Approximation of DAC Codeword Distribution for Equiprobable Binary Sources Along Proper Decoding Paths

Nan Yang and Yong Fang(✉)

School of Information Engineering, Chang'an University, Xi'an, Shaanxi, China
{nyang,fy}@chd.edu.cn

Abstract. Distributed Arithmetic Coding (DAC) is an effective implementation of Slepian-Wolf coding. To research its properties, the concept of DAC codeword distribution along proper decoding paths has been introduced. For DAC codeword distribution of equiprobable binary sources along proper decoding paths, the problem was formatted as solving a system of functional equations. However, in general cases, to find the closed form of DAC codeword distribution still remains a very difficult task. This paper proposes an approximation method for DAC codeword distribution of equiprobable binary sources along proper decoding paths: polynomial approximation. At rates lower than 0.5, DAC codeword distribution can be well approximated by a polynomial. Some simulation results are given to verify theoretical analyses.

Keywords: Slepian-Wolf coding · Distributed arithmetic coding · Codeword distribution · Polynomial approximation

1 Introduction

1.1 Background

Consider the problem of Slepian-Wolf Coding (SWC) with decoder Side Information (SI), i.e. the encoder compresses discrete source X in the absence of Y, discretely-correlated SI. Slepian-Wolf theorem states that lossless compression is achievable at rates $R \geq H(X|Y)$ [1], where $H(X|Y)$ is the conditional entropy of X given Y. Conventionally, channel codes, e.g., turbo codes [2] or Low-Density Parity-Check (LDPC) codes [3], are used to implement the SWC.

Ever since a long time ago, Arithmetic Coding (AC) has been proposed as the successor of Huffman coding to implement source coding and shows near-entropy performance [4–6]. Recently, the AC is applied to implement the SWC. One approach is to allow overlapped intervals, which mirrors the work in [7]. Such examples include Distributed Arithmetic Coding (DAC) [8,9] and Overlapped Quasi-Arithmetic Coding (OQAC) [10]. Another approach is to puncture some bits of AC bitstream, e.g. Punctured Quasi-Arithmetic Coding (PQAC) [11],

B. Li et al. (Eds.): IoTaaS 2020, LNICST 346, pp. 3–13, 2021.
https://doi.org/10.1007/978-3-030-67514-1_1

which mirrors the work in [12]. There are also some variants of the DAC. The symmetric SWC is implemented by the time-shared DAC (TS-DAC) [13]. The rate-compatible DAC is proposed in [14]. Furthermore, decoder-driven adaptive DAC [15] is proposed to estimate source probabilities on-the-fly.

To analyze the properties of the DAC, [16] introduces the concept of codeword distribution. DAC codeword distribution is a function defined over interval $[0, 1)$. For equiprobable binary sources, both codeword distribution along proper decoding paths and codeword distribution along wrong decoding paths are researched. For codeword distribution along proper decoding paths, the problem is formatted as solving a system of functional equations including four constraints [16]. It is affirmed that rate $R = 0.5$ is a watershed: when $R > 0.5$, DAC codeword distribution is an unsmooth function; while when $R \leq 0.5$, DAC codeword distribution is a smooth function. Especially, a closed form is obtained at $R = 0.5$. In spite of these achievements, it remains a very difficult task to find the closed form of codeword distribution along proper decoding paths in general. Though a special closed form of $f(u)$ is found in [16], the procedure is very complex. As a universal approach, a numeric method for finding $f(u)$ was proposed in [17].

1.2 Contribution

This paper makes some advances on the work in [16]. An approximation method is proposed for codeword distribution of equiprobable binary sources along proper decoding paths: polynomial approximation. Then comparisons of polynomial approximation with numeric approximation, among them, numeric approximation is a general approach. Proved that at low rates ($R \leq 0.5$), polynomial approximation works well.

This paper is arranged as follows. In Sect. 2, after a brief introduction to binary DAC codec, DAC decoding process is analyzed in detail to show the significance of DAC codeword distribution. Then the investigated problem is formulated in Sect. 3. Section 4 describes in detail polynomial approximation, where simulation results are also reported. Finally, Sect. 5 concludes this paper.

2 Binary Distributed Arithmetic Coding

2.1 Encoding

Consider a binary source $X = \{x_i\}_{i=1}^{I}$ with bias probability $p = \Pr(x_i = 1)$. In the classic AC, source symbol x_i is iteratively mapped onto sub-intervals of $[0, 1)$, whose lengths are proportional to $(1-p)$ and p, giving rate $R = H(X)$. Instead, in the DAC [8,9], sub-interval lengths are proportional to enlarged probabilities $(1-p)^\gamma$ and p^γ, where $H(X|Y)/H(X) \leq \gamma \leq 1$, giving rate $R = \gamma H(X) \geq H(X|Y)$. For conciseness, we refer to γ as overlap coefficient hereinafter. More specifically, symbols $x_i = 0$ and $x_i = 1$ correspond to sub-intervals $[0, (1-p)^\gamma)$ and $[1-p^\gamma, 1)$, respectively. It means that to fit the $[0, 1)$ interval, the sub-intervals have to be partially overlapped. This overlapping leads to a larger final

interval, and hence a shorter codeword. However, as a cost, the decoder can not decode X unambiguously without Y.

Note that when $\gamma \geq 1/C \geq 1$, where C is channel capacity, it becomes the Error Correcting AC (ECAC).

2.2 Decoding

To describe the decoding process, a ternary symbol set $\{0, \mathcal{A}, 1\}$ is defined, where $\mathcal{A}$ represents the ambiguous symbol. Let C_X be DAC codeword and $\tilde{x}_i$ be the i-th decoded symbol, then

$$\tilde{x}_i = \begin{cases} 0, & 0 \leq C_X < 1 - p^\gamma \\ \mathcal{A}, & 1 - p^\gamma \leq C_X < (1-p)^\gamma \\ 1, & (1-p)^\gamma \leq C_X < 1 \end{cases}. \tag{1}$$

After $\tilde{x}_i$ is decoded, if $\tilde{x}_i = \mathcal{A}$, the decoder will perform a branching: two candidate branches are generated, corresponding to two alternative symbols $x_i = 0$ and $x_i = 1$. For each new branch, its metric is updated and the corresponding interval is selected for next iteration. To reduce complexity, every time a symbol is decoded, the decoder uses the M-algorithm to keep at most M paths with the best partial metric, and prunes others [8,9]. Finally, after all source symbols are decoded, the path with the best metric is output as the estimate of X.

2.3 Analysis

It deserves to point out that during DAC decoding, the metric of each path is indeed the Hamming distance between this path and SI Y. As we know, each DAC codeword defines a set of possible decoding paths and each possible decoding path corresponds to a sequence of decoded symbols. However, among all possible decoding paths, there is one and only one proper path which corresponds to source X. Let $\tilde{X} = \{\tilde{x}_i\}_{i=1}^{I}$ be a sequence of decoded symbols. Let $D(Y, \tilde{X})$ be the Hamming distance between Y and $\tilde{X}$. Similarly, $D(X, \tilde{X})$ and $D(X, Y)$ are also defined. Obviously,

$$D(Y, \tilde{X}) \leq D(X, Y) + D(X, \tilde{X}). \tag{2}$$

The task of a DAC decoder is in fact to find a path X' that minimizes $D(Y, \tilde{X})$, i.e.

$$X' = \arg\min_{\tilde{X}} D(Y, \tilde{X}). \tag{3}$$

However, this is not always followed by $D(X, X') = 0$. If $D(X, X') \neq 0$, then a decoding failure occurs. To find the probability of decoding failure, we need to know the distribution of $D(Y, \tilde{X})$ and $D(X, \tilde{X})$.

Though it is very difficult to find the distribution of $D(Y, \tilde{X})$ and $D(X, \tilde{X})$, this problem can be tackled by means of DAC codeword distribution. As shown in [16], if we know codeword distributions along proper and wrong decoding

paths, it seems promising to find the number of possible decoding paths and the distribution of $D(Y, \tilde{X})$ and $D(X, \tilde{X})$.

The rest of this paper makes some advances on DAC codeword distribution along proper decoding paths.

3 Problem Formulation

To simplify the analysis, we consider an infinite-length, stationary, and equiprobable binary source $X = \{x_i\}_{i=1}^{\infty}$. As $p = 0.5$, symbols $x_i = 0$ and $x_i = 1$ correspond to sub-intervals $[0, q)$ and $[1 - q, 1)$ respectively, where $q = 0.5^{\gamma}$. The resulting rate $R = \gamma H(X) = \gamma$.

Let C_X be the DAC codeword of X and $f(u)$ $(0 \leq u < 1)$ be the distribution of C_X, then

$$\int_0^1 f(u)du = 1. \tag{4}$$

Due to the symmetry, we have

$$f(u) = f(1 - u), \quad 0 < u < 1. \tag{5}$$

Symbols $x_1 = 0$ and $x_1 = 1$ correspond to intervals $[0, q)$ and $[1 - q, 1)$, respectively. If $x_1 = 0$, the remaining sequence $X_2 = \{x_i\}_{i=2}^{\infty}$ will be iteratively mapped onto the sub-intervals of $[0, q)$; If $x_1 = 1$, X_2 will be iteratively mapped onto the sub-intervals of $[1 - q, 1)$. Let $C_{X_2}^0$ be the DAC codeword of X_2 given $x_1 = 0$ and $f_0(u)$ be the distribution of $C_{X_2}^0$, then

$$\int_0^q f_0(u)du = 1. \tag{6}$$

Since X is infinite-length and stationary, $f_0(u)$ must have the same shape as $f(u)$, i.e.,

$$f_0(u) = f(u/q)/q, \quad 0 \leq u < q. \tag{7}$$

Similarly, let $C_{X_2}^1$ be the DAC codeword of X_2 given $x_1 = 1$ and $f_1(u)$ be the distribution of $C_{X_2}^1$, then

$$f_1(u) = f_0(u - (1 - q)) = f(\frac{u - (1 - q)}{q})/q, \quad (1 - q) \leq u < 1. \tag{8}$$

Due to the symmetry,

$$f_1(u) = f_0(1 - u). \tag{9}$$

The relations between $f(u)$, $f_0(u)$ and $f_1(u)$ can be illustrated by Fig. 1. Obviously,

$$f(u) = \Pr(x_1 = 0)f_0(u) + \Pr(x_1 = 1)f_1(u) = (f_0(u) + f_1(u))/2. \tag{10}$$

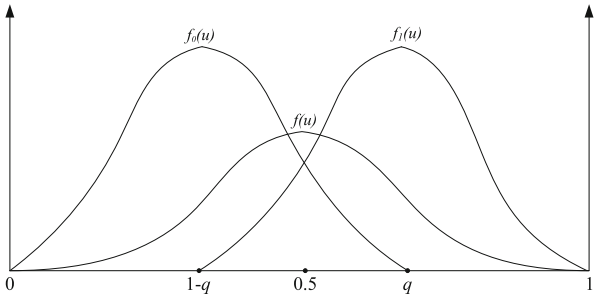

Fig. 1. Illustrations of the relations between $f(u), f_0(u)$ *and* $f_1(u)$

In [16], we know that $(\sqrt{5}-1)/2$ and $1/\sqrt{2}$ are two watersheds in the interval $(0.5, 1)$ of q. Below the properties of $f(u)$ in different sub-intervals will be analyzed.

A. When $q = 0.5$, it is the classic AC. Then

$$f(u) = \begin{cases} f(2u), & 0 \le u < 0.5 \\ f(2u-1), & 0.5 \le u < 1 \end{cases}. \tag{11}$$

This is a uniform distribution, so the classic AC can achieve source entropy theoretically.

B. When $0.5 < q < 1$, $f(u)$ is a piecewise-defined function, sub-intervals $[0, q)$ and $[1-q, 1)$ are partially overlapped.

(1) $0 \le u < (1-q)$: In this interval, $f_1(u) = 0$, so

$$f(u) = f_0(u)/2 = f(u/q)/(2q). \tag{12}$$

(2) $q \le u < 1$: In this interval, $f_0(u) = 0$, so

$$f(u) = f_1(u)/2 = f(\frac{u-(1-q)}{q})/(2q). \tag{13}$$

(3) $1-q \le u < q$: In this interval, we have

$$f(u) = \frac{f(\frac{u}{q}) + f(\frac{u-(1-q)}{q})}{2q}. \tag{14}$$

C. when $q = 1/\sqrt{2}$, it is the closed form of $f(u)$, In [16], only one closed form is obtained at $q = 1/\sqrt{2}$ (i.e. $\gamma = 0.5$)

In addition, it is proved in [16] that when $0.5 < q \le \frac{\sqrt{5}-1}{2}$ (corresponds to $0.6942 \le \gamma < 1$), $f(\frac{q^n}{q+1}) = f(1-\frac{q^n}{q+1}) = 0$, $\forall n \in \mathbb{N}$.

4 Polynomial Approximation at Low Rates

Though a special closed form of $f(u)$ is found for $p = \gamma = 0.5$ in [16], the procedure is very complex. As a universal approach for solving this problem, [17] proposes a numeric method for finding $f(u)$.

However, it was affirmed in [16] that $f(u)$ is a smooth function when $q \geq 1/\sqrt{2}$, i.e. $R \leq 0.5$.This property suggests that polynomials may be good approximation to $f(u)$ at low rates ($R \leq 0.5$).Below we propose polynomial approximation to $f(u)$ for $1/\sqrt{2} \leq q < 1$.

To simplify the analysis, we exploit the symmetry and consider only the left half of $f(u)$

$$f(u) = \begin{cases} f(u/q)/(2q), & 0 \leq u \leq (1-q) \\ \frac{f(\frac{u}{q}) + f(\frac{u-(1-q)}{q})}{2q}, & (1-q) \leq u \leq 0.5 \end{cases}. \tag{15}$$

We rewrite (15) as

$$f(u) = \begin{cases} 2qf(qu), & 0 \leq u \leq v_1 \\ 2qf(qu) - f(u - v_1), & v_1 \leq u \leq 0.5 \end{cases}. \tag{16}$$

where $v_n = (1-q)/q^n$, $n \in \mathbb{N}$. Note that $v_1 < 0.5$ when $q \geq 1/\sqrt{2}$. Hence, $f(u)$ is a piecewise-defined function over interval $[0, 0.5]$.

At first, in sub-interval $[0, v_1]$, $f(u)$ can be obtained by solving functional equation $f(u) = 2qf(qu)$ [18]

$$f(u) = \phi(u) = \Theta(u)u^{\lambda}, \quad 0 \leq u \leq v_1, \tag{17}$$

where $\lambda = (1-\gamma)/\gamma$ and $\Theta(u) = \Theta(uq^k)$, $\forall k \in \mathbb{Z}$.

Then, we need to determine $f(u)$ in sub-interval $[v_1, 0.5]$. Because $qu < u$ and $u - v_1 < u$, it is possible to recursively map sub-interval $[v_1, 0.5]$ onto sub-interval $[0, v_1]$ by scaling down or shifting u, over which $f(u)$ has been given by (17), i.e.

$$\begin{cases} f(qu) = \phi(qu), & v_1 \leq u \leq v_2 \\ f(u - v_1) = \phi(u - v_1), & v_1 \leq u \leq 2v_1 \end{cases}. \tag{18}$$

This is the key to solving this problem.

Due to $(u - v_1) - qu = (1-q)(u - 1/q) < 0$, i.e. $u - v_1 < qu$ for $u \in [v_1, 0.5]$. Hence,

$$f(u) = 2qf(qu) - \phi(u - v_1), \quad v_1 \leq u \leq 2v_1. \tag{19}$$

On solving $2v_1 = 0.5$, we obtain $q = 0.8$. Hereinafter, to facilitate our description, we divide interval $1/\sqrt{2} \leq q < 1$ into two sub-intervals $1/\sqrt{2} \leq q \leq 0.8$ (corresponding to $0.5 \leq 2v_1$) and $0.8 < q < 1$ (corresponding to $2v_1 < 0.5$) (Fig. 2).

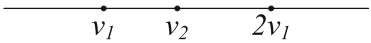

Fig. 2. Illustrations of v_1, v_2 *and* $2v_1$ in the interval $[0, 1]$

4.1 $1/\sqrt{2} \leq q \leq 0.8$

In this sub-interval, since $0.5 \leq 2v_1$, we have

$$f(u) = \begin{cases} \phi(u), & 0 \leq u \leq v_1 \\ 2qf(qu) - \phi(u - v_1), & v_1 \leq u \leq 0.5 \end{cases}. \tag{20}$$

Hence, we need to consider only the term $2qf(qu)$. Depending on the relations between v_n and 0.5, this sub-interval can be further divided into three smaller sub-intervals.

(1) $0.5 \leq v_2$: On solving $v_2 = 0.5$, we obtain $q = \sqrt{3} - 1$, so this sub-interval corresponds to $1/\sqrt{2} \leq q \leq \sqrt{3} - 1$. Since $0.5 \leq v_2$, we have $qu \leq v_1$ for $u \in [v_1, 0.5]$, i.e. $f(qu) = \phi(qu)$. Remember $\phi(u) \equiv 2q\phi(qu)$. Thus

$$f(u) = \begin{cases} \phi(u), & 0 \leq u \leq v_1 \\ \phi(u) - \phi(u - v_1), & v_1 \leq u \leq 0.5 \end{cases}. \tag{21}$$

As affirmed in [16], $f(u)$ is a smooth function for $q \geq 1/\sqrt{2}$. Hence we approximate $\Theta(u)$ by a const c and then obtain

$$f(u) \approx \begin{cases} cu^{\lambda}, & 0 \leq u \leq v_1 \\ cu^{\lambda} - c(u - v_1)^{\lambda}, & v_1 \leq u \leq 0.5 \end{cases}. \tag{22}$$

Now we need to determine c. Let us integrate $f(u)$ over interval $[0, 0.5]$

$$\begin{aligned} \int_0^{0.5} f(u)du &= c\left(\int_0^{0.5} u^{\lambda}du - \int_{v_1}^{0.5} (u - v_1)^{\lambda}du\right) \\ &= \frac{c(u^{\lambda+1}|_0^{0.5} - (u - v_1)^{\lambda+1}|_{v_1}^{0.5})}{\lambda + 1} \\ &= \frac{c(0.5^{\lambda+1} - (0.5 - v_1)^{\lambda+1})}{\lambda + 1} = 0.5. \end{aligned} \tag{23}$$

Thus,

$$c = \frac{0.5(\lambda + 1)}{0.5^{\lambda+1} - (0.5 - v_1)^{\lambda+1}}. \tag{24}$$

Due to $\lambda + 1 = 1/\gamma$,

$$c = \frac{1}{2\gamma(0.5^{(1/\gamma)} - (0.5 - v_1)^{(1/\gamma)})}. \tag{25}$$

(2) $v_2 < 0.5 \leq v_3$: On solving $v_3 = 0.5$, we obtain $q \approx 0.77$, so this sub-interval corresponds to $\sqrt{3} - 1 < q \leq 0.77$. At first, it can be obtained directly

$$f(u) = \begin{cases} \phi(u), & 0 \leq u \leq v_1 \\ \phi(u) - \phi(u - v_1), & v_1 \leq u \leq v_2 \end{cases}. \tag{26}$$

Then, for $u \in [v_2, 0.5]$, we have $qu \in [v_1, v_2]$, i.e. $f(qu) = \phi(qu) - \phi(qu - v_1)$. Thus

$$\begin{aligned} f(u) &= 2qf(qu) - \phi(u - v_1) \\ &= 2q(\phi(qu) - \phi(qu - v_1)) - \phi(u - v_1), \quad v_2 \le u \le 0.5. \end{aligned} \tag{27}$$

Because $2q\phi(qu - v_1) = 2q\phi(q(u - v_2)) = \phi(u - v_2)$, we obtain

$$f(u) = \phi(u) - \sum_{i=1}^{2} \phi(u - v_i), \quad v_2 \le u \le 0.5. \tag{28}$$

Therefore, we can obtain the following approximation

$$f(u) \approx \begin{cases} cu^{\lambda}, & 0 \le u \le v_1 \\ cu^{\lambda} - c(u - v_1)^{\lambda}, & v_1 \le u \le v_2 \\ cu^{\lambda} - c\sum_{i=1}^{2} (u - v_i)^{\lambda}, & v_2 \le u \le 0.5 \end{cases}, \tag{29}$$

where

$$c = \frac{1}{2\gamma(0.5^{(1/\gamma)} - \sum_{i=1}^{2} (0.5 - v_i)^{(1/\gamma)})}. \tag{30}$$

(3) $v_3 < 0.5 \le v_4$: On solving $v_4 = 0.5$, we obtain $q \approx 0.8$, so this sub-interval corresponds to $0.77 < q \le 0.8$. By iterations, we can obtain

$$f(u) \approx \begin{cases} cu^{\lambda}, & 0 \le u \le v_1 \\ cu^{\lambda} - c(u - v_1)^{\lambda}, & v_1 \le u \le v_2 \\ cu^{\lambda} - c\sum_{i=1}^{2} (u - v_i)^{\lambda}, & v_2 \le u \le v_3 \\ cu^{\lambda} - c\sum_{i=1}^{3} (u - v_i)^{\lambda}, & v_3 \le u \le 0.5 \end{cases}, \tag{31}$$

where

$$c = \frac{1}{2\gamma(0.5^{(1/\gamma)} - \sum_{i=1}^{3} (0.5 - v_i)^{(1/\gamma)})}. \tag{32}$$

4.2 $0.8 < q < 1$

The problem becomes very complex in this sub-interval because $f(u - v_1) = \phi(u - v_1)$ does not hold for $u \in [2v_1, 0.5]$ so that we need to deal with not only $2qf(qu)$ but also $f(u - v_1)$.

Let us consider a simple case first, i.e. $v_1 < 0.5 - v_1 \le v_2$, which corresponds to sub-interval $0.8 < q \le \sqrt{2/3}$. We have $u - v_1 \in [v_1, v_2]$ for $u \in [2v_1, 0.5]$. Hence

$$f(u - v_1) = \phi(u - v_1) - \phi(u - 2v_1), \quad 2v_1 \le u \le 0.5. \tag{33}$$

Therefore, the problem becomes

$$f(u) = \begin{cases} 2qf(qu), & 0 \le u \le v_1 \\ 2qf(qu) - \phi(u - v_1), & v_1 \le u \le 2v_1 \\ 2qf(qu) - (\phi(u - v_1) - \phi(u - 2v_1)), & 2v_1 \le u \le 0.5 \end{cases}. \tag{34}$$

Now we need to deal with only $2qf(qu)$, which has been discussed in detail in Sect. 4.1.

For $\sqrt{2/3} < q < 1$, the idea is the same but the procedure becomes more and more complicated as q increases. Therefore, at very low rates, polynomial approximation is not a good choice.

4.3 Simulation Results

Some examples of polynomial approximation have been included in Fig. 3. In the simulation, set numerical approximation parameters $N = 10^5$ and $\delta = 10^{-10}$. Considering the complexity, only the results for $1/\sqrt{2} \leq q \leq 0.8$ are reported. Figure 3 shows that in general, the curves of polynomial approximation fit those of numeric approximation very well. Especially, as q increases, the curves of polynomial approximation almost coincide with those of numeric approximation.

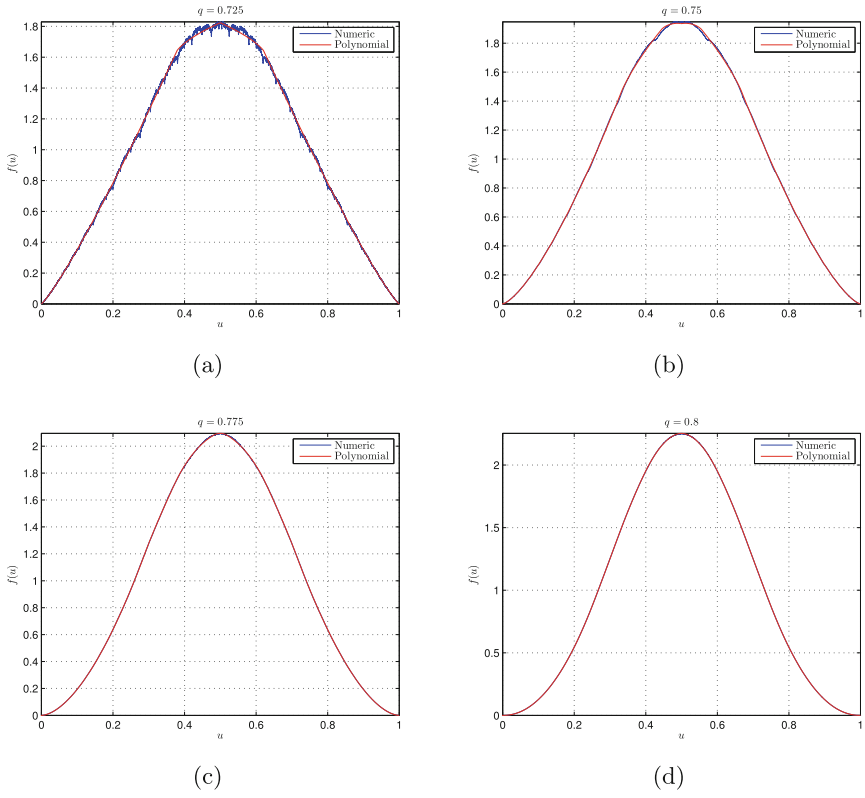

Fig. 3. Comparisons of polynomial approximation with numeric approximation. These results show that polynomial approximation fits numeric approximation very well. Especially, as q increases, polynomial approximation almost coincides with numeric approximation. (a) $q = 0.725$. (b) $q = 0.75$. (c) $q = 0.775$. (d) $q = 0.8$.

In addition, Fig. 3(a) also shows the affirmation in [16] may fail because $q > 1/\sqrt{2}$ does not guarantee smooth $f(u)$. Nevertheless, $f(u)$ does become less irregular as q increases.

5 Conclusion

This paper proposes a polynomial approximation method for DAC codeword distribution of equiprobable binary sources along proper decoding paths. The polynomial approximation fits those of numeric approximation very well.

However, when $0.8 < q < 1$, the polynomial approximation becomes very complicated as q increases. So, the future research direction is to find a simpler approximation method in this interval.

References

1. Slepian, D., Wolf, J.K.: Noiseless coding of correlated information sources. IEEE Trans. Inf. Theory **19**(4), 471–480 (1973)
2. Garcia-Frias, J., Zhao, Y.: Compression of correlated binary sources using turbo codes. IEEE Commun. Lett. **5**(10), 417–419 (2001)
3. Liveris, A., Xiong, Z., Georghiades, C.: Compression of binary sources with side information at the decoder using LDPC codes. IEEE Commun. Lett. **6**(10), 440–442 (2002)
4. Rissanen, J.: Generalized Kraft inequality and arithmetic coding. IBM J. Rese. Dev. **20**(3), 198–203 (1976)
5. Rissanen, J., Langdon, G.: Arithmetic coding. IBM J. Res. Dev. **23**(2), 149–162 (1979)
6. Rissanen, J.J.: Arithmetic codings as number representations. Acta Polytech. Scand. **31**, 44–51 (1979)
7. Boyd, C., Cleary, J.G., Irvine, S.A., Rinsma-Melchert, I., Witten, I.H.: Integrating error detection into arithmetic coding. IEEE Trans. Commun. **45**(1), 1–3 (1997)
8. Grangetto, M., Magli, E., Olmo, G.: Distributed arithmetic coding. IEEE Commun. Lett. **11**(11), 883–885 (2007)
9. Grangetto, M., Magli, E., Olmo, G.: Distributed arithmetic coding for the Slepian-Wolf problem. IEEE Trans. Signal Process. **57**(6), 2245–2257 (2009)
10. Artigas, X., Malinowski, S., Guillemot, C., Torres, L.: Overlapped quasi-arithmetic codes for distributed video coding. In: Proceedings of the IEEE ICIP, vol. 2, no. 2, pp. 9–12 (2007)
11. Malinowski, S., Artigas, X., Guillemot, C., Torres, L.: Distributed coding using punctured quasi-arithmetic codes for memory and memoryless sources. In: Proceedings of the IEEE PCS, Chicago, IL (2009)
12. Sodagar, I., Chai, B.B., Wus, J.: A new error resilience technique for image compression using arithmetic coding. In: Proceedings of the IEEE ICASSP, pp. 2127–2130 (2000)
13. Grangetto, M., Magli, E., Olmo, G.: Symmetric distributed arithmetic coding of correlated sources. In: Proceedings of the IEEE MMSP, pp. 111–114 (2007)
14. Grangetto, M., Magli, E., Tron, R., Olmo, G.: Rate-compatible distributed arithmetic coding. IEEE Commun. Lett. **12**(8), 575–577 (2008)

15. Grangetto, M., Magli, E., Olmo, G.: Decoder-driven adaptive distributed arithmetic coding. In: Proceedings of the IEEE ICIP, pp. 1128–1131 (2008)
16. Fang, Y.: Distribution of distributed arithmetic codewords for equiprobable binary sources. IEEE Signal Process. Lett. **16**(12), 1079–1082 (2009)
17. Fang, Y.: DAC spectrum of binary sources with equally-likely symbols. IEEE Trans. Commun. **61**(4), 1584–1594 (2013)
18. http://eqworld.ipmnet.ru/en/solutions/fe/fe1111.pdf