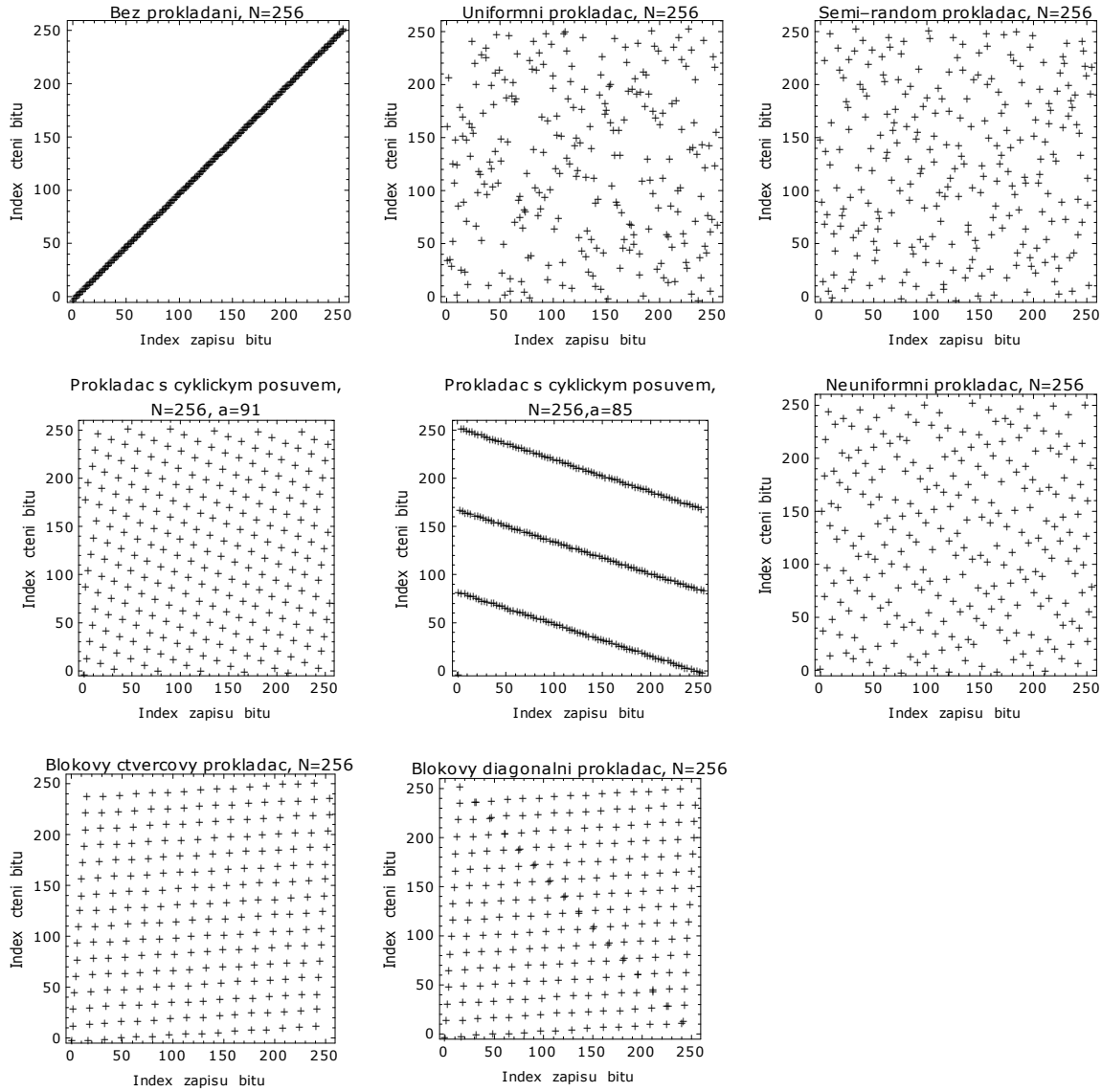
Na obrázku (Obr. 47) jsou rozptylové diagramy těchto prokladačů s velikostí $N = 256$, které ukazují, jak se mění rozprostření indexů pro čtení bitů při daných indexech zápisu. U uniformního náhodného prokladače je možné pozorovat shluky bodů na určitých souřadnicích. Tyto shluky mohou znamenat, že několik sousedních vstupních bitů se objeví na sousedních pozicích i v proložené sekvenci.

Semi-random prokladač by měl garantovat jistou minimální vzdálenost skupiny vstupních sousedních bitů v proložené sekvenci. Absence shluků naznačuje, že jsou splněny zmíněné předpoklady a celkové rozprostření indexů je rovnoměrnější.

Jedním z parametrů mapovací funkce cyklického prokladače je velikost kroku posuvu *a*, viz kapitola (kap. 3.4.5). V rámci grafické reprezentace tohoto prokladače byly zvoleny hodnoty *a* = 91 a *a* = 85. Ze srovnání obou obrázků je patrný značný vliv tohoto parametru na finální rozložení indexů.

Berrou a kol. [1] využívá ve své práci neuniformní prokladač s velikostí $N = 65536$, kde je pro výpočet indexů využita funkce $P(\xi)$, viz tabulka (Tab. 2). Pro velikost $N = 256$, byly použity hodnoty: $P(0) = 5$, $P(1) = 13$, $P(2) = 3$, $P(3) = 11$.

Rozprostření indexů obou blokových prokladačů, odpovídá poměrně jednoduchým pravidlům, podle kterých pracují jejich mapovací funkce.

