Probability estimation for the Q-Coder

by W. B. Pennebaker
J. L. Mitchell

The Q-Coder is an important new development in binary arithmetic coding. It combines a simple but efficient arithmetic approximation for the multiply operation, a new formalism which yields optimally efficient hardware and software implementations, and a new technique for estimating symbol probabilities which matches the performance of any method known. This paper describes the probability-estimation technique. The probability changes are estimated solely from renormalizations in the coding process and require no additional counters. The estimation process can be implemented as a finite-state machine, and is simple enough to allow precise theoretical modeling of single-context coding. Approximate models have been developed for a more complex multi-rate version of the estimator and for mixed-context coding. Experimental studies verifying the modeling and showing the performance achieved for a variety of image-coding models are presented.

1. Introduction

Arithmetic coding, introduced several years ago by Rissanen [1] and Pasco [2] and generalized by Langdon and Rissanen [3] (see Langdon [4] for a comprehensive review article), is a powerful technique for coding of strings of data symbols. It derives its power from an ability to approach the entropy limit in coding efficiency and to dynamically alter the estimate of the probability of the symbol being encoded.

A new binary arithmetic coding system, the Q-Coder, has been developed as a joint effort between the authors of this paper and colleagues at the IBM Almaden Research Center. The new probability-estimation technique used in the Q-Coder is presented in this paper; companion papers describe the basic principles of the Q-Coder [5], software implementations of the Q-Coder [6], and the arithmetic coding procedures which allow compatible yet optimal hardware and software structures [7, 8]. The Q-Coder is part of a proposal submitted to the CCITT and ISO Joint Photographic Experts Group (JPEG) for color photographic image compression [9].

A description of the general structure of the Q-Coder arithmetic coding section is given in [5, 6]. Briefly, the arithmetic coder contains two key registers, the interval register A and the code register C. The interval register contains the measure of the current probability interval, and the code register contains a pointer to the interval. In order to use fixed-precision integer arithmetic for the coding process, the interval and code registers must be periodically renormalized.

When a given symbol is coded, the interval measure in A is reduced to the subinterval for that symbol, and the code string is repositioned to point within the subinterval. Ideally, the scaling of the interval is done by multiplying the current-interval measure A by the probability estimate of the symbol which occurred. If the less probable symbol (LPS or L) probability q is estimated as Q_e and the more probable symbol (MPS or M) probability p is estimated as $1 - Q_e$, the binary coding process divides the interval into two subintervals, $A \times Q_e$ and $A - (A \times Q_e)$. The multiplication can be avoided by introducing a tight constraint on the renormalization [3–5, 10]. If the probability-interval measure A falls within the bounds $0.75 \leq A < 1.5$, A can be approximated by 1 when multiplying by Q_e. The subintervals are then approximated by Q_e and $A - Q_e$, and the multiplication is avoided.

Although the renormalization is required for the arithmetic approximation, it can serve another important purpose—it can also be used to estimate the probability of the symbol being coded.

A number of different but somewhat related techniques have been used to estimate symbol probabilities. Langdon and Rissanen [11] and Pennebaker and Mitchell [12] have both used confidence-interval techniques to determine whether the current estimate Q_e of the LPS probability should be changed. In [12], the degree to which the confidence limit is exceeded is used to determine the degree to which Q_e should be changed; this gives a multi-rate estimation process. Goertzel and Mitchell [13] used a counting technique with periodic renormalization; although in principle a divide operation is required, the precision is small enough that a lookup table inversion and multiply can be used. Mohiuddin, Rissanen, and Wax [14] devised an intriguing multi-rate adapter in which the choice of estimation rate is based on a local minimization of the code string being generated. Although relatively complex to implement, this technique is quite powerful; we regard it as a standard against which other estimation techniques can be measured. Finally, Helman et al. [15] used a Monte Carlo technique involving the LPS renormalization and symbol counts for updating the estimate of the LPS probability in the Skew Coder [3].

The rest of this paper is devoted to an analysis of a probability-estimation technique in which the probability is estimated solely from renormalization.[1] The renewed attempt to use renormalization as the basis for probability estimation was inspired by earlier work on the Log Coder [12]. In that work the computations for probability estimation were minimized by estimating probability each time one byte of compressed data was generated. While this system proved to be simple to implement and provided accurate estimates, it failed to adapt quickly enough in coding of facsimile data sets. On the other hand, calculating a new probability after the coding of each symbol, as was done by Rissanen and Mohiuddin [10], provided good estimates and fast adaptation but involved far too much computation. Estimation after each renormalization appeared to be an attractive compromise between these two schemes.

In Section 2 the estimation process is described. Section 3 develops the exact theoretical modeling of that process for a single context. Section 4 continues the theoretical modeling for mixed contexts and a random-interval model. Theory and experiments are compared for a single context in Section 5. Section 6 extends the probability estimation to a multi-rate system. Mixed-context coding is analyzed in Section 7.

[1] In unpublished work G. Goertzel and J. L. Mitchell explored and abandoned this possibility because they were unable to obtain good coding efficiencies.

2. Estimation process

The basic concept is as follows: The estimated LPS probability, Q_e, is taken from a fixed table of allowed values. Renormalization occurs either when an L event is encountered or when the interval falls below 0.75 following an M event. When renormalization after an LPS is encountered, the index to the current Q_e is shifted to a larger Q_e. Conversely, whenever the MPS renormalization is encountered, the index is shifted to a smaller Q_e. (The terms *MPS renormalization* and *LPS renormalization* are usually abbreviated as “MPS renorm” and “LPS renorm” in the text following.)

The following approximate calculation suggests that the estimate of the probability obtained from the table of allowed Q_e values will adapt to and closely approach the true LPS probability q of a binary symbol sequence. Given a starting value A for the interval register immediately following the last renormalization, N successive MPS events must occur to reach the MPS renorm point:

$$N = 1 + [\Delta A / Q_e], \tag{1}$$

where Q_e is the current estimated value of q, ΔA is the change in the interval $(0.75 > \Delta A \geq 0)$, and the brackets denote the greatest integer function (rounding down to the nearest integer). The probability of getting N MPS events in a row (and an MPS renorm) is

$$P_{\text{mpsr}} = (1 - q)^N. \tag{2}$$

For simplicity, consider the case where q is small. Taking the natural logarithm of Equation (2) and approximating $\ln(1 - q)$ by $-q$,

$$P_{\text{mpsr}} \simeq e^{-\Delta A (q/Q_e)}. \tag{3}$$

The magnitude of ΔA is dependent on the type of renormalization. If the MPS renorm occurred last, ΔA is close to 0.75; if the LPS renorm occurred last, ΔA is typically somewhat smaller than 0.75. If the effective value of ΔA is assumed to be an appropriate average and the change in Q_e is the same for both types of renorms, the renorm probabilities are balanced at the point where

$$\frac{P_{\text{mpsr}}}{1 - P_{\text{mpsr}}} = 1. \tag{4}$$

Solving for Q_e,

$$Q_e = \frac{\Delta A}{\ln(2)} q. \tag{5}$$

The equilibrium is stable at this balance point. If Q_e is too large, P_{mpsr} is also large and the system tends to move to smaller Q_e. Conversely, if Q_e is too small, P_{mpsr} is small and the system tends to move to larger Q_e. Therefore, the system adapts to and balances approximately at the point $q = Q_e$. Although these calculations are approximate, exact calculations which follow the same general approach and prove the point more rigorously are described below.

As will be seen, the coding efficiencies achieved by the Q-Coder with this probability estimator for pseudorandom data sets are usually not as good as those obtainable with simple estimates of probability from counts [13]. Coding inefficiency is due partly to the lower coding efficiency inherent in the arithmetic approximation to the multiply, partly to small but systematic errors in Q_e, partly to the granularity of the set of allowed values of Q_e, and partly to the intrinsic distribution in Q_e resulting from the stochastic estimation process. However, the estimation process tracks variations in symbol probability very well. Consequently, the coding efficiency achieved with the less stable symbol probabilities encountered in many real coding environments is extremely good, competitive with the best that can be done by any technique currently known.

This estimation process works well for both single-context and mixed-context coding. For a single-context system, the renormalization process is used to estimate only one probability. For mixed-context coding, the coding decisions are *conditioned* by past history, and a different probability must be estimated for each conditioning state or context. It is perhaps somewhat unexpected that renormalization of a single A register can be used to estimate the many different probabilities required in the mixed-context case.

3. Modeling of the estimation process for a single context

The estimator can be defined as a finite-state machine, that is, a table of Q_e values and associated next states for each type of renorm (i.e., new table positions). The rate of change of Q_e is determined by the granularity of the table of Q_e values and by the new state associated with each Q_e value for the two types of renorms. **Figure 1** diagrams sections of the actual finite-state machine used to estimate the probabilities. The leftmost section illustrates the exchange of MPS and LPS definitions at $Q_e \cong 0.5$ (k_{ex} is defined to be the particular state index, k, where this exchange occurs). The center section shows a region where the finite-state machine changes from a single-state jump on LPS to a double-state jump. Some parts of the finite-state machine require a jump of more than one state in order to correctly estimate the probability. The rightmost section shows the diagram for the smallest values of Q_e. This last section shows how the transition at the MPS renorm for the smallest Q_e value is returned to that state.

Conditional changes in estimated probability, such as changing Q_e only after the occurrence of two MPS renorms in a row, can readily be incorporated by allowing multiple entries of a given Q_e value. A related form of this can be seen in the diagram for the lowest Q_e state, where entry to that state can only occur after two MPS renorms in sequence. Handling conditional effects in this manner greatly simplifies the theoretical treatment, the only penalty being the need to solve a relatively large number of simultaneous equations when complex conditional structures are being considered.

Figure 2 illustrates the sequencing of the probability estimator for an LPS followed by a sequence of MPSs. In Figure 2 the ordinate is the interval (A-register) value, and the abscissa is the discrete allowed values of Q_e. The LPS renormalization causes a transition to a known A-register value and a known state in the finite-state machine (in this case from a Q_e of 0.42206 to the appropriate starting A-register value at $Q_e = 0.46893$. (The particular Q_e values in the figure are taken from the actual optimized 5-bit Q_e values in Table 1, shown later. As MPSs are coded, the interval decays until it drops below 0.75. At that point a transition is made to a smaller Q_e, and the interval is renormalized by doubling until it is greater than 0.75. In most cases only one doubling is needed. Thus, the pair of doublings shown at $Q_e = 0.32831$ is the exception rather than the rule. Whether one or two doublings occur is of no consequence for the probability estimation. However, since each doubling produces one bit in the code string (ignoring bit stuffing for a carry), the extra doubling is important in the calculation of the coding efficiency.

Figure 2 illustrates the sequencing behavior which underlies the calculation of the estimation process. The first half of the problem is determining the probability that the estimate will be at each of the allowed Q_e values. If we define n_k as the occupation probability for the state corresponding to $Q_e[k]$ (the kth allowed value of Q_e), balance of transition probabilities into and out of the kth state gives

$$\sum_j n_j X_j (1 - X_j)^{t_{kj}} - n_j X_j (1 - X_j)^{r_{kj}} = n_k X_k , \qquad (6)$$

where $X_k = q$ for $k \geq k_{\mathrm{ex}}$ and $X_k = 1 - q$ for $k < k_{\mathrm{ex}}$. The symbols r_{kj} and t_{kj} are defined below. The table of allowed Q_e is defined to have mirror symmetry at the boundary between k_{ex} and $k_{\mathrm{ex}} - 1$. Thus, k_{ex} is the index in the table of Q_e where the definitions of least probable symbol and most probable symbol are exchanged. (For $k < k_{\mathrm{ex}}$ the table provides an estimate of $1 - q$ rather than q.)

The first term in the summation represents the transition probability into the state at $Q_e[k]$ from all states j which can reach the state k *by an LPS followed by a sequence of MPSs.* The exponent t_{kj} is the number of MPSs needed to just enter the kth state when starting from an LPS at state j. Thus, for the example sketched in Figure 2, state j is marked with an asterisk, and state k could be any one of the states which is reached by the MPS sequence following the LPS.

The second term in the summation represents the transition probability out of the state k, given that the MPS

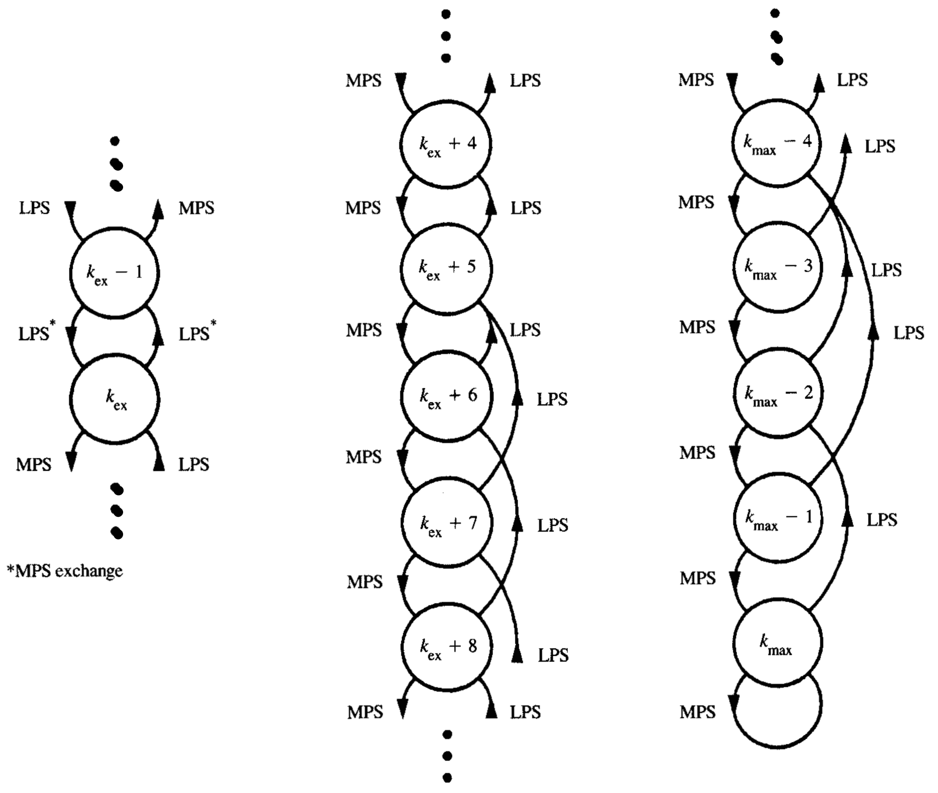


Figure 1

Sections of the state diagram for the probability estimator.

sequence continues until the interval decays below 0.75 for the kth state. The exponent r_{kj} is the number of MPSs needed to just leave the kth state, counting from the symbol after the LPS at state j.

The summation therefore represents the net gain in n_k due to transitions from all states j which can reach state k through an LPS followed by a sequence of MPSs. It is equated to the probability of transition out of state k due to an LPS event. (All probabilities are per-symbol encoded.) The probability of transition out of state k via the LPS path is the probability of the LPS multiplied by the occupation probability n_k.

Normalization requires

$$\sum_j n_j = 1. \tag{7}$$

The numerical solution of these equations can present problems, in that the n_k can be vanishingly small when the index k is far from the value for the most probable value of Q_e. Therefore, the equations are reduced to a subset involving only the n_k near the most probable value of Q_e. Contributions from members outside this range are assumed to be zero. The set of equations must be large enough that the error in truncating the set is small, yet small enough to avoid arithmetic precision problems in the calculation of determinants by the method of Gaussian elimination. Except near the end of the Q_e table, the center value of k for the subset is defined as the index for which $Q_e[k]$ is closest to q.

Because the table of allowed values of Q_e is finite in extent, the equations must be reformulated to take end conditions into account. This is done by assuming that either $t_{kj} = 0$ or $r_{kj} = \infty$ in Equation (6). The latter condition exactly describes the closure of the state diagram for k_{max} in Figure 1, in that the system cannot exit from the k_{max} state after the MPS renorm. It also approximates the closure if the equation set is truncated before k reaches k_{max}. Truncation near the other endpoint, $k = k_{min}$, is described by one of the two approximations, the choice depending on whether a particular k is less than the exchange index, k_{ex}, or not. The two assumptions are equivalent to assuming that either LPS or MPS renormalization is highly unlikely. Note that the approximate closure condition at the smallest value of k is not needed, as it is replaced by Equation (7).

Given a table of Q_e values and associated index changes to new Q_e values for each renormalization path, these equations provide an exact solution of the probability of the system being at each $Q_e[k]$. However, they hold only for single-context coding.

The second half of the problem is the calculation of coding rate. Refer again to Figure 2. The current occupation of each state in the system is determined by the balance of LPS and MPS transitions into and out of that state. For each Q_e the probability of the LPS renormalization is known by definition (q), and the probability of each succeeding MPS renormalization is readily calculated. The bits generated by each renormalization are also readily calculated. The net bit rate R_k for the kth state is thus

$$R_k = n_k X_k \left[B_{\text{LPS},k} + \sum_j B_{\text{MPS},j} (1 - X_j)^{r_{kj}} \right], \qquad (8)$$

where j ranges over all MPS renormalizations which can occur following the LPS renorm. $B_{\text{LPS},k}$ is the number of bits generated in renormalizing $Q_e[k]$ to the allowed interval range. $B_{\text{MPS},j}$ is the number of bits generated by the jth MPS renorm. As defined earlier, $X_k = q$ for $k \geq k_{ex}$, $X_k = 1 - q$ for $k < k_{ex}$, and the exponent r_{kj} is the number of MPSs needed to reach the jth MPS renorm after the LPS event from state k.

The total coding rate in bits per symbol is therefore

$$R = \sum_k R_k. \qquad (9)$$

4. Mixed contexts: The random-interval model

The calculations in Section 3 are not applicable to coding of mixed-context symbols. If the context varies from one symbol to the next, Q_e also varies. The calculation of probabilities of MPS renormalization and the associated bit rate is therefore far more complex. Let us consider the following hypothesis: The probability of the various interval values is sufficiently randomized by the effects of multiple contexts that the interval-register values are uniformly distributed in the interval from 0.75 to 1.5.

Assuming that the above hypothesis is valid, the following equations give the LPS renormalization probability $P_{1,k}$ and the MPS renormalization probability $P_{m,k}$:

$$P_{1,k} = q, \qquad (10)$$

$$P_{m,k} = (1 - q)(Q_e[k]/0.75). \qquad (11)$$

The equations describing the balance in transition probabilities are similar to those developed in Section 3, except that the probability of the MPS renorm is calculated

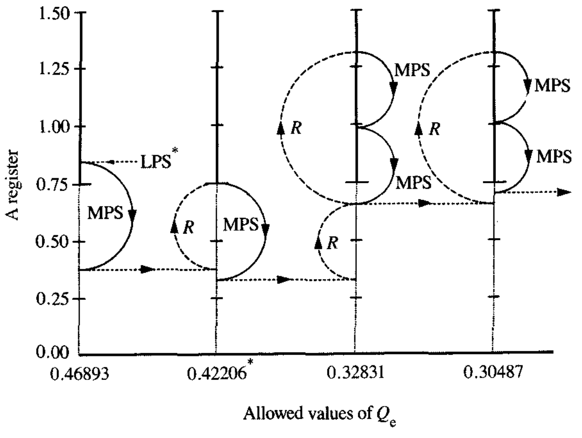


Figure 2

Example of the sequencing of the probability estimator following an LPS.

from Equation (11) rather than from the equations for the probability of a sequence of MPSs. Therefore, the balance for state k is given by

$$\sum_s n_s P_{1,s} + \sum_t n_t P_{m,t} - n_k (P_{m,k} + P_{1,k}) = 0, \qquad (12)$$

where s is summed over all states which can make a transition to k via LPS, and t is summed over all states which can make a transition to k via a single MPS renorm. The normalization condition, Equation (7), completes the set of equations to be solved. Numerical precision again requires that the set of equations be truncated. Therefore, endpoint conditions are handled in the same manner as discussed in Section 3.

The calculation of coding efficiency is done differently for the random-interval model. For a given interval A and a given estimated LPS probability Q_e, the relative coding efficiency is

$$E = \frac{\sum_k n_k R_k - H}{H}, \qquad (13)$$

where H is the entropy and R_k is the bit rate per symbol for state k. Defining $p = 1 - q$, the entropy is given by

$$H = -q \log_2(q) - p \log_2(p), \qquad (14)$$

and, for a uniform distribution of A values in the interval 0.75 to 1.5, R_k is given by

$$R_k = \frac{1}{0.75} \int_{0.75}^{1.5} \left[-q \log_2\left(\frac{Q_e[k]}{A}\right) - p \log_2\left(1 - \frac{Q_e[k]}{A}\right) \right] dA. \qquad (15)$$

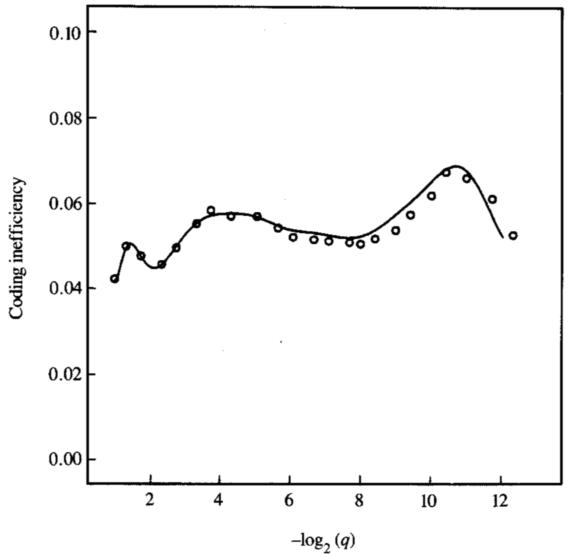


Figure 3

Coding inefficiency as a function of q, calculated and measured for a single context using Table 1. The measured points were obtained from pseudorandom data sets of fixed q.

Table 1 Probability estimates for 5-bit Q_e.

Q_e index	Q_e value	Decimal Q_e value	Decr (LPS)	Incr (MPS)	MPS exch
0	X'0AC1'	0.50409	0	1	1
1	X'0A81'	0.49237	1	1	0
2	X'0A01'	0.46893	1	1	0
3	X'0901'	0.42206	1	1	0
4	X'0701'	0.32831	1	1	0
5	X'0681'	0.30487	1	1	0
6	X'0601'	0.28143	1	1	0
7	X'0501'	0.23456	2	1	0
8	X'0481'	0.21112	2	1	0
9	X'0441'	0.19940	2	1	0
10	X'0381'	0.16425	2	1	0
11	X'0301'	0.14081	2	1	0
12	X'02C1'	0.12909	2	1	0
13	X'0281'	0.11737	2	1	0
14	X'0241'	0.10565	2	1	0
15	X'0181'	0.07050	2	1	0
16	X'0121'	0.05295	2	1	0
17	X'00E1'	0.04120	2	1	0
18	X'00A1'	0.02948	2	1	0
19	X'0071'	0.02069	2	1	0
20	X'0059'	0.01630	2	1	0
21	X'0053'	0.01520	2	1	0
22	X'0027'	0.00714	2	1	0
23	X'0017'	0.00421	2	1	0
24	X'0013'	0.00348	3	1	0
25	X'000B'	0.00201	2	1	0
26	X'0007'	0.00128	3	1	0
27	X'0005'	0.00092	2	1	0
28	X'0003'	0.00055	3	1	0
29	X'0001'	0.00018	2	0	0

Integration yields

$$R_k = qR_{k,\mathrm{LPS}} + pR_{k,\mathrm{MPS}}, \tag{16}$$

$$R_{k,\mathrm{LPS}} = \log_2(e)\,\frac{X_1 \ln(X_1) - X_0 \ln(X_0) - X_1 + X_0}{X_1 - X_0}, \tag{17}$$

$$R_{k,\mathrm{MPS}} = \frac{\log_2(e)}{X_1 - X_0} \times \left[X_1 \ln\left(\frac{X_1}{X_1 - 1}\right) - X_0 \ln\left(\frac{X_0}{X_0 - 1}\right) + \ln\left(\frac{X_1 - 1}{X_0 - 1}\right)\right], \tag{18}$$

where $X_0 = 0.75/Q_e[k]$ and $X_1 = 1.5/Q_e[k]$.

5. Comparison between theory and experiments for single-context coding

In **Figure 3**, the results of calculations (solid curve) based on the equations derived in Sections 3 and 4 are compared to experimental results (circles). For single-context coding, the agreement between experiment and the exact theory for that case is excellent (as expected). The corresponding table of estimated Q_e and the associated schedule of changes in index following renormalization are given in **Table 1**. This table was selected after much experimentation [5]; it represents the best compromise among simplicity, minimum storage requirements for each context (6 bits),[2] reasonable coding efficiency for fixed statistics, and good performance on mixed-context data obtained from both facsimile-compression models and continuous-tone image-compression models.

The Q_e values in Table 1 are expressed as hexadecimal integers. Divide these Q_e values by X'1000' × 4/3 to convert to the decimal fractional representation. The "Decr" column shows the decrement in the Q_e index when moving to larger Q_e following an LPS renormalization. The "Incr" column shows the increment in the Q_e index when moving to smaller Q_e following an MPS renormalization. Where the "MPS exch" column entry is 1, an LPS will cause an exchange in the MPS definition.

Figure 4 shows several theoretical (dashed curves) and experimental (solid curves) distributions of Q_e for the 5-bit Q_e case. Again, agreement between calculation and measurement is excellent.

Figures 5 and **6** show similar experimental and theoretical calculations for the 6-bit Q_e table (**Table 2**). As might be expected, the finer granularity of this table significantly decreases the coding inefficiency for stationary statistics. (Compare Figure 3 with Figure 5.)

Figure 6 shows several calculated and measured distributions of Q_e for the 6-bit Q_e case. Comparing these distributions to those in Figure 4, the increase in coding

[2] The initial impetus to use very small amounts of storage per context came from G. G. Langdon. