Софтуерна система за симулационно изследване на конволюционни кодери и декодери

Адриана Бороджиева

A Software System for Simulation Investigating Convolutional Encoders and Decoders: *The paper presents a software system developed for investigating convolutional encoders and decoders in conditions of noise in the digital communication systems. The system will be applied in the educational process in the course "Coding in Telecomunication systems". The work with the system will enable students to comprehend the principle of convolutional encoding and decoding based on Viterbi algorithm.*

Key words: *Convolutional Encoders and Decoders, Hard-decision and Soft-decision decoding, MATLAB and Communications Toolbox.*

ВЪВЕДЕНИЕ

Откриването и коригирането на грешки в телекомуникациите и теорията на информацията има голямо практическо значение при поддържане на целостта на данните, предавани по зашумени канали и записвани в ненадеждни среди за съхраняване [7].

Известни са два основни подхода за проектирането на шумоустойчив код и протокол за система, коригираща грешките: автоматична заявка за повторение (Automatic repeat-request, ARQ) и пряка корекция на грешките (Forward error correction, FEC). Възможно е тези два режима да се комбинират така, че по-малките грешки да се коригират без повторно предаване, а по-големите грешки само да се откриват и да се изисква повторно предаване. Комбинацията от тях се нарича хибридна автоматична заявка за повторение (hybrid automatic repeat-request) [7].

В FEC-системите, предавателят кодира данните с код, коригиращ грешки и изпраща кодираното съобщение. Приемникът декодира приеманата информация в "най-вероятните" данни и никога не предава съобщения обратно до предавателя.

Целта на пряката корекция на грешките е да се подобри капацитетът на комуникационния канал чрез добавяне на излишна информация към предаваните данни. Двете основни форми на каналното кодиране са блоковото и конволюционното кодиране.

В доклада е описана софтуерна система за симулационно изследване на конволюционни кодери и декодери, които притежават най-добри коригиращи способности при условия на шум в цифровите комуникационни системи.

1. КОНВОЛЮЦИОННИ КОДОВЕ

В телекомуникациите, конволюционният код е вид код, коригиращ грешки, при който всеки k-битов информационен символ (всеки k-битов стринг), който ще се кодира, се трансформира в n-битов символ, където k/n е скоростта на кода (code rate) ($n \geq k$), а трансформацията е функция на последните L информационни символа, където L е дължината на кодовото ограничение на кода [1, 2, 7].

Конволюционното кодиране с декодиране по алгоритъма на Витерби широко се използва в съвременните комуникационни системи. Конволюционните кодове често се използват с цел подобряване на характеристиките на цифровото радио, мобилните телефони, сателитните комуникации и Bluetooth реализациите.

2. СОФТУЕРНА СИСТЕМА ЗА ИЗСЛЕДВАНЕ НА КОНВОЛЮЦИОННИ КОДЕРИ

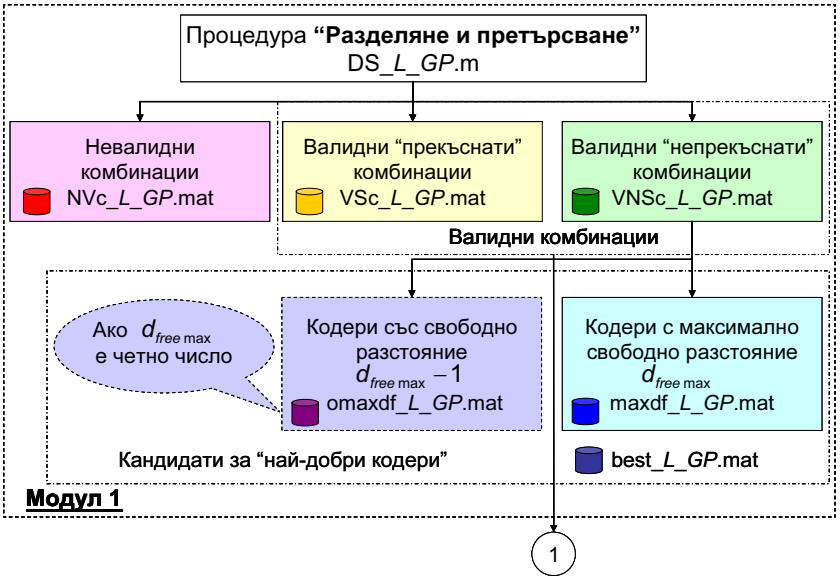
Анализът при симулационното изследване на конволюционни кодери чрез разработената и описаната в [5] система, показва, че е възможно, при условията на експеримента, друг кодер, различен от посочения в таблиците с "най-добри" кодери в специализираната литература [1, 2], да се явява "най-добрият", т.е. да допуска най-малко сгрешени битове, измервано чрез стойността на параметъра BER (Bit Error Rate). Анализът показва, че "кандидати за най-добри кодери" могат да са

кодерите с максимална стойност на параметъра свободно разстояние $d_{free\,\max}$, а ако този параметър е четно число, тогава "кандидати за най-добри кодери" могат да бъдат и кодерите с декрементираната с 1 стойност на свободното разстояние $d_{free\,\max} - 1$. Това налага усъвършенстване на разработената софтуерна система, като в отделна "база от данни" се съхраняват и кодерите със свободно разстояние $d_{free\,\max} - 1$ (ако $d_{free\,\max}$ е четно число).

При използване на изчислителната и графична среда MATLAB под WINDOWS и разширенията Communications Toolbox и Symbolic Math Toolboxes е разработена софтуерна система за симулационно изследване на конволюционни кодери, съставена от следните модули: разделяне и претърсване, конволюционно кодиране и декодиране, определяне на работните характеристики на кодерите. Обект на разглеждане в доклада са първите два модула на системата.

Първият модул (фиг. 1) съдържа множество разработени скриптове DS_*L*_*GP*.m, които изпълняват процедурата "разделяне и претърсване", включваща процесите "разделяне" на комбинациите на генераторните полиноми на валидни и невалидни, и "претърсване" на валидните комбинации с цел откриване на кодерите, притежаващи максимална стойност на параметъра свободно разстояние $d_{free\,\max}$. Параметърът $d_{free\,\max}$ се явява мярка за коригиращите свойства на кодера [2].

В отделна "база от данни" се съхраняват и кодерите със свободно разстояние $d_{free\,\max} - 1$ (при четно $d_{free\,\max}$), които също могат да са "кандидати за най-добри кодери". Скриптовете DS_*L*_*GP*.m позволяват да се обходят всички възможни комбинации за кодери със скорост на кодиране $R_c = 1/n$, $n = 2 \div 5$, при указана дължина на кодовото ограничение *L* и указан брой на генераторните полиноми *GP*.



Фиг. 1. Архитектура на разработената и доусъвършенствана софтуерна система за симулационно изследване на конволюционни кодери – модул 1

След обхождане на всички възможни комбинации за полиномни генератори, те се разделят на две групи: 1) Невалидни комбинации, които се съхраняват в "база от данни" NVc_*L_GP*.mat. Това са комбинации, за които изходните кодови символи на кодера не се определят от текущия входен бит на съобщението или от най-крайния десен тригер на регистъра, генериращ конволюционния код, с което се нарушава условието за дефиниране на параметъра дължина на кодовото ограничение на кода; 2) Валидни комбинации, които също се разделят на две подгрупи: валидни "прекъснати" комбинации, съхранявани в "база от данни" VSc_*L_GP*.mat и валидни "непрекъснати" комбинации, съхранявани в VNSc_*L_GP*.mat. За първите симулацията се прекъсва след изтичане на фиксирано време в опит да се определи параметърът свободно разстояние, а за вторите симулацията по "разделяне и претърсване" е приключила успешно, като е определен техният параметър свободно разстояние. В отделна "база от данни" VNSc_*L_GP*_df.mat е съхранена информация за съответната стойност на параметъра свободно разстояние, за всички валидни "непрекъснати" комбинации.

От всички валидни "непрекъснати" комбинации, в отделна "база от данни" maxdf_*L_GP*.mat се записват онези комбинации, за които се получава максимална възможна стойност на параметъра свободно разстояние $d_{free\max}$. Ако $d_{free\max}$ е четно число се генерира и "базата от данни" omaxdf_*L_GP*.mat, в която се съхраняват кодерите със свободно разстояние $d_{free\max}-1$. Тези две "бази от данни" формират "базата от данни" best_*L_GP*.mat, в която се съхраняват всички възможни "кандидати за най-добри кодери".

До момента, процедурата "разделяне и претърсване" е реализирана за следните варианти: $L=3$ и $GP=2\div5$, $L=4$ и $GP=2\div5$, $L=5$ и $GP=2\div4$.

В табл. 1, за тази варианти, е посочен броят на всички възможни комбинации на кодери (N), на невалидните комбинации (N_{NV}), на валидните "прекъснати" (N_{VS}), на валидните "непрекъснати" (N_{VNS}), на кодерите с максимална стойност на параметъра свободно разстояние $d_{free\max}$ (N_{maxdf}), на кодерите със свободно разстояние $d_{free\max}-1$ (N_{omaxdf}) и на всички кодери, които се явяват "кандидати за най-добри кодери" (N_{best}).

Табл. 1. Параметри на конволюционните кодери при известна дължина на кодовото ограничение *L* и известен брой на генераторните полиноми *GP*

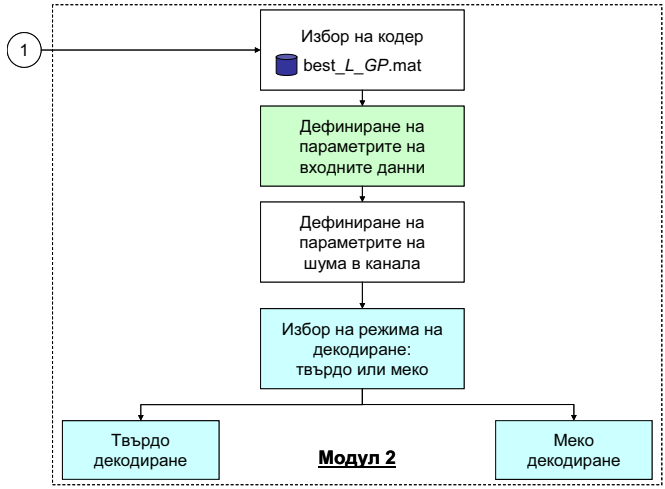
L_GP	*N*	N_{NV}	N_{VS}	N_{VNS}	N_{maxdf}	N_{omaxdf}	N_{best}
3_2	**49**	17	4	28	2	0	**2**
3_3	**343**	53	8	282	3	15	**18**
3_4	**2401**	161	16	2224	34	112	**146**
3_5	**16807**	485	32	16290	10	0	**10**
4_2	**225**	89	14	122	16	32	**48**
4_3	**3375**	659	52	2664	12	54	**66**
4_4	**50625**	4721	170	45734	24	0	**24**
4_5	**759375**	33371	484	725520	830	3340	**4170**
5_2	**961**	401	50	510	2	14	**16**
5_3	**29791**	6407	245	23139	6	72	**78**
5_4	**923521**	98849	1506	823166	54	528	**582**

При използване на втория модул на разработената система (фиг. 2), реализиращ процедурата "конволюционно кодиране и декодиране по алгоритъма на Витерби" (с меко или твърдо вземане на решения), са изследвани всички кодери от "базата от данни" best_*L_GP*.mat, като за конкретните условия на симулационния експеримент действително се открива "най-добрият" – този, който допуска най-малко

сгрешени битове.

Броят на анализираните кодери с дължина на кодовото ограничение $L = 3$ и с брой на генераторните полиноми $GP = 2$ е 32. Получените симулационни резултати са дадени в [3, 4]. Броят на кодерите с дължина на кодовото ограничение $L = 3$ и с брой на генераторните полиноми $GP = 3$ е 290. Симулационните резултати за "най-добрите" и за "най-лошите" кодери са дадени в [5].

Резултатите от симулационното изследване на кодерите чрез втория модул на системата, получени след сортиране по нарастващ ред, за дължина на кодовото ограничение *L* и брой на генераторните полиноми *GP*, при *L* = 3 и *GP* = 5 и при *L* = 5 и *GP* = 2, са представени в табл. 2 и табл. 3. Всички кодери от табл. 2 имат една и съща стойност (13) на параметъра свободно разстояние, а кодерите от табл. 3 са със стойност 8 или 7 на параметъра свободно разстояние.



Фиг. 2. Архитектура на разработената софтуерна система за симулационно изследване на конволюционни кодери – модул 2

Алгоритмите, заложени във втория модул на софтуерната система, са представени в детайли в [3] и [4].

Всички кодери са поставени при едни и същи условия за симулационно изследване на процесите на кодиране и декодиране (твърдо или меко), с цел анализ и сравнение на получените резултати, а именно [6]:

1) Броят на битовете на съобщението, подлежащо на конволюционно кодиране и последващо предаване по комуникационния канал, е *k* = 1 000 000 бита, двоични данни, съхранени в променливата *data*, генерирани чрез *data* = *randint*(*k*,1,2,135); случайният генератор на цели числа *randint* е установен първоначално със стойност 135 (*state* = 135) с цел получаване на едни и същи генерирани данни всеки път по време на симулацията.

2) Дължината на обратния път (trace-back length) на кодера *tblen* се изчислява по формулата: *tblen* = 6*(*sum*(*L*) + 1), където *L* е дължината на кодово ограничение на кодера [2, 6].

3) Конвертирането на параметрите на кодера (*L* и *GP*) в решетка се извършва чрез прилагане на функцията *poly2trellis*, и е необходимо за работата на функциите *convenc* и *vitdec*. Конволюционното кодиране на двоичните данни се реализира чрез реда: *coded_data* = *convenc*(*data*, *trellis*), където *trellis* е решетъчното представяне на кодера.

4) За режим на декодиране с меко вземане на решения: добавянето на бял Гаусов шум към кодираните данни се реализира чрез инструкцията *ncoded_data = awgn* (*coded_data*,6, *'measured'*, 135), като първоначалното състояние на генератора на адитивен бял Гаусов шум (additive white Gaussian noise – AWGN) също е установено на стойност 135; опцията *'measured'* определя факта, че AWGN измерва мощността на сигнала преди добавяне на шума, а отношението сигнал-шум (signal-to-noise ratio, SNR) е 6 dB; квантуването на кодираните данни, представянето на квантуваните данни чрез три бита и реализирането на декодиране по алгоритъма на Viterbi при меко вземане на решение се реализират чрез редовете:

qcoded_data = quantiz(*ncoded_data*, [0.001,.1,.3,.5,.7,.9,.999]);

decoded_data = vitdec(*qcoded_data*, *trellis*, *tblen*, *'cont'*, *'soft'*, 3);

Опцията *'cont'* предполага, че кодерът започва от нулево състояние, декодерът се връща обратно от състоянието с най-добра метрика и се използва закъснение *delay*, равно на *tblen* символа. Режимът непрекъснато действие на функцията (опция *'cont'*) предизвиква закъснение *delay*, равно на дължината на обратния път, така, че *data* (1) не съответства на *decoded_data* (1), а на *decoded_data* (*tblen* + 1). Следователно, при изчисляването на параметъра степен на битовата грешка (bit error rate, BER) (когато се сравняват декодираните данни и оригиналното съобщение), закъснението при декодиране трябва да се отчете:

[*number*, *ratio*] = *biterr*(*decoded_data*(*delay* + 1 : *end*), *data*(1 : *end* – *delay*).

5) За режима на декодиране с твърдо вземане на решения: за добавяне на шум към кодираните данни се използва редът:

ncoded_data = rem (*coded_data* + *randerr* (*length*(*coded_data*), 1,...

[0 1;.95 .05], 135), 2),

като първоначалното състояние на генератора на образци на грешки също е установен на стойността 135; декодирането по алгоритъма на Viterbi с твърдо вземане на решения се реализира чрез реда:

decoded_data = vitdec(*ncoded_data*, *trellis*, *tblen*,*'cont'*,*'hard'*).

Оказва се, че кодерите с генератори [*i ... j*], [*j ... i*] и т.н., т.е. с еднакви, но разместени генератори, нямат едни и същи характеристики (параметър BER), а сравнително близки (табл. 2 и табл. 3). Например, при режим на меко декодиране кодерът [27 25] сгрешава 2 006 от 1 000 000 предадени бита, а кодерът [25 27] допуска 2 054 грешки (в повечето случаи по-малко от съответните стойности при режим на твърдо декодиране).

Табл. 2. Резултати от симулационното изследване на кодери с дължина на кодовото ограничение *L* = 3 и с брой на генераторните полиноми *GP* = 5

Брой предадени битове: 1000000					
Твърдо декодиране			**Меко декодиране**		
Кодер	**Брой гр. битове**	**BER**	**Кодер**	**Брой гр. битове**	**BER**
[5 7 7 7 5]	0	0	[5 7 5 7 7]	0	0
[7 5 7 5 7]	2	0,000002	[7 5 7 7 5]	0	0
[7 5 5 7 7]	3	0,000003	[7 7 5 7 5]	0	0
[7 5 7 7 5]	3	0,000003	[7 7 7 5 5]	0	0
[7 7 5 5 7]	3	0,000003	[5 5 7 7 7]	1	0,000001
[5 7 5 7 7]	4	0,000004	[7 5 5 7 7]	1	0,000001
[7 7 5 7 5]	4	0,000004	[7 5 7 5 7]	1	0,000001
[7 7 7 5 5]	4	0,000004	[5 7 7 5 7]	2	0,000002
[5 7 7 5 7]	5	0,000005	[5 7 7 7 5]	2	0,000002
[5 5 7 7 7]	9	0,000009	[7 7 5 5 7]	2	0,000002

Табл. 3. Резултати от симулационното изследване на кодери с дължина на кодовото ограничение $L = 5$ и с брой на генераторните полиноми $GP = 2$

Брой предадени битове: 1000000							
Твърдо декодиране				Меко декодиране			
Кодер	df max	Брой гр. битове	BER	Кодер	df max	Брой гр. битове	BER
[27 25]	7	4654	0,00465	[27 25]	7	2006	0,00201
[25 27]	7	4735	0,00474	[27 19]	7	2012	0,00201
[29 25]	7	4758	0,00476	[23 25]	7	2050	0,00205
[27 19]	7	4773	0,00477	[25 27]	7	2054	0,00205
[19 27]	7	4877	0,00488	[25 23]	7	2152	0,00215
[19 23]	7	4927	0,00493	[19 27]	7	2181	0,00218
[25 29]	7	4932	0,00493	[23 19]	7	2225	0,00223
[23 19]	7	4988	0,00499	[19 29]	7	2229	0,00223
[19 29]	7	5068	0,00507	[25 29]	7	2285	0,00229
[29 19]	7	5136	0,00514	[29 25]	7	2305	0,00231
[25 23]	7	5137	0,00514	[29 19]	7	2319	0,00232
[23 25]	7	5190	0,00519	[19 23]	7	2333	0,00233
[23 29]	8	499541	0,49956	[23 29]	8	499425	0,49944
[29 23]	8	499760	0,49978	[21 27]	7	499841	0,49986
[21 27]	7	499896	0,49991	[29 23]	8	500352	0,50037
[27 21]	7	499951	0,49997	[27 21]	7	500389	0,50041

Конволюционните кодери [23 29], [29 23], [21 27], [27 21] (табл. 3) допускат твърде много сгрешени битове – близо половината от предадените битове, при това кодерите [23 29] и [29 23] са единствените с максимално-възможната стойност на параметъра свободно разстояние за разглеждания случай. Именно това обоснова необходимостта от разширяването на "базата от данни" на най-добрите кодери и с тези, които имат по-малка (с единица) стойност на параметъра свободно разстояние.

Генераторни полиноми на кодерите [23 29], [29 23], [21 27] и [27 21] притежават общ полиномиален множител, а съгласно Меси и Сейн [2], наличието на общ полиномиален множител довежда до разпространението на катастрофални грешки. Например, за кодерите с генератори 23 и 29 – този полиномиален множител е $1 \oplus X$.

В табл. 4 са дадени десетичното и двоичното представяне на генераторните полиноми 21, 23, 27 и 29 и съответно полиномите чрез фиктивната променлива X, и тяхното разложение на множители.

Табл. 4. Полиномно представяне на конволюционните кодери и разложение на полиномите на множители, за кодери с дължина на кодовото ограничение $L = 5$

№	Дв.код	Полином	Разложение на полиномите на множители
21	10101	$1 \oplus X^2 \oplus X^4$	$(1 \oplus X^2).(1 \oplus X^2) =$ $= (1 \oplus X).(1 \oplus X).(1 \oplus X).(1 \oplus X)$
23	10111	$1 \oplus X^2 \oplus X^3 \oplus X^4$	$1 \oplus X^2 \oplus X^3.(1 \oplus X) =$ $= (1 \oplus X).(1 \oplus X) \oplus X^3.(1 \oplus X) =$ $= (1 \oplus X \oplus X^3).(1 \oplus X)$
27	11011	$1 \oplus X \oplus X^3 \oplus X^4$	$1 \oplus X \oplus X^3.(1 \oplus X) = (1 \oplus X).(1 \oplus X^3) =$ $= (1 \oplus X).(1 \oplus X).(1 \oplus X \oplus X^2)$
29	11101	$1 \oplus X \oplus X^2 \oplus X^4$	$1 \oplus X \oplus X^2.(1 \oplus X^2) =$ $= 1 \oplus X \oplus X^2.(1 \oplus X).(1 \oplus X) =$ $= (1 \oplus X).(1 \oplus X^2 \oplus X^3)$

Доказва се, че при конкретните условия на симулационния експеримент, при $L = 5$ и $GP = 2$, "най-добър" кодер е [27 25], т.е. получава се противоречие с литературните източници [1, 2], според които "най-добър кодер" е [19 29]. Кодерът [27 25] притежава стойност на параметъра свободно разстояние, равна на 7, т.е. с единица по-малка от максималната възможна стойност на кодери с дължина на кодовото ограничение $L = 5$ и с брой на генераторните полиноми $GP = 2$.

ЗАКЛЮЧЕНИЕ

Разработената система дава възможност да се изследват конволюционни кодери и декодери в условията на шум в цифровите комуникационни системи. За разлика от известните литературни източници [1, 2], в резултат на проведените експерименти бе установено, че при конкретни условия на симулационния експеримент, "най-добър" кодер може да бъде и кодер, който има стойност на параметъра свободно разстояние с единица по-малка от максимално-възможната. Това дава възможност за разширяване на съществуващите "бази от данни" на най-добрите кодери [1, 2]. От направения в доклада анализ (табл. 2 и табл. 3) е установено, че върху коригиращите свойства на кодерите влияе не само дължината на кодовото ограничение, но и броят на генераторните полиноми.

Създадената софтуерна система може да намери практическо приложение и в учебния процес по дисциплината "Кодиране в телекомуникационните системи" за специалност „Комуникационна техника и технологии".

ЛИТЕРАТУРА

[1] Блейхут, Р., Теория и практика кодов, контролирующих ошибки. Перевод с английского И.И. Грушко и В.М. Блиновского. Москва, Мир, 1986.

[2] Скляр Б. Цифровая связь. Теоретические основы и практическое применение. Москва, Вильямс, 2003.

[3] Borodzhieva, A., P. Manoilov. Simulating the Operation of Feed-forward Convolutional Encoders and Viterbi Decoders Applied in Additive White Gaussian Noise Communication Channels. International Conference CompSysTech'08, Gabrovo, 12 – 13 June 2008, in the press.

[4] Borodzhieva, A., P. Manoilov. Software Tool for Simulating Convolutional Encoding and Decoding Used in Communication Systems. Fourth International Bulgarian-Greek Conference – Computer Science'2008, 18 – 19 September 2008, Kavala, Greece, in the press.

[5] Borodzhieva, A. Software Instruments for Investigating Convolutional Encoding and Decoding Processes Applied in Communication Systems. International Symposium for Design and Technology of Electronic Packaging, 14th Edition, SIITME 2008, September 18 – 21, Predeal, Romania, Conference proceedings, pp. 135 – 139, ISSN 1843-5122.

[6] Communications Toolbox 3. User's Guide. http://www.mathworks.com/access/helpdesk/help/pdf_doc/comm/comm.pdf.

[7] http://en.wikipedia.org.

За контакти:

Гл. ас. инж. Адриана Найденова Бороджиева, Катедра "Комуникационна техника и технологии", Русенски университет "Ангел Кънчев", Тел.: 082 888 734, E-mail: aborodjieva@ecs.ru.acad.bg.

Докладът е рецензиран.