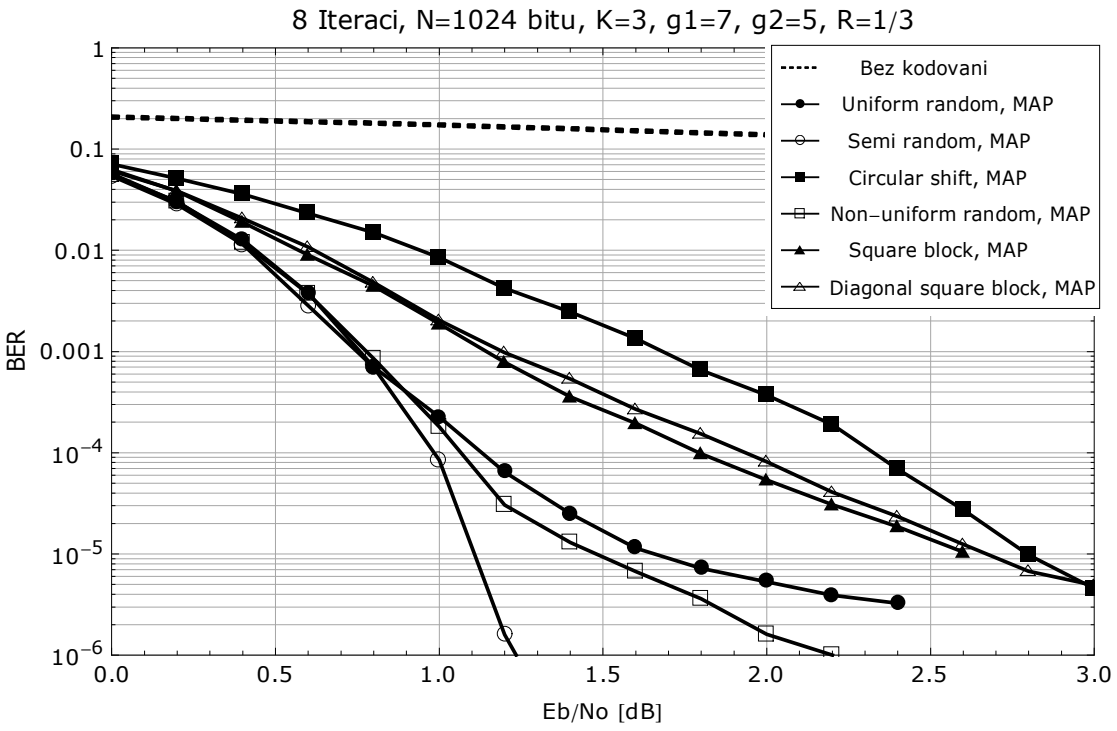


Obr. 47. Rozptylové diagramy jednotlivých typů prokladačů s velikostí N=256 bitů

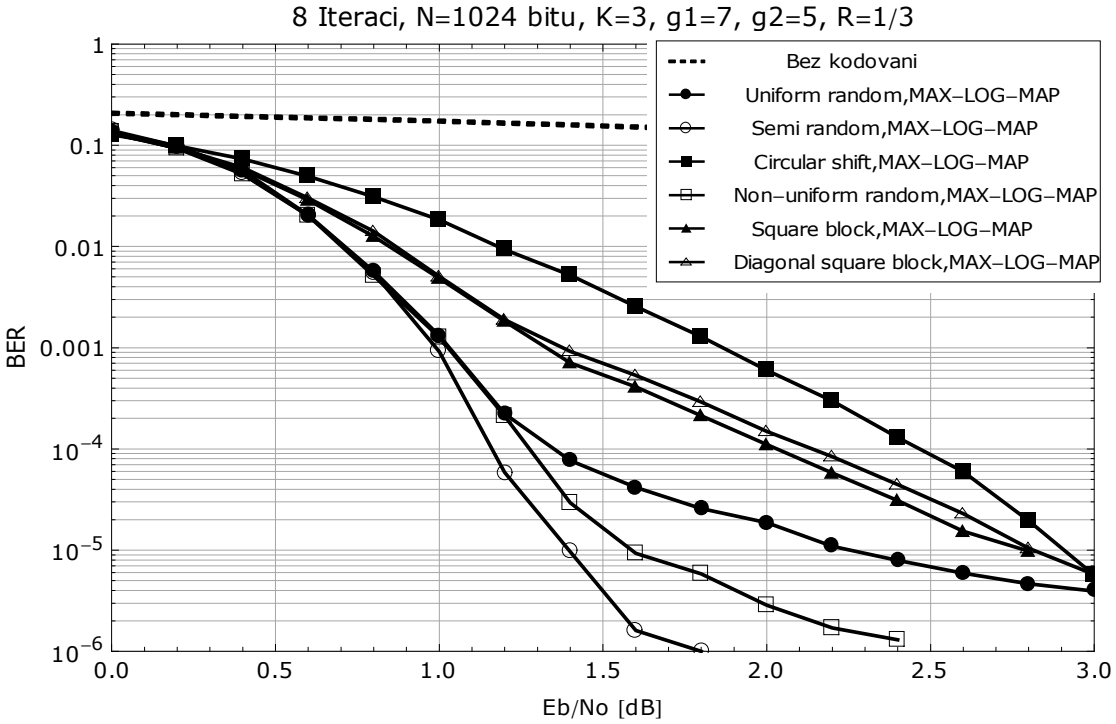
Na obrázku (Obr. 48) jsou zachyceny výsledky simulací pro různé typy prokladačů při délce prokládané sekvence N = 1024 bitů a použitém MAP dekódovacím algoritmu. Dobré vlastnosti semi-random prokladače vynikají především pro hodnoty BER $<10^{-4}$. Při BER 10^{-5} odpovídá výkonnost asi 1,2 dB, a je tak asi o 0,4 dB lepší, než u neuniformního náhodného prokladače a asi o 0,6 dB lepší, než při použití uniformního náhodného prokladače. Oba typy blokových prokladačů poskytují zhruba stejnou výkonnost, která je při chybovosti 10^{-4} asi 1,75 dB. Nejhorší výkonnost byla zaznamenána při použití prokladače s cyklickým posuvem.

Na obrázku (Obr. 49) jsou uvedeny výsledky testů chybovosti při využití Max-Log-MAP algoritmu. Pro stejné typy prokladačů bylo dosaženo podobných výsledků. Ze srovnání obou algoritmů, které je uvedeno na obrázku (Obr. 50), je stejně jako v předchozích případech patrné, že MAP algoritmus vykazuje vyšší výkonnost v řádu desetin decibelů.

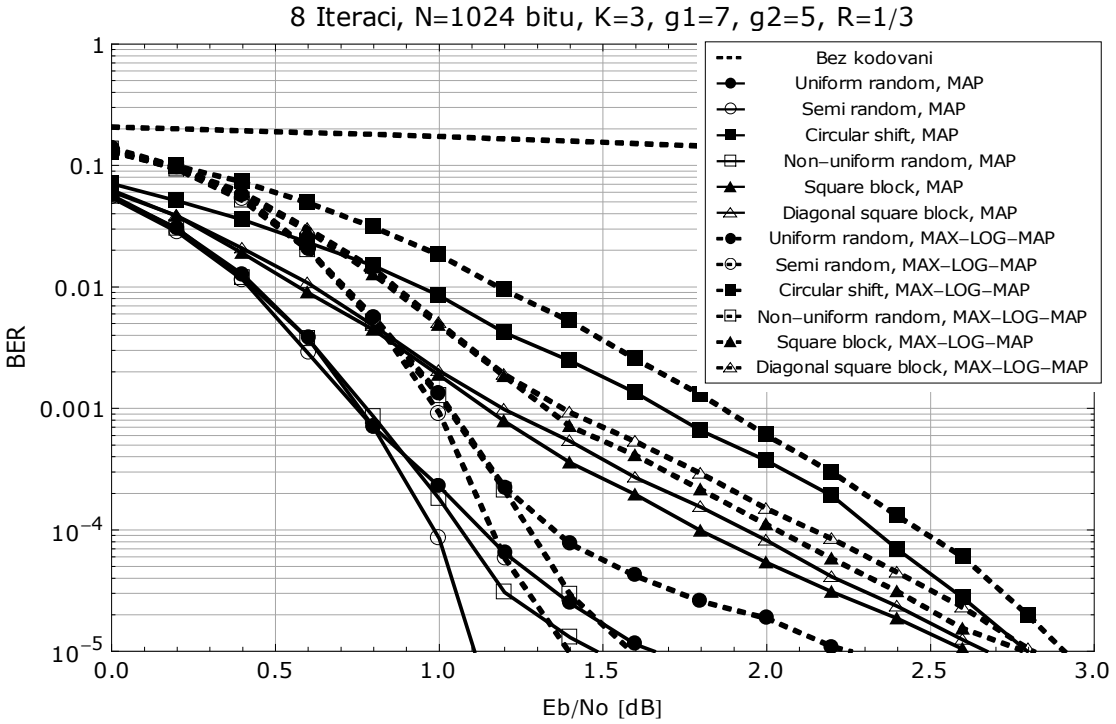
Prostřednictvím simulací byl prokázán značný vliv zvoleného typu pokladače na celkovou výkonnost turbo-kódu. Přesto, že jsou při použití jednotlivých typů viditelné poměrně velké výkonnostní rozdíly, je třeba brát úvahu také technickou náročnost hardwarové implementace, s jakou mohou být jednotlivé prokladače realizovány. Dalším souvisejícím parametrem je také doba zpoždění, která může hrát pro mnohé aplikace významnou roli.



Obr. 48. Závislost BER na typu prokladače při použití Map algoritmu



Obr. 49. Závislost BER na typu prokladače při použití Max-Log-MAP algoritmu

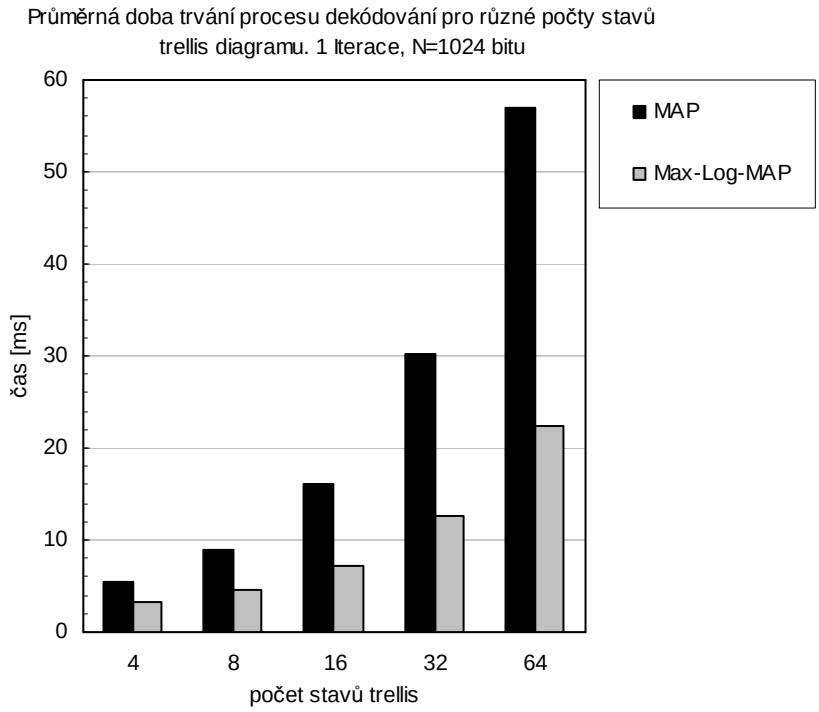


Obr. 50. Srovnání Map a Max-Log-MAP algoritmu pro různé typy prokladačů

7.4 Srovnání rychlosti dekódovacích algoritmů

Během tohoto testu byla pro každý z dekódovacích algoritmů stanovena průměrná doba potřebná na zpracování sekvence délky N = 1024 bitů v rámci jedné celé iterace při různém počtu stavů trellis diagramu. Cílem nebylo vyšetření konkrétní rychlosti dekódování, ale poměru, v jakém se liší rychlost obou dekódovacích algoritmů.

V souladu s předpokládanou výpočetní složitosti, která byla odhadnuta v tabulce (Tab. 4), bylo zjištěno, že Max-Log-MAP algoritmus umožňuje rychlejší dekódování. Rozdíl v rychlostech obou algoritmů navíc narůstá s rostoucí hodnotou kódového omezení, resp. počtem stavů trellis diagramu. Z těchto, i jiných důvodů, které byly uvedeny v teoretické části práce, je možné Max-Log-MAP algoritmus označit jako vhodný pro praktické využití.



Obr. 51. Srovnání rychlosti dekódovacích algoritmů

8 PROGRAMOVÁ IMPLEMENTACE TURBO-KODEKU

8.1 Základní informace

Pro tvorbu programové implementace turbo-kodeku a grafického uživatelského rozhraní (*GUI*) byly využiti následující knihovny a vývojové nástroje:

- *MinGW GCC 4.3.2* - kompilátor GCC
- *wxWidgets-2.8.0* - multiplatformní knihovna (toolkit) pro vytváření GUI
- *CodeLite v1.0.2893* - vývojové prostředí pro programovací jazyk C/C++
- *wxFormBuilder 3.1.67 Beta* – GUI designer pro knihovnu wxWidgets

Zdrojový kód turbo-kodeku je napsán v jazyku C a je v rámci přílohy poskytnut jak v čisté C verzi, tak ve verzi doplněné GUI, které bylo vytvořeno prostřednictvím toolkitu wxWidgets (C++) a wxFormBuilder designeru.

Důvodem pro použití jazyku C je především jeho paměťová efektivita a vysoká výpočetní rychlost, která je výhodná zejména při zpracovávání sekvencí s délkou *N* v řádu desítek tisíc bitů. Ve spojení s wxWidgets je další výhodou přenositelnost programu na jiné platformy (nutnost překladu).

8.1.1 Základní funkce programu

Na tomto místě je uveden krátký popis základních funkcí, nastavitelných maker a globálních proměnných. Kompletní, okomentovaný zdrojový kód se nachází v souboru *Turbo_Codec_DP10.c* na přiloženém CD.

Definice maker, které je možné měnit v uvedeném rozsahu (od,..,do):

```
#define N   1024                //Délka prokládané sekvence (8,..,*)
#define Iteration 8             //Počet iterací při dekódování (1,..,*)
#define Int_type 1              //Typ prokladače (1,..,6) (1-Unif.rand,
                                //2-S-rand,3-Circular shift,4-Non-unif.
                                //rand,5-Square-Block,6-Diagonal block)
#define DecAlg_type 1           //Typ algoritmu dekódování (0,1) (1-
                                //MAP,0-Max-Log-MAP)
#define sigma 1.0               //Směrodatná odchylka šumového signálu
#define INF 1e5                 //Využito jako hodnota „Infinity“
                                //v Max-Log-MAP algoritmu
```

Makra a proměnné související s parametry RSC kodéru:

```
#define m 2                     //Paměť RSC[7,5] kodéru
#define S 4                     //Počet stavů RSC[7,5] kodéru
int g1[m+1]={1,1,1};            //Generátor g1 RSC[7,5] kodéru
int g2[m+1]={1,0,1};            //Generátor g2 RSC[7,5] kodéru
```

Pozn.: tyto hodnoty lze nahradit jinými parametry libovolného RSC kodéru, je však třeba dbát na zákonitosti, které platí mezi velikostí paměti, počtem stavů a délkou či tvarem generátorů.

Prototypy a popis základních funkcí:

```
void EncoderTable();
```

Funkce provádí výpočet a uložení všech informací, které jsou obsaženy ve stavovém diagramu, viz (Obr. 7) či v tabulce (Tab. 3). Tzn., že pro každý možný stav určí nový stav, do jakého kodér přejde při vstupu log1 nebo log0, a paritní bit který je spojen s tímto přechodem. Vypočítané hodnoty jsou použity při kódování a dekódování.

```
void TurboEncoder(int *U,int *Xp1,int *Xp2,int *Interleav);
```

Funkce zakóduje vygenerovanou vstupní sekvenci. Parametry jsou ukazatele na dynamicky alokovaná pole (U – vstupní bitová sekvence, Xp1 – hodnoty zakódovaných paritních bitů kodéru RSC#1 , Xp2 – hodnoty zakódovaných paritních bitů kodéru RSC#2, Interleaver - hodnoty indexů pro čtení z prokladače)

```
void Channel(int *U,int *Xp1,int *Xp2,double *Ys1,double *Yp1,double
*Yp2,double Sigma);
```

Funkce pro simulaci přenosového kanálu. Parametry jsou ukazatele na dynamicky alokovaná pole (U – vstupní bitová sekvence, Xp1 – hodnoty odeslaných paritních bitů kodéru RSC#1 , Xp2 – hodnoty odeslaných paritních bitů kodéru RSC#2, Ys1 - hodnoty přijatých systematických symbolů, Yp1 - hodnoty přijatých paritních symbolů kodéru RSC#1, Yp2 - hodnoty přijatých paritních symbolů kodéru RSC#2, Sigma – směrodatná odchylka šumového signálu)

```
double NormalDistribution(double Sigma, double Mi);
```

Funkce vrací náhodně vygenerované reálné číslo dané normálním rozdělením s parametry (Sigma - směrodatná odchylka šumového signálu, Mi – střední hodnota). Výpočet probíhá podle *Marsaglia-Bray* algoritmu.

```
void TurboDecoder(double *Ys1,double *Yp1,double *Yp2,int
*Interleav,double Sigma, double *LLRd);
```

Funkce iterativního dekódování. Parametry jsou ukazatele na dynamicky alokovaná pole (Ys1 - hodnoty přijatých systematických symbolů, Yp1 - hodnoty přijatých paritních symbolů kodéru RSC#1, Yp2 - hodnoty přijatých paritních symbolů kodéru RSC#2, Sigma - směrodatná odchylka šumového signálu, Interleaver - hodnoty indexů pro čtení z prokladače, LLRd - finální, vypočítané hodnoty a-posteriorních LLR jednotlivých bitů po všech iteracích)

```
void MaxLogMAP(double *Ys,double *Yp, double *Lu,double Lc,int
It,double *LLR);

void MAP(double *Ys,double *Yp, double *Lu,double Lc,int It,double
*LLR);
```

Funkce Map a Max-Log-MAP algoritmu. Parametry jsou ukazatele na dynamicky alokovaná pole a pomocné proměnné (Ys - hodnoty přijatých systematických symbolů, Yp - hodnoty přijatých paritních symbolů spojených s horním nebo dolním RSC kodérem, Lu - a-priorní LLR jednotlivých bitů vypočítaný v předešlém dekodéru, Lc - spolehlivost kanálu, LLR - Vypočítané hodnoty a-posteriorních LLR)

Kromě uvedených funkcí obsahuje zdrojový kód také několik pomocných funkcí a funkce jednotlivých prokladačů.

Pozn.: obdobná funkcionalita (soubor *Turbo_Codec_GUI_DP10.h*) je využita také ve verzi doplněné GUI, pro tyto účely však bylo nutné kód částečně upravit a doplnit o další podpůrné funkce, které přímo nesouvisí s turbo-kodekem. Z tohoto důvodu doporučuji pro případné studium zdrojového kódu turbo-kodeku využívat striktně soubor *Turbo_Codec_DP10.c*.

ZÁVĚR

Cílem této práce bylo ucelení teoretických informací o základním typu turbo-kódů a programová implementace algoritmu kódování a dekódování.

Turbo-kódy představili v roce 1993 autoři Berrou, Glavieux a Thitimajshim [1], a způsobili tak převrat v oblasti digitálních komunikací. Původní koncept turbo-kódů využívá tři základní principy:

- *paralelní zřetězení rekurzivních konvolučních kódů* (vytváření dlouhých kódových slov při únosné míře složitosti kódování a dekódování)
- *prokládání* (zajištění velké váhy kódových slov, zabezpečení proti shlukům chyb)
- *zřetězení SISO dekodérů s aplikací iterativního dekódování* (výměna informací mezi oběma dekodéry a postupné zpřesňování odhadu)

Tyto vlastnosti umožnily inženýrům a výzkumným pracovníkům navrhovat komunikační systémy schopné pracovat v podmínkách blízkých teoretické limitní kapacitě kanálu. V současnosti turbo-kódy pokrývají rozsáhlou oblast praktických aplikací, která zahrnuje např. systémy pro vesmírnou komunikaci, mobilní komunikační systémy, digitální satelitní přenosy aj. Je však třeba zmínit také nepříznivou vlastnost, kterou představuje časové zpoždění vznikající v důsledku prokládání.

V úvodní části práce je vysvětlena základní problematika přenosového AWGN kanálu, nesystematických (NSC) a rekurzivních systematických konvolučních (RSC) kódů. Po krátkém seznámení se zřetězenými kódy následuje stěžejní část práce, která popisuje konstrukci a činnost turbo-kodéru se zaměřením na objasnění role prokladače. Algoritmus MAP a princip iterativního dekódování je popsán v několika rovinách, a to prostřednictvím teoretického rozboru, v pseudokódu a na podrobném příkladu. Uvedena je také krátká kapitola věnovaná algoritmům Log-MAP a Max-Log-MAP, které vznikly transformací MAP algoritmu z důvodu snížení jeho výpočetní složitosti. Závěr teoretické části obsahuje krátké pojednání o praktických aplikacích turbo-kódů.

V rámci praktické části byl v jazyku C implementován algoritmus kódování, iterativního MAP dekódování a model přenosového AWGN kanálu. Tato funkcionalita byla doplněna uživatelským rozhraním vytvořeným prostřednictvím knihovny wxWidgets. Program umožňuje simulovat činnost turbo-kodeku a testovat jeho výkonnost. Výsledky simulací,

jsou v dobrém souladu s teoretickými předpoklady, a je možné je shrnout do několika závěrů: S rostoucím počtem iterací roste také relevance vypočítaných odhadů v dekodéru a potlačení chyb je výrazně vyšší. S rostoucí velikostí kódového omezení či délkou prokládané sekvence roste výkonnost, ale také složitost turbo-kódu. Volba správného typu prokladače může výrazně ovlivnit výkonnost turbo-kódu. Výkonnost Max-Log-MAP algoritmu je v řádu desetin decibelů nižší než výkonnost algoritmu MAP, rychlost dekódování Max-Log-MAP je však výrazně vyšší, což jej činí vhodným pro praktické nasazení.

Hlavním přínosem této práce je především její vznik ve formě učebního textu psaného v českém jazyce (kterých bohužel není mnoho) a snaha o dostatečně podrobné vysvětlení základních principů s návazností na praktické výsledky simulací. Je třeba poznamenat, že problematika turbo-kódování zahrnuje celou řadu odvozených typů kódů a širokou oblast dalších tematických okruhů, které však výrazně přesahují rozsah diplomové práce. Tato práce tedy splní svůj účel v případě, že poslouží jako informační základ pro hlubší studium a výzkum problematiky turbo-kódů.

CONCLUSION

The aim of this work was to complete theoretical information about basic turbo-codes and a program implementation of encoding and decoding algorithm.

In 1993, turbo-codes were introduced by Berrou, Glavieux and Thitimajshima [1], and thus caused a revolution in digital communications. The original concept of tubo-code uses three basic principles:

- *parallel concatenation of recursive convolutional code* (long codeword generating by reasonable encoding and decoding complexity)
- *interleaving* (reducing the number of Low-weight codewords, burst error protection)
- *concatenated of SISO decoders and an iterative decoding* (the information exchange between two SISO decoders and a subsequent estimation precising)

These features allow engineers and researchers to design communication systems which are able to work in conditions close to the theoretical limit of the channel capacity. Turbo-code currently provides a vast coverage of practical applications that includes space communication systems, satellite communications, the utilization in digital video broadcasting, and other applications. There is also downside of turbo-codes which is time delay due to interleaving.

The introductory part provides an explanation of of AWGN channel basic problems, non-systematic convolutional (NSC) and recursive systematic convolutional (RSC) code. A brief introduction in to the concatenated code problems is followed by a key chapter containing a description of the turbo-enocder structure and uncovering the role of interleaver. MAP algorithm and the principle of iterative decoding is described in several levels, by means of theoretical analysis, in the pseudocode, and by the detailed example. There si also a chapter devoted to Log-MAP and Max-Log-MAP algorithms, which have originated from MAP algorithm by a transformation for reducing its computational complexity. Short discourse about practical applications of turbo-codes closes the theoretical part of this thesis.

While using C language the program implementation of encoding algorithm, the iterative decoding algorithm and the model of AWGN channel was made within this work practical part. This functionality were completed by the graphical user interface created through the

wxWidgets library. This program allows the turbo-codec function simulating and testing its performance. The simulation results are in good agreement with theoretical predictions and can be summarized as follows: The number of iterations increases, the relevance of estimates calculated by the turbo-decoder increases as well and the error correcting capability is significantly better. As the constraint length or length of interleaved sequence increases, increases performance as well but the complexity of turbo-code increases too. Choosing the right interleaving type can affect significantly the turbo-code performance. The performance of the Max-Log-MAP algorithm is in a few tenths decibel lower than performance of MAP decoding. However, the decoding speed of the Max-Log-MAP is much higher, making it suitable for practical use.

Above all the main contribution of this work is its creation in the form of study text written in Czech (not many such works occurred) and furthermore it is the pursuit of detailed explanations of basic principles in the relation to simulations results. . There is a good reason for a remark that the issue of turbo-coding include many derived types of codes and many other topics that are quite beyond the scope of this thesis. This work will fulfil its purpose if it gives fundamental information for deeper understanding and research of turbo-codes.

SEZNAM POUŽITÉ LITERATURY

[1] BERROU, C., GLAVIEUX, A., THITIMAJSHIMA, P. *Near Shannon limit error-correcting coding and decoding: Turbo-codes,* in *Proc. ICC'93*, pp. 1064-1070, Geneva, May 1993.

[2] DOBEŠ, J., ŽALUD, V. *Moderní radiotechnika* . 1. vyd. [s.l.] : BEN - technická literatura, 2006. 768 s. ISBN 80-7300-132-2.

[3] ŠEBESTA, V. *Teorie sdělování*. Brno: Vitium, 2001. 92 s. ISBN 80-214-1843-5.

[4] HEBÁK, P. *Počet pravděpodobnosti v příkladech*. 1. vyd. Praha: Státní nakladatelství technické literatury, 1978. 311 s.; ISBN (brož.).

[5] SOLEYMANI, M. R. YINGZI, G., VILAPORNSAWAI, U., *Turbo Coding for Satellite and Wireless Communications*, Kluwer Academic Publishers, Massachusets, USA, 2002, ISBN 1-4020-7197-3.

[6] KETTNER, J. *Kódování - cyklické kódy*. Bakalářská práce na Fakultě aplikované informatiky UniverzityTomáše Bati ve Zlíně. Vedoucí bakalářské práce RNDr. Ing. Miloši Krčmář, 2008. 59 s., 3 s příloh.

[7] VLČEK, K. *Komprese a kódová zabezpečení v multimediálních komunikacích*, Praha, BEN – technická literatura, 2004, ISBN 80-86056-68-6.

[8] SHU, L. COSTELLO, D., Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[9] AUDRLICKÝ, J. *Principy turbo-kódování*. Diplomová práce na Katedře elektroniky. České vysoké učení technické v Praze. Vedoucí diplomové práce Ing. Jan Šimša, CSc., 2003. 68 s.

[10] KOVACI, M., BALTA, H. G., NAFORNIȚA, M. *The Performances of Interleavers used in Turbo Codes.* Electronics and Telecommunications Faculty 2 Bd. V. Parvan, Timisoara, Romania. 4 s.

[11] BRIFFA, J. A. *Interleavers for Turbo Codes*, M.Phi1. thesis,. University of Malta, Faculty of Engineering, 1999.

[12] DOLINAR, S., DIVSALAR, D. *Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations*, JPL TDA Progress Report 42- 122, Aug. 15, 1995

[13] VUCETIC, B., JINHONG, Y. *Turbo Codes, Principles and Applications*, Kluwer Academic Publishers, 2000.

[14] BERROU, C., GLAVIEUX, A. *Near optimum error correcting coding and decoding: turbo-codes*, IEEE Transactions on Communications , vol. 44, no. 10, pp. 1261-1271, 1996.

[15] TAKESHITA, O. Y., COSTELLO, D. J. *New deterministic interleaver designs for turbo code*," IEEE Transactions on Information Theory, Vol. 46, No. 6, pp. 1988–2006, September 2000.

[16] HANZO, L., LIEW, T. H., YEAP, B. L. *Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Wireless Channels*, Wiley-IEEE Press, 2002. 768 s. ISBN 978-0-470-84726-8

[17] SCHLEGEL, C. B., PÉREZ, L. C. *Trellis and turbo coding*. New York : IEEE Press/Wiley, 2004. 393 s. ISBN 0-471-22755-2

[18] ABRANTES, S. A. *From BCJR to turbo decoding: MAP algorithms made easier*. Departamento de Engenharia Eléctrotécnica e de Computadores, 2004

[19] CLAUSSEN, H., KARIMI, H., MULGREW, B. *Improved max-log-MAP turbo decoding by maximization of mutual information transfer*, EURASIP Journal on Applied Signal Processing, vol. 6, pp. 820–827, 2005.

[20] SADJADPOUR, H. *Maximum a posteriori decoding algorithms for Turbo codes*, Invited paper in SPIE conference, Orlando, FL 2000

[21] MASERA, G., SRIPIMANWAT, K. *Turbo Code Applications: a journey from a paper to realization*, Ed. Springer Netherlands, 2005.

[22] FU-HUA, H. *Evaluation of Soft Output Decoding for Turbo Codes*. Blacksburg, Virginia, 1997. 93 s. Diplomová práce. Virginia Polytechnic Institute and State University. Dostupné z WWW: <http://scholar.lib.vt.edu/theses/available/etd-71897-15815/>.

[23] VALENTI, M. C. *Iterative detection and decoding for wireless communications* [online]. Blacksburg, Virginia, 1999. 217 s. Dizertační práce. Virginia Polytechnic Institute. Dostupné z WWW: <http://www.csee.wvu.edu/~mvalenti/documents/valenti1999a.pdf>.

[24] GAZI, O. *Parallelized structure fir low latnecy Turbo structures* [online]. [s.l.], 2007. 145 s. A thesis submited to the graduate schoul of natural and applied science. Middle East Technical University. Dostupné z WWW: <http://www.arehna.di.uoa.gr/thesis/uploaded_data/PARALLELIZED_ARCHITECTURES_FOR_LOW_LATENCY_TURBO_STRUCTURES_2007_thesis_1260199549.pdf>.

[25] *Wikipedie: Otevřená encyklopedie: Markovův řetězec* [online]. c2010 [citováno 12. 05. 2010]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Markov%C5%AFv_%C5%99et%C4%9Bzec&oldid=5317054>

[26] *Wikipedie: Otevřená encyklopedie: BER* [online]. c2009 [citováno 12. 05. 2010]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=BER&oldid=4765334>

[27] Signal-to-noise ratio (SNR), Eb/N0, Es/N0. In *Proj_tips1.pdf*. [s.l.] : [s.n.], 2009 [cit. 2010-04-29]. Dostupné z WWW: <http://www.eelab.usyd.edu.au/ELEC5507/UserFiles/File/proj_tips1.pdf>.

[28] VECEK, P. *SatCentrum : Česko-Slovenská satelitní doména* [online]. 21.08.2006 [cit. 2010-05-12]. Na internet přes satelit s DVB-RCS . Dostupné z WWW: <http://www.satcentrum.com/clanky/2042/na-internet-pres-satelit-s-dvb-rcs/>.

[29] BERROU, C., KEROUÉDAN, S,. *Scholarpedia : The free peer reviewed encyclopedia* [online]. 4 February 2008, 9 April 2010 [cit. 2010-05-12]. Turbo code. Dostupné z WWW: <http://www.scholarpedia.org/article/Turbo_code>.

[30] *NASA Official* [online]. 21.6.2005 [cit. 2010-05-13]. *Mars Reconnaissance Orbiter (Artist's Concept)*. Dostupné z WWW: <http://www.nasa.gov/mission_pages/MRO/multimedia/MRO-front-view.html>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

B	šířka pásma
C	kapacita kanálu
$d_{free,eff}$	efektivní volná vzdálenost
E_b	střední energie na jeden bit přenášené informace
g_1, g_2	generátory kodéru
G	generující matice
$H(X)$	entropie abecedy X
$H(Y)$	entropie abecedy Y
$I(X,Y)$	vzájemná informace
k	časový index
K	délka kódového omezení kódu
$L(u_k)$	Log- věrohodnostní poměr bitu u_k (a-priori)
$L(u_k \mid \underline{y})$	Log- věrohodnostní poměr bitu u_k, podmíněný přijatou sekvencí $\underline{y}$ (a-posteriori)
L_C	spolehlivost kanálu
$L_e(u_k)$	extrinsická (vnější) informace
m	paměť kodéru
M	celková paměť kodéru
n	počet výstupů kodéru
N	velikost prokladače (délka vstupní datové, sekvence)
N_0	spektrální výkonová hustota šumu
P	výkon signálu
$P(y_j \mid x_i)$	podmíněná pravděpodobnost (y_j za podmínky x_i)

$P(y_j)$	pravděpodobnost symbolu y_j
$P(y_j)$	pravděpodobnost symbolu x_i
$P(x_i \cap y_j)$	společná pravděpodobnost y_j a x_i
R , R_c	kódový poměr kódu
$S_{k-1} = `s$	stav kodéru v čase *k*-1
$S_k = s$	stav kodéru v čase *k*
u	vstupní datová sekvence
u_k	bit vstupní datové sekvence v čase *k*
w	Hammingova váha
$x_{k,l}$	výstupní bit *l*-tého výstupu kodéru v čase k
$\underline{x}$	vyslané kódové slovo (výstupní sekvence kodéru)
$x_{k,1}^S$	Systematický informační bit na výstupu *l*=1 vyslaný v čase *k*
$x_{k,l}^P$	paritní bit na výstupu *l* vyslaný v čase *k*
X	vstupní abeceda diskrétního kanálu
$y_{k,1}^S$	přijatá verze systematického bitu vyslaná v čase *k*
$y_{k,l}^P$	přijatá verze paritního bitu vyslaná v čase *k*
$\underline{y}$	přijatá sekvence
Y	výstupní abeceda diskrétního kanálu
$\alpha_k(s)$, $\gamma_k(`s,s)$ , $\chi_k(`s,s)$, $\beta_k(s)$	metriky využívané při dekódování
κ	počet vstupů kodéru
η	spektrální účinnost
π , π^{-1}	prokladač, inverzní prokladač

3GPP	3rd Generation Partnership Project
APP	A-Posteriori Probability
ARQ	Automatic Repeat Request
AWGN	Additive White Gaussian noise
BCJR	Bahl, Cocke, Jelinek, Raviv algorithm
BER	Bit Error Rate
BPKS	Binary Phase Shift Keying
BTC	Block Turbo Code
CCSDS	Consultative Committee for Space Data Systems
CDMA	Code Division Multiple Access
CTC	Convolutional Turbo Code
DSN	Deep Space Network
DVB	Digital Video Broadcasting
DVB-RCS	Digital Video Broadcast-Return Channel via Satellite
ESA	The European Space Agency
EUTELSAT	European Telecommunications Satellite Organization
FEC	Forvard Error Correction
FIR	Finite Impulse Response
GUI	Graphical User Interface
GCC	GNU Compiler Collection
IEEE	Institute of Electrical and Electronics Engineers
IIR	Infinite Impulse Response
IMT-2000	International Mobile Telecommunications by the year 2000
INMARSAT	International Maritime Satellite Organization
INTELSAT	International Telecommunications Satellite Organization

IP	Internet Protocol
ITU	International Telecommunication Union
LLR	Log-Likelihood Ratio
LTE	Long Term Evolution
MAP	Maximum A-Posteriori
MF-TDMA	Multi-Frequency Time Division Multiple Access
MIMO	Multiple-Input Multiple-Output
ML	Maximum likelihood
MRO	Mars Reconnaissance Orbiter
NASA	National Aeronautics and Space Administration
NSC	Non-Systematic Convolutional
RSC	Recursive Systematic Convolutional
SISO	Soft-Input Soft-Output
SOVA	Soft Output Viterbi Algorithm
UTMS	Universal Mobile Telecommunication System
WiMAX	Worldwide Interoperability for Microwave Access
XOR	eXclusive OR

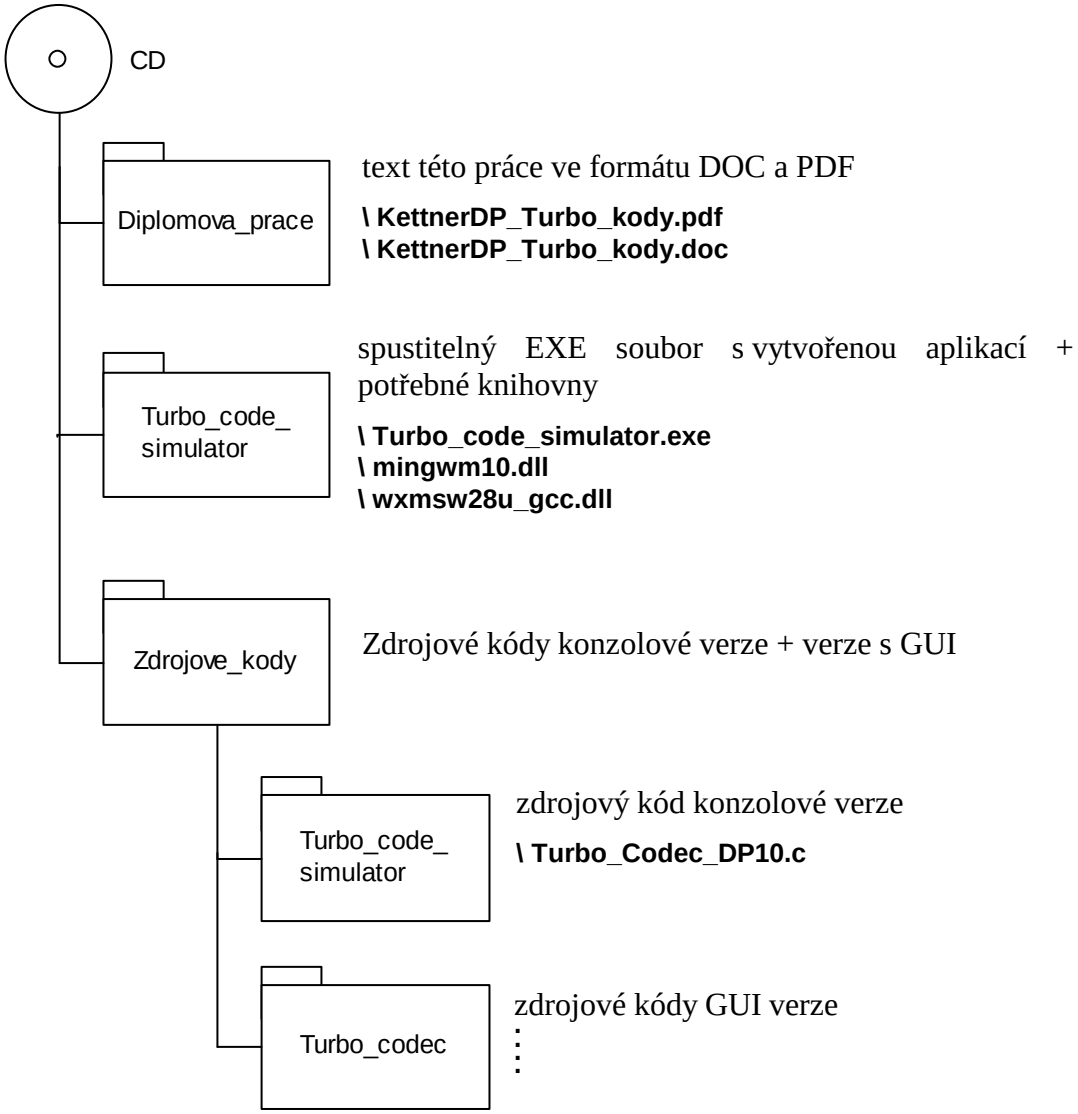
SEZNAM OBRÁZKŮ

SEZNAM TABULEK

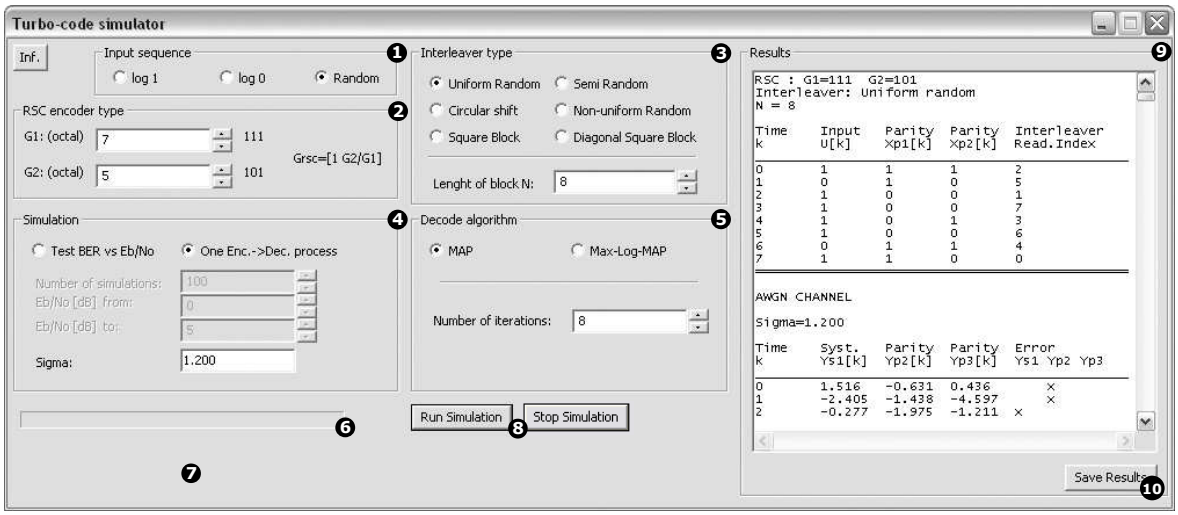
SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH PŘILOŽENÉHO CD

Adresářová struktura přiloženého CD:



PŘÍLOHA P II: OBSLUHA PROGRAMU PRO SIMULACI TURBO-KODEKU



Obr. I. Okno programu Turbo-code simulator

Na obrázku (Obr. I.) je zachyceno okno programu Turbo-code simulator. Uživatelské rozhraní tvoří několik skupinových panelů, které sdružují výběrové prvky, jimiž lze nastavit požadované parametry turbo-kódu, ovládací prvky v podobě tlačítek a výstupní prvky, které tvoří textové pole a progress bar. Význam ovládacích prvků je následující:

❶ **Input sequence** (Vstupní sekvence) - Nastavení logických hodnot, které budou tvořit vstupní informační sekvenci.

- log1 - Vstupní sekvenci tvoří hodnoty logická 1
- log0 - Vstupní sekvenci tvoří hodnoty logická 0
- Random - Vstupní sekvenci tvoří pseudonáhodně generované hodnoty log1 a log0

❷ **RSC encoder type** (Typ RSC kodéru) - Nastavení generátorů RSC kodéru.

- G1 - Hodnota generátoru g_1 vyjádřená v osmičkové soustavě
- G2 - Hodnota generátoru g_2 vyjádřená v osmičkové soustavě

❸ **Interleaver type** (Typ prokladače) - Nastavení typu a velikosti prokladače, který bude využit v kodéru a dekodéru turbo-kódu.

- Length of block N - Nastavení velikosti prokladače (délky vstupní sekvence) N
- Uniform random - Uniformní pseudonáhodný prokladač
- Semi-random - Polo-náhodný prokladač

- Circular shift - Prokladač s cyklickým posuvem
- Non-uniform random - Neuniformní pseudonáhodný prokladač (pouze pro velikosti N=64, 1024 nebo 65536)
- Square block - Čtvercový blokový prokladač (velikost N musí být beze zbytku odmocnitelná 2-ma)
- Diagonal Square block - Čtvercový blokový prokladač s diagonálním čtením (velikost N musí být beze zbytku odmocnitelná 2-ma)

❹ **Simulation** (Simulace)

- Test BER vs. E_b / N_0 - V rámci simulace bude spuštěn test výkonnosti turbo-kódu s výpisem vypočítaných hodnot BER vs. E_b / N_0, viz tabulka (Tab. II.). Je možné nastavit rozsah E_b / N_0 [dB], v jakém test proběhne (bude rozdělen na 20 hodnot) a počet simulací (opakování v rámci aktuální hodnoty E_b / N_0 [dB]), pro zvýšení statistické významnosti výsledků.
- One Enc. Dec. process - V rámci simulace bude spuštěn jeden proces kódování a dekódování s detailním výpisem vypočítaných hodnot, viz tabulka (Tab. I.). Tato volba umožňuje také nastavení libovolné směrodatné odchylky šumu σ.

❺ **Decode algorithm** (Algoritmus dekódování) - Nastavení typu alg. a počtu iterací

- MAP - Maximum A-Posteriori Algoritmus
- Max-Log-MAP - Max-Log- Maximum A-Posteriori Algoritmus
- Number of iterations - počet iterací

❻ Ukazatel průběhu simulace

❼ Textový panel pro zobrazení chybových hlášek a informací

❽ **Run Simulation** (Spustit simulaci), **Stop Simulation** (Zastavit simulaci) - Tlačítka pro spuštění a zastavení simulace

❾ **Results** (Výsledky) - Textové pole pro zobrazení výsledků simulací

❿ **Save Results** (Ulož výsledky) - Tlačítko pro uložení výsledků do textového souboru

Následující tabulka (Tab. I.) obsahuje příklad zjednodušeného výpisu výsledků při volbě (One Enc. Dec. process) s N = 8. V tabulce (Tab. II.) je uveden příklad výpisu výsledků při volbě (Test BER vs. E_b / N_0).

Tab. I. Příklad výstupu programu – One Enc. Dec. Process

```
ENCODER

RSC : G1=111  G2=101
Interleaver: Uniform random
N = 8

Time    Input   Parity  Parity  Interleaver
k       U[k]    Xp1[k]  Xp2[k]  Read.Index
_____________________________________________
0       1       1       1       2
1       0       1       0       5
2       1       0       0       1
3       1       0       0       7
4       1       0       1       3
5       1       0       0       6
6       0       1       1       4
7       1       1       0       0
=============================================
AWGN CHANNEL

Sigma=1.200

Time    Syst.   Parity  Parity  Error
k       Ys1[k]  Yp2[k]  Yp3[k]  Ys1 Yp2 Yp3
_____________________________________________
0       1.516   -0.631  0.436       x
1       -2.405  -1.438  -4.597      x
2       -0.277  -1.975  -1.211  x
3       0.532   -0.500  -0.678
4       0.810   -0.566  0.504
5       -0.843  -0.818  -1.498  x
6       -0.797  0.073   -0.540          x
7       0.914   0.039   -2.223
=============================================
DECODER

Decoding algorithm: MAP
Iteration = 8

================ Iteration 1 ===========

DEC#1

k       (s`,s)  U[k]    Gamma[k](s`,s)<-Uk
_____________________________________________
0       (0,0)   0       0.541
0       (1,2)   0       0.541
0       (2,3)   0       0.225
0       (3,1)   0       0.225
0       (0,2)   1       1.849
0       (1,0)   1       1.849
0       (2,1)   1       4.442
0       (3,3)   1       4.442

⋮         ⋮     ⋮         ⋮

7       (2,1)   1       1.836
7       (3,3)   1       1.836


k       (s)     Alpha[k](s)
_____________________________________________
0       (0)     1.000
0       (1)     0.000
0       (2)     0.000
0       (3)     0.000

⋮         ⋮       ⋮

8       (2)     0.219
8       (3)     0.327


k       (s)     Beta[k](s)
_____________________________________________
0       (0)     0.122
0       (1)     0.274
0       (2)     0.467
0       (3)     0.137
⋮         ⋮       ⋮
8       (3)     0.000
```

- sekce s výsledky kódování
- nastavené parametry turbo-kódu
- hodnoty systematických bitů, paritních bitů a indexů pro čtení z prokladače
- sekce s výstupem AWGN kanálu
- hodnoty přijatých systematických symbolů (bitů) a paritních symbolů (bitů). Případné „x“ odpovídá chybě na dané pozici, která by zůstala neodhalena při přímé aplikaci tvrdého rozhodnutí
- sekce s hodnotami vypočítanými v procesu dekódování
- hodnoty vypočítané v 1. iteraci v DEC#1
- hodnoty metriky přechodu $\gamma_k(`s, s)$ pro všech $N$ (8) bitů a pro všechny možné přechody ze stavu $`s$ do s při vstupním bitu $u_k = 0$ nebo $u_k = 1$
- hodnoty dopředné metriky $\alpha_k(s)$ pro všech N (8) bitů a pro všechny možné stavy s
- hodnoty zpětné metriky $\beta_k(s)$ pro všech N (8) bitů a pro všechny možné stavy s

DEC#2

k	(s`,s)	U[k]	Gamma[k](s`,s)<-Uk
0	(0,0)	0	1.107
⋮	⋮	⋮	⋮
7	(3,3)	1	3.614

k	(s)	Alpha[k](s)
0	(0)	1.000
⋮	⋮	⋮
8	(3)	0.156

k	(s)	Beta[k](s)
0	(0)	0.363
⋮	⋮	⋮
8	(3)	0.250

- hodnoty vypočítané v 1. iteraci v DEC#2

Extrinsic information

k	Le1[k]	Le2[k]	Le2_inv[k]
⋮	⋮	⋮	⋮

- hodnoty extrinsické informace $L_{e11}(u_k)$ z DEC#1 a $L_{e21}(u_k)$ z DEC#2

Decisions

Time k	Soft D. L(u[k]|y)	Hard D. U[k]	Error
⋮	⋮	⋮	⋮

- hodnoty LLR $L(u_k \mid \underline{y})$ pro bit u_k a dílčí, tvrdé rozhodnutí o hodnotě bitu u_k. Případné „x“ odpovídá neopravené chybě.

========== Iteration 2 ==========

⋮

========== Iteration 8 ==========

⋮

- hodnoty $\gamma_k(\grave{s},s)$, $\alpha_k(s)$, $\beta_k(s)$, $L_{e1i}(u_k)$, $L_{e2i}(u_k)$ a $L(u_k \mid \underline{y})$ vypočítané v dalších iteracích

Extrinsic information

k	Le1[k]	Le2[k]	Le2_inv[k]
0	3.314	10.867	3.075
1	-5.579	9.958	-5.550
2	7.028	-5.550	10.867
3	5.656	6.232	2.029
4	9.348	2.029	2.242
5	7.616	-4.284	9.958
6	-5.299	2.242	-4.284
7	5.440	3.075	6.232

- hodnoty extrinsické informace $L_{e18}(u_k)$ z DEC#1 a $L_{e28}(u_k)$ z DEC#2 po 8-mé iteraci

Decisions

Time k	Soft D. L(u[k]|y)	Hard D. U[k]	Error
0	8.495	1	
1	-14.470	0	
2	17.511	1	
3	8.424	1	
4	12.716	1	
5	16.404	1	
6	-10.689	0	
7	12.941	1	

- finální hodnoty LLR $L(u_k \mid \underline{y})$ pro bit u_k a tvrdé rozhodnutí o hodnotě bitu u_k. Případné „x“ odpovídá neopravené chybě.

Tab. II. Příklad výstupu programu – Test BER vs. E_b / N_0

```
BER TEST

RSC : G1=111  G2=101
Interleaver: Semi-random
N = 1024

Decoding algorithm: MAP
Iteration = 7

Eb/No   BER            Sigma   Errors
[dB]                           AVG
_____________________________________
0.000   0.0574267578   1.225   58.805000
0.200   0.0292373047   1.197   29.939000
0.400   0.0113984375   1.170   11.672000
0.600   0.0036416016   1.143   3.729000
0.800   0.0006894531   1.117   0.706000
1.000   0.0000654297   1.092   0.067000
1.200   0.0000166016   1.067   0.017000
1.400   0.0000078125   1.042   0.008000
1.600   0.0000039063   1.019   0.004000
1.800   0.0000000000   0.996   0.000000
2.000   0.0000000000   0.973   0.000000
2.200   0.0000000000   0.951   0.000000
2.400   0.0000000000   0.929   0.000000
2.600   0.0000000000   0.908   0.000000
2.800   0.0000000000   0.887   0.000000
3.000   0.0000000000   0.867   0.000000
3.200   0.0000000000   0.847   0.000000
3.400   0.0000000000   0.828   0.000000
3.600   0.0000000000   0.809   0.000000
3.800   0.0000000000   0.791   0.000000
4.000   0.0000000000   0.773   0.000000

 AVG Decoding Duration = 38.580ms
```

- výsledky testu výkonnosti turbo kódu
- nastavené parametry turbo-kódu
- hodnoty poměru $\frac{E_b}{N_0}$ [dB] a odpovídající chybovost BER, směrodatná odchylka šumu v AWGN kanálu a průměrný počet chyb v rámci všech simulací
- průměrná doba trvání dekódování