算術符号の設計要素

小野 文孝[†]

†東京工芸大学

E-mail: ono@image.t-kougei.ac.jp

あらまし 算術符号化はその対象系列ごとに独立で符号を決定できるという特性から，マルコフモデル符号化の実用的かつ効率的符号化手段として脚光を浴び，まず 2 値画像符号化標準に適用された．その後，適用範囲を拡大し多値画像符号化標準，動画符号化標準にも採用されている．これは算術符号がコンテクスト認知により汎用的なモデルに適用できるという効果に基づくものといえる．本報告では算術符号化の実用化・高速化の歴史を振り返るとともに，算術符号を構成する様々な設計要素について解説する．

キーワード 算術符号，静止画符号化， ハフマン符号，符号化標準

Design Factors of the Arithmetic Coder

Fumitaka ONO[†]

† Professor Emeritus, Tokyo Polytechnic University

E-mail: ono@image.t-kougei.ac.jp

Abstract In the arithmetic coding it is possible to allocate the codeword to the occurred symbol sequence independent to other possible sequences. Such feature makes it possible to encode Markov model source in practical and efficient way. Therefore arithmetic coding was firstly adopted in the Markov-model based bi-level image coding standard, and then it has been adopted in still image coding standards and also in video coding standards. The reason of such expansion is that the arithmetic coding can realize the Markov model entropy rate through the automatically recognizing the context. In this paper, the author reviews the history of the arithmetic coding focusing on the effort to speed up the execution process of arithmetic coding. The various design factors of the arithmetic coding are also reviewed.

Keyword arithmetic coder, still image coding, Huffman coding, coding standard

1. まえがき

算術符号の最初の実用的提案[1)]は 2 値画像の符号化を目的とするものであった．その結果，算術符号の持つブロック符号（非算術符号）では成し得ないマルコフモデル情報源（マルチコンテクスト情報源）の情報源拡大符号化効果が評価され，マルコフモデルに基づく 2 値画像符号化の最初の標準である JBIG-1[2)]におけるエントロピー符号手段としてまず採用された．

次に多値静止画標準への適用であるが，最初の多値静止画符号化標準である JPEG[3)]では DCT（Discrete Cosine Transform）係数のランレングス符号化が採用されたためハフマン符号で充分であったが，プログレッシブ符号化やロスレス符号化などへの適用も考慮して算術符号がオプション符号として採用された．続く，JPEG-LS[4,5)]では多値マルコフモデルに基づく符号化となったため算術符号の採用が検討されたが最終的にはエントロピー符号／復号器の複雑性を考慮して必須機能である Part1[4)]ではモード分離を行い自然画モードでは情報源拡大を行わないブロック符号が採用され，情報源拡大が必要な CG・文書画像モードではランレングス符号化を採用し算術符号の採用は見送られた．オプションである Part2[5)]では算術符号が採用され，その結果 Part2 ではモード分離処理を不要とできた．続く標準である JPEG2000[6)]では DWT（Discrete Wavelet Transform）係数をビットプレーン分解して符号化するモデルが採用され，ついに算術符号が唯一のエントロピー符号化方式に位置付けられた．

一方動画では H.264[7)]で初めて算術符号（CABAC）が採用されたがブロック符号である CAVLC と共存する形であった．そして最新の H.265[8)]では CABAC が唯一のエントロピー符号化となっている．

つまり算術符号は当初はマルチコンテクスト情報源の効率的符号化手段という機能が評価されて導入され，最近では符号化の絶対的な高効率性が評価されると同時に情報源のコンテクストの自動認知効果により達成目標エントロピーを低減化する効果も評価される

ようになったといえる．そのような観点から，算術符号の位置づけは重要性を増している [9]．本文では算術符号の実用化の歴史を振り返るとともに算術符号を構成する様々な設計要素について解説する．

2. 算術符号の原理

2.1 数直線符号化

算術符号は非ブロック符号の代表といえる．今，任意の符号データの先頭に小数点をつけると，0から1の数直線上のある領域を表すことになり，符号系列がnビットの符号語には数直線上で $1/2^n$ の領域が割り当てられていると解釈できる．これを数直線マッピングという．

さて，情報理論では，確率pの事象に $\log_2(1/p)$ ビットを割り当てるのが理想符号化である．逆にnビットの符号語を割り当てられた事象については，その確率が $1/2^n$ である場合が理想となる．数直線マッピングでは，nビットの符号語には数直線上で $1/2^n$ の領域が割り当てられるので，「シンボルに,その出現確率に基づいて数直線上の領域を割り当てる」処理を，数直線符号化と定義すると，数直線符号化においては，「確率pの事象の数直線上での理想的領域は $(1/2)^{**} \log_2(1/p)$ （=p）である」という極めてシンプルな原理が導かれる [10]．

したがって数直線利用符号化ではその対応領域をシンボルの出現確率に応じて分割してゆき，最終的にその領域に存在することがわかるような符号語を出力すればよいことになる．

2.2 算術符号化の定義

さて，数直線符号化において途中段階でのシンボル復号を可能とするためには，1つのシンボルの対応領域は連続させてとることが必要条件になる.この結果，数直線利用符号における最小必要符号長Lは該当系列の生起確率（符号化を行う上での想定確率）をPとするとき，

$$-\log P \le L < -\log P + 2 \qquad (1)$$

で与えられる．

通常のブロック符号化では

$$-\log P \le L < -\log P + 1 \qquad (2)$$

となるので，この1ビットの差は1つのシンボルの対応領域は連続させてとるという条件，あるいはそれから必然的に導かれる,「符号割り当てに際し,シンボル系列を出現確率順で並べ替えることが許されない」という条件に起因している．ただ算術符号での符号長の上限は符号系列全体での冗長度を議論しており，ブロック符号においては通報あたりの冗長度を議論しているため，実際はむしろ算術符号の方がブロック符号より高い符号化効率が期待できると考えてよい．

以上の考えに基づき，数直線利用符号を実用化する算術符号は以下のように定義できる [10]．

［算術符号の定義］

[0,1)の数直線上でシンボルの出現確率に応じてその対応領域を分割してゆき，最終的に選定された区間内に有り，他の区間には含まれない領域を表現するのに必要な座標を符号語とする符号形式を算術符号と規定する．座標については，小数点以下N桁の有効数字で表現される2進小数γの座標により $[\gamma, \gamma + 2^{-N}]$ で表現される領域が表現されるものとする．上記定義に従って対応区間に完全に含まれる座標（N bitの符号語γ）を対応区間に対する符号語とすれば復号が可能である．（以上）

2.3 正規化とフラッシュ処理

算術符号では，上述の累積確率を精度よく計算する上で，正規化という処理を導入する．これは，符号化が進むにつれ有効領域がどんどん小さくなってゆくので，演算精度を保つために，有効領域の大きさをレジスタサイズで決まる最大値の1/2以上に回復させる処理である．具体的には，有効領域のレジスタのMSBが0になれば1bitシフトして，領域を2倍にするビット単位方式が普通で．これは，等価的に1bitの符号出力追加に対応する．処理の高速化のために有効領域が最大有効値の1/256以下になったときに正規化を行うバイト単位方式もあるが，その場合は，後述の最小確率にも影響が生じる．

なお，上記の定義に従う算術符号において，算術符号の一意復号可能性とは「その後どのような符号データ系列が続いても対応領域に存在することが保証されている」ということである [10]．このとき最終シンボル符号化時のフラッシュビット長（レジスタ内の数直線アドレスデータの出力桁数）は最終正規化有効領域をSとすると $(1/2<S\le 1)$，Sの小数点以下高々2bitでよいことが示される．これが前記の数直線符号を終端する際に，生じる損失の具体的証明にもなっている．

2.4 算術符号によるマルコフ情報源符号化

既に述べたように算術符号はマルチコンテクスト情報源の効率的符号化手段として登場した．では何故それが可能であったかというと算術符号では発生シンボルごとに有効領域の大きさとその下端アドレスを計算すればよいからである．各シンボルに1bit以上の符号語が割り振られるような情報源の場合は情報源の拡大を行わなくて良いが，確率のきわめて高いシンボルが存在すると情報源拡大が望まれる．その場合，ブロック符号では必要となる「通報の設定とそれに伴う符号語の割り当て」作業が不要でありそれぞれのシンボルの符号化を他のシンボルの符号化とは独立に行える

というのが算術符号の大きな特徴である[11]．算術符号で情報源拡大が行われているとみなすかどうかについては若干議論があり，シンボル単位で個別の符号化処理をしていると考えれば情報源拡大ではない．一方最後のシンボルで符号語を割り当てるという点で系列として情報源を拡大していると考えることもできる．一般には算術符号は情報源の拡大という概念自体がないとみなすべきであろう．

3. 算術符号の実用化

以上で述べたように算術符号は高い性能を有するがハフマン符号と比較するとその計算負荷は大きいため，これまでその実用化に向け様々な簡易化手法の検討が行われてきた．その最初は情報源の LPS（劣勢シンボル）出現確率の限定（近似）である．LR 型算術符号[1]がその代表で LPS の想定確率を(1/2)のべき乗に制限することにより LPS の領域計算は有効領域のシフト演算のみで行うことが可能となる．すなわち演算処理の高速化，あるいは LPS の想定確率の記憶容量の減少化が可能といえる．問題は効率の低下で，LPS 確率が 1/2 と 1/4 の丁度中間では 95%程度に下がる．次に低いのは 1/4 と 1/8 との丁度中間で，その効率は 97.2%程度である．このため 1/2 と 1/4 の丁度中間である 3/8 を LPS 確率セットに追加するなどの提案もある．

3.1 減算型算術符号

LR 符号に続いて提案された Q-Coder[12]は LPS の領域幅を有効領域幅によらず固定とするものである．したがって領域計算自体が不要となる．有効領域幅は最大で 2 倍の範囲を動くので想定 LPS 確率は 2 倍の範囲で変動するが（平均的）LPS 確率は任意の値をとれるので，領域計算がより正確ではあるものの確率が限定される LR 符号よりも平均的に高い性能が得られる．Q-Coder では確率が 1/2 近辺で LPS の領域幅が MPS（優勢シンボル）の領域幅を上回ることがあるのでその補正を行う CE 処理[13]を加えた符号が QM-coder[2]や MQ-coder[6]として標準に採用されている．

H.264 で採用された CABAC は減算型算術符号であるが有効領域のサイズを 4 通りに分けてそれぞれの LPS 幅を定めている．すなわち，最大有効領域の 5/8 未満，5/8 以上 6/8 未満，6/8 以上 7/8 未満，7/8 以上である．それぞれの場合の有効領域の期待値は最大有効領域の 9/16,11/16, 13/16, 15/16 であるため，それぞれの有効領域での LPS 幅の比は LPS 確率推定値によらずほぼ 9:11:13:15 となる．ただ LPS 確率の高い方から 3 つのコンテクストでは有効領域の最も小さいケースでの LPS 幅のみを本来より小さくし 128（最大有効領域の 1/4）で固定としている．したがってこれらのケースでは LPS 幅と MPS 幅の逆転は生じない．LPS 確率が高い方から 2 つのコンテクストでは 5/8 以上 6/8 未満，6/8 以上 7/8 未満，7/8 以上の有効領域において MPS/LPS サイズの逆転が生じうることになる．表 1 に 64 状態のうち LPS 確率の高い方から 8 状態のみの LPS 幅を示す．

表 1　CABAC の LPS 幅テーブル（抜粋）

State	Quant CodeIntRange			
	0	1	2	3
0	128	176	208	240
1	128	167	197	227
2	128	158	187	216
3	123	150	178	205
4	116	142	169	195
5	111	135	160	185
6	105	128	152	175
7	100	122	144	166

3.2 算術型 MELCODE

算術型 MELCODE[14]はブロック符号である MELCODE[15]を算術符号として再構成したものである．ブロック型 MELCODE のパラメータは次数（整数値）であり次数個の MPS で 1bit の符号が発生する．このため次数から LPS 領域幅が求まるのに対し算術型 MELCODE のパラメータは直接 LPS 領域幅で与えられる．したがって算術型 MELCODE では非整数次数の MELCODE が定義できることになる．また有効領域幅が「最大有効領域幅の 1/2＋LPS 領域幅」より大きい場合は減算型符号と同様に振舞うがそれより小さいと LPS 領域幅が規則的に小さくなり，本来の算術符号により近い振る舞いをするので減算型算術符号における LPS 確率が 0.5 付近での効率低下がみられないという特長がある．

3.3 状態遷移型算術符号 STT-coder

ハフマン符号では符号ツリーをたどることで符号化／復号が行える．これは符号ツリーのノード（節）アドレスを遷移先とする状態遷移として符号化／復号を記述することであり，符号が確定すると符号語を出力し，遷移先はルート（根）に戻る．したがって算術符号の動作をハフマン符号と同様に状態遷移によって記述できるようにすることで簡易化・高速化につながるといえる．筆者らが提案している STT-coder[16]はこのような考えに基づいている．

算術符号を状態遷移で記述するには下端アドレスをゼロにする必要がある．しかし LPS 領域が上位配置の場合，MPS 発生では常に下端アドレスがゼロのままで問題がないものの LPS 発生では LPS と MPS の境界に有効領域の下端アドレスが移動するため，その位置

を常に正規化後のアドレスゼロに対応させるには，LPS 幅がとりうる値に制約が生じてしまう．そこで有効領域の下端アドレスと確定済み符号の与えるアドレスとの差をオフセットと呼び，オフセット値としてゼロ以外も許容することによりオフセットの種類数に応じて記憶すべき状態遷移のテーブルサイズが増大するものの，許される LPS 幅の自由度が大きくなるため，より高い効率が期待できる．とはいえ，オフセット値が大きいと有効領域の期待値が小さくなってしまうので必ずしもオフセットの種類の増加が符号化効率の上昇につながるとはいえない．検討の結果，たとえば 6 ビットレジスタシステム（有効領域幅 64）では効率が最大であるオフセットの種類数は 4 でオフセット値の組み合わせは(0,16,24,28)となることが示されている．このときの効率低下は理想算術符号に比べ 0.5%程度であり，充分実用的であると判断できる．また，この結果の示す規則性（0,1/4,3/8,7/8）はレジスタ長をさらに拡大したときの設計指針として有効であると期待されている．

STT-coder は減算型算術符号と比較すると各シンボルの領域幅を有効領域幅やオフセットに基づいて決める点ではより複雑であるが ROM によるテーブル参照で領域幅決定に加えすべての処理が実現できることや符号構成においてキャリー伝播の対処が不要という点でより簡易であるともいえる．

4. 算術符号の設計要素

以下では算術符号の代表的設計要素について述べる．

4.1 算術符号のレジスタ長

算術符号の必要レジスタ長についてはレジスタ長が長いほどシンボルの領域幅の精度が高くとれるため効率がよくなると想定できるが，4.2 でも述べるように LPS シンボルの発生確率が高い場合の影響は小さいので，対象情報源の LPS シンボルの最小発生確率に基づいて設計して良い．つまり LPS シンボルの発生確率が低い情報源に対してはレジスタ長をそれだけ長く設定する必要がある．

そこで符号化コンテクストの LPS 発生確率によって実質的な符号化レジスタ長を変化させることも検討の価値があると考えられる．たとえば 4bit-coder を一時的に 3bit-coder に縮退させようとすると有効領域幅によって表 2 のような LPS 割り当て幅の制限が生じる．この例であれば有効領域幅によらず適切な LPS 比率が選択できるためには情報源の LPS 確率が 1/3 近辺以上である必要があり，もともと 3bit で設計された場合と比べ縮退の時点では効率が低下するがその後はレジスタ長が短くでき簡易化が望めるといえる．

表 2 レジスタ長の縮退

4bitcoderから3bit coderに縮退させる場合

有効領域幅	LPS幅	LPS比率	LPS幅	LPS比率	LPS幅	LPS比率	LPS幅	LPS比率
16	2	0.125	4	0.250	6	0.375	8	0.500
15	1	0.067	3	0.200	5	0.333	7	0.467
14	2	0.143	4	0.286	6	0.429		
13	1	0.077	3	0.231	5	0.385		
12	2	0.167	4	0.333	6	0.500		
11	1	0.091	3	0.273	5	0.455		
10	2	0.200	4	0.400				
9	1	0.111	3	0.333				

3bitcoder

有効領域幅	LPS幅	LPS比率	LPS幅	LPS比率	LPS幅	LPS比率	LPS幅	LPS比率
8	1	0.125	2	0.250	3	0.375	4	0.500
7	1	0.143	2	0.286	3	0.429		
6	1	0.167	2	0.333	3	0.500		
5	1	0.200	2	0.400				

4.2 確率近似

算術符号の利点は設定されたシンボル出現確率をもつ情報源がそのまま符号化方式のパラメータを導くことである．各種情報源に応じて個別に方式を設計するのは現実的ではないので，あらかじめモデル情報源を準備することにすると，モデル情報源の設定が符号化のパラメータ設定となる．理想的な符号化が可能と仮定して符号化効率の最悪値を設定すると一つの状態（モデル情報源）がカバーすべき MPS/LPS の出現確率比範囲が決まり符号化対象情報源の分布範囲にいくつのモデル情報源が必要となるかが求まる．逆にモデル情報源数を先に決めれば最適な状態（符号化パラメータ）が切り替わる際の効率を共通化することで符号化効率の最悪値が定まる．もちろん現実の符号化においてはレジスタ長の制限や確率近似による絶対的な効率低下はあるが，それを一様とみなせば，相対的な効率設計が可能と言える．

1) 情報源の状態数

図 1 は最悪値を 99%として状態数を設計したもので 6bit レジスタの対応範囲では 8 状態でよいことが示されている [16]．そこで，より長い符号レジスタの下では状態数がいくつ必要であるかを求めたのが表 3(a)である．ここで 1-p が LPS 確率を表している．この表より，8bit レジスタでは 10 状態，10bit では 12 状態，12bit では 14 状態，16bit では 16～17 状態が必要といえる．表 2 で LOG は 1/(1-p)の対数（底が 2）を表している．

次に，図 1 の各状態の切り替わりにおける符号化効率の最悪値を 99.5%として設計すると表 3(b)に示すモデル情報源セットが得られる．状態数は 6bit レジスタでは 11 状態，8bit では 14 状態，10bit では 17 状態，12bit では 19 状態，16bit では 23～24 状態となる．

表 3(a)や表 3(b)に示す LPS 確率のセットは LPS 確率値が丁度この値（センター値）をとるときに効率が 100%になることを示している．また相隣るセンター値の中間でどちらのモデル情報源に対する符号を適用しても丁度同じ効率値をとる LPS 確率値を境界値とよぶ．

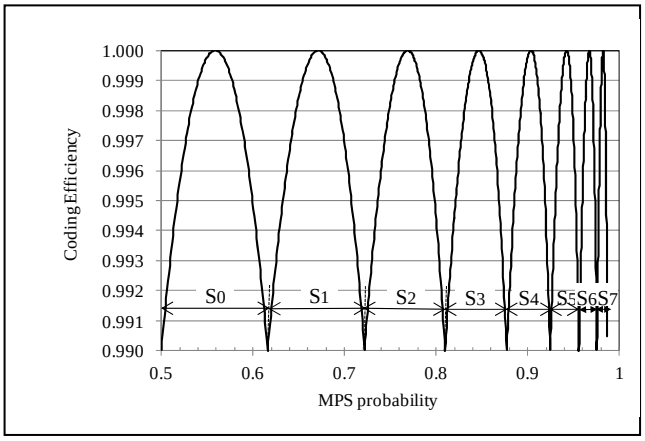

図 1　各状態での理想符号化効率

表 3(a)　最低符号化効率が 99%でのモデル情報源

#	1-p	1/(1-p)	LOG
1	0.44104090	2.267363	1.181016
2	0.32860000	3.043214	1.605596
3	0.23094600	4.330017	2.114373
4	0.15329500	6.52337	2.705617
5	0.09635100	10.37872	3.375557
6	0.05752600	17.38344	4.119642
7	0.03272350	30.55908	4.933529
8	0.01777967	56.24401	5.813628
9	0.00924430	108.1748	6.75722
10	0.00460520	217.1458	7.76252
11	0.00219989	454.5682	8.828353
12	0.00100813	991.9356	9.954103
13	0.00044330	2255.809	11.13943
14	0.00018706	5345.878	12.38421
15	0.00007575	13202.02	13.68847
16	0.00002943	33976.62	15.05225
17	0.00001097	91128.63	16.47562

表 3(b)　最低符号化効率が 99.5%でのモデル情報源

#	1-p	1/(1-p)	LOG
1	0.45834000	2.181786	1.12551
2	0.37698500	2.652625	1.407421
3	0.30121000	3.319943	1.731158
4	0.23378500	4.277434	2.096746
5	0.17636200	5.670156	2.503388
6	0.12942200	7.726662	2.949845
7	0.09249000	10.81198	3.434559
8	0.06444000	15.51831	3.9559
9	0.04381900	22.82115	4.5123
10	0.02910900	34.35364	5.102391
11	0.01890610	52.89298	5.725004
12	0.01201360	83.239	6.379188
13	0.00747240	133.8258	7.064213
14	0.00455100	219.7319	7.779601
15	0.00271485	368.3445	8.524912
16	0.00158655	630.2984	9.299891
17	0.00090839	1100.849	10.1044
18	0.00050961	1962.285	10.93832
19	0.00028013	3569.771	11.80162
20	0.00015089	6627.301	12.69421
21	0.00007964	12556.19	13.61611
22	0.00004119	24277.74	14.56735
23	0.00002087	47905.34	15.5479

境界値のセットは表 4 に示すとおりで，この境界値での（理想）符号化効率が 99%の場合のセンター値が表 3(a)に，99.5%の場合のセンター値が表 3(b)に対応する．ここで 1-p=0.5 はモデル情報源#1 で MPS の値が異なる場合の境界となる．境界値は相隣るセンター値の相加平均と相乗平均との間に位置することが確かめられ、LPS 確率が高い部分ではより相加平均に近く，LPS 確率が低い部分ではより相乗平均に近いといえる．

表 4　最低符号化効率が 99.5%,99%でのモデル情報源の境界値

	1-p (99.5%)	1-p (99%)
-1	0.50000000	0.50000000
1-2	0.41728750	0.38378200
2-3	0.33840400	0.27801300
3-4	0.26659200	0.19009700
4-5	0.20407200	0.12293500
5-6	0.15190000	0.07541300
6-7	0.11005200	0.04401850
7-8	0.07769300	0.02451550
8-9	0.05350600	0.01305620
9-10	0.03598250	0.00665930
10-11	0.02365050	0.00325628
11-12	0.01520430	0.00152740
12-13	0.00956570	0.00068750
13-14	0.00589220	0.00029698
14-15	0.00355450	0.00012313
15-16	0.00210055	0.00004899
16-17	0.00121615	0.00001871
17-18	0.00068991	
18-19	0.00038350	
19-20	0.00020889	
20-21	0.00011150	
21-22	0.00005832	
22-23	0.00002989	

なお，STT-coder のシミュレーション [16]では 6bit レジスタ，オフセット値 4 種に限定という条件下でも S_0 から S_4 程度までによりカバーされる範囲では理想の 0.5%程度の効率低下（98.5%～99.5%）でおさまっている．つまりこのような制限下であっても LPS 確率が低い部分を除き，理想符号長からの後退はきわめてわずかであるといえる．

2)　動的学習での状態数

次に動的学習の検討の前提として最適な確率推定からずれた場合の効率低下について述べる．表 3(a)の例では最適な確率推定が行われた場合，理想符号であれば常に 99%以上の効率が得られるが，もし状態推定が 1 レベルずれるとセンターでの確率値に対しては確率推定が低い側（左）および高い側（右）にずれた場合における効率はそれぞれ 95.5%，96.5%程度となる．また境界での確率値に対しては確率推定が低い側(左)および高い側（右）に 1 レベルずれた場合の効率はそれぞれ 90.5%，92.5%程度となる．（図 2 参照）

もし表 3(b)のように 99.5%を最悪値とするモデル情報源であれば図 3 に示すように 1 段階のずれでの効率はセンター値では 97.5%，境界の確率値では 94.5%以上になる．動的符号化では状態推定がずれた場合の効率も考慮の必要があるので動的符号化も含めた情報源モデルの設計においては静的符号化効率の最悪符号化効率値のみに着目するのでは不十分といえる．

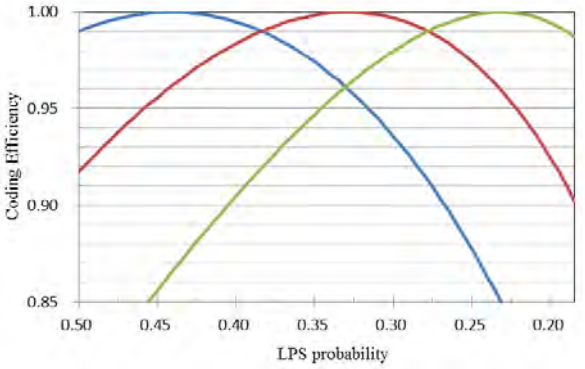

図 2　最悪効率 99%セットでの効率変動

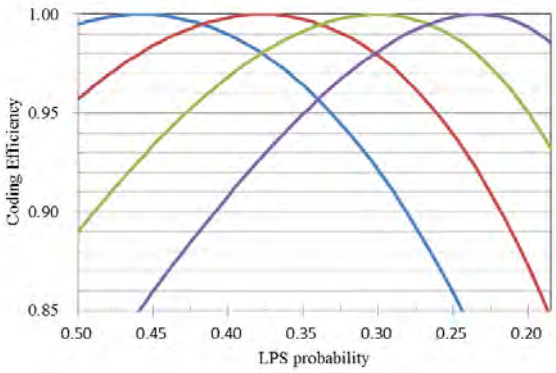

図 3　最悪効率 99.5%セットでの効率変動

STT-coder における状態数に関する研究結果 [16]を紹介しておく．8 状態のままでの動的学習では効率低下が大きいが，たとえば 2 倍の 16 状態にすると 2.5%程度の低下でおさまる．図 2 のモデル情報源が対応する範囲を 16 個の状態に分割すると 16 状態をすべて異なる MPS/LPS 比率とすることができるので最低符号化効率は 0.996 程度にできる．ただ 16 状態ごとに有効領域幅に応じた MPS/LPS 割当表が必要なのでメモリ容量を節約するために符号化状態としては 8 状態のままとし，遷移の過程として 1 符号化状態を 2 つのサブ状態で構成すると，独立 16 状態に対する効率の低下は僅かに 0.2%程度で済む．

なお Q-Coder では 13bit レジスタで 30 状態を用いている．その 6bit レジスタ部分を取り出すとおよそ 20 状態であり，ほぼ対応していると言える．

4.3　ROM のサイズ

1)減算型算術符号

減算型算術符号において各シンボルの領域割り当てを与える ROM は入力が状態番号で出力は LPS のサイズである．状態遷移に関しては現在の状態番号，発生シンボルの MPS/LPS，状態更新の有無が入力で出力は MPS 再正規化での次期状態番号，および LPS 再正規化での次期状態番号，MPS 反転の有無 1bit である．なお Q-Coder や QM-coder，MQ-coder では LPS 出現においては必ず再正規化されるので LPS 発生での更新判定は不要である．

2)STT-coder

STT-coder ではすべての符号化作業が ROM によるテーブル参照により行われる．6bit レジスタの coder の例では入力が有効領域の上端アドレス（5bit）・下端アドレス（オフセットの種類数として 2bit），発生シンボルの MPS/LPS（1bit），状態番号（最低符号化効率 99%として 3bit）の計 11bit である．出力は符号化後の有効領域の上端アドレス（5bit）・下端アドレス（オフセットの種類数として 2bit），符号長（符号なしを含め 3bit），符号語（6bit）である．もし，符号長と符号語を独立に出力せずあらゆる符号語のパターンを番号付け（code 化）した結果を出力し，そのあとでデコーダを外部接続して符号長と符号語に変換するなら，6bit 符号が 64 通り，5bit 符号が 32 通りであり，以下同様にして 1bit 符号が 2 通り，符号無し 1 通りを含めて 127 通りなので符号長と符号語を含めて 7bit 出力でよい．よって ROM の出力 bit 数は 14bit である．

8bit レジスタであれば入力情報は有効領域の上端アドレス（7bit）・下端アドレス（オフセットの種類数も同じ 4 種として 2bit），発生シンボルの MPS/LPS（1bit），状態番号（最低符号化効率 99.5%としても 4bit）の計 14bit である．出力は符号化後の有効領域の上端アドレス（7bit）・下端アドレス（オフセットの種類数として 2bit），符号長（符号なしを含め 4bit），符号語（8bit）である．もし，出力にデコーダを外部接続するなら上記と同様に符号長と符号語を含めて 9bit の情報 で済むので ROM の出力は 18bit である．

状態遷移については STT-coder では MPS 発生でも LPS 発生でも条件付きでの遷移であるので 6bit レジスタの場合入力が状態番号（3bit），各状態における上下サブ状態（1bit），発生シンボルの MPS/LPS（1bit），状態更新判定用入力である．

状態更新判定には確率変数が必要であり，その比率の精度，比率の最小値の逆数として必要 bit 数が定まる．また有効領域の値を流用できると ROM サイズは不変であっても装置としての簡易化につながるといえる．

4.4　学習制御

Q-Coder の学習制御は再正規化で状態遷移を行うこと, MPS での再正規化では 1 段階 LPS 確率の低い方へ移動し，LPS での再正規化では LPS 確率が上位の状態では LPS 確率の高い方へ 1 段階，LPS 確率が下位の状態では LPS 確率の高い方へ 2 段階移動するという簡潔な規則である．その設計規則の背景についてはあまり詳しく記述されていないが，もともと各状態での LPS サイズの設計も含め学習を前提にシステム設計が行われており，シミュレーションで LPS 確率に依らずフラットな効率特性が得られること，特定の入力系列で極端な効率低下が生じないことを重視して設計されていると思われる．

QM-coder では状態数が 113 と Q-Coder より多く，Q-Coder にはない初期学習の高速化を実現している．MQ-coder では初期学習の高速化効果をある程度維持しつつ状態数を 47 状態に減らしている．

一方，STT-coder では静的効率に基づいて得られるモデル情報源を前提として動的効率を求め，LPS 発生でも一定比率で状態遷移を行う LPS 遷移比率を導入して効率特性のフラット化を実現し，その絶対的性能から上記のサブ状態を導いている．

4.5　桁上がり制御

算術符号を有限精度のレジスタで実現する上で符号化レジスタから掃き出すべき符号データに桁上がりが波及する可能性があると符号データを即時に送出できない問題がある．

これが桁上がり制御問題であり,算術符号に固有な大きな課題である．桁上がり制御の例としてはビットスタッフィング方式があり，一定桁数の範囲までは「01･･･1」のパターンを送出し，そのあとに制御信号（スタッフィングビット）を挿入し，復号側では挿入信号を解読してけた上がりの有無を復号に反映させる．この方式は MQ-coder にも採用されている．ビットスタッフィング方式より効率の良い方式として筆者らの提案による領域切捨て方式，適応的境界スライド方式などがある [17]．これらは桁上がりが発生しないように有効領域の一部を棄却したり，各シンボル領域の境目をずらすもので，一時的に確率推定を変更していると解釈することができる.なお STT-coder では ROM 駆動によりキャリー伝播の心配は不要である.

4.6　性能評価

設計した算術符号の評価を行う上での問題は有効領域幅の変動である。レジスタ長が短ければたとえば LPS 確率を 0.5 以下で一様分布と仮定し，状態間の遷移確率に基づいて各有効領域幅の滞在確率を理論的に求めることで総合的な効率計算が可能となる．STT-coder でもレジスタ長が 3bit や 4bit 程度であれば、理論解析が充分可能である，しかしレジスタ長が長くなって状態数が多くなったり，確率推定に学習機能を導入して動的性能を求めようとする場合は，コンテクストを複数設定し，全体の LPS 確率分布を一様として情報源モデルを構成し．その出力を充分な回数分観察して性能を求める手法がとられる．

5. むすび

算術符号が多くの画像符号化標準に取り入れられつつある状況のもとで算術符号の設計要素について考察した．算術符号の本質はシンボル系列に対する符号割り当てが他の系列とは独立に行えることであり，その効能がマルコフモデル情報源の実用的符号化の道を拓いたといえる．またその高効率性と学習性能から本来マルコフモデルを想定していなくてもコンテクスト認知を自動的に行うことでマルコフモデルとしての予期しない情報量を実現できることになる．一方でその高速化には多値算術符号の検討が必須と思われる．今後も算術符号の研究解析が盛んになりさらに普及が進むことを期待したい．

参考文献

1) G.G. Langdon, J.J. Rissanen: “Compression of Black-White Images with Arithmetic Coding”, IEEE Trans. on Communications, Vol. COM-29, No.6, pp.858--867 (1981).
2) ISO/IEC 11544, “Progressive bi-level image compression” (Dec. 1993)
3) ISO/IEC 10918-1,“Digital Compression and coding of continuous-tone still images: Requirements and guidelines”(Dec. 1993)
4) ISO/IEC 14495-1:“Lossless and near-lossless compression of continuous-tone still images - baseline” (Dec. 1999)
5) ISO/IEC 14495-2:“Lossless and near-lossless compression of continuous-tone still images – Extensions” (Mar. 2002)
6) ISO/IEC15444-1:”JPEG2000 Image Coding System: Core coding system”(Dec.2000)
7) ITU-T H.264 | ISO/IEC 14496-10 Advanced Video Coding (2014)
8) ITU-T H.265 | ISO/IEC 23008-2 High Efficiency Video Coding (2015)
9) 小野：“算術符号とその画像符号化標準への適用” 画像電子学会 第 39 回 VMA 研究会 #4 July17, 2015
10) 小野：“マルコフ情報源のエントロピ符号化－算術符号化処理の必然性と MELCODE への適用” 画電誌 Vol.21,No.5, pp.475-485(1992.10)
11) 小野：“算術符号とその画像符号化への適用に関する考察”画電誌 Vol.31,No.5,pp.745-754(2002. 9)

12) W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, Jr., R.B. Arps: “An Overview of the Basic Principles of the Q-Coder, Adaptive Binary Arithmetic Coder”, IBM Journal of R&D, Vol.32, No.6, pp.717-726 (1989).
13) 小野,吉田,木村,木野：“減算型算術符号における符号化効率とその向上方式”画電全大 3(1990)
14) 小野，木村他：“算術符号型 MEL-CODE” 信学春全大 A-152（1989）
15) 大西,上野,小野：“2 値情報源の符号化圧縮”，信学論 vol.60-A, No.12, pp. 1114–1121(1977.12)
16) 上野，小野：“状態遷移テーブル参照型算術符号 STT-coder の設計”画像電子学会誌 Vol.43 No.1, pp.62-70, 2014 年 1 月
17) 小野，木村：“算術符号化における符号長ロスを減少させたけた上り制御方式”，信学論 Vol.J84-A No.3, pp.414-417（2001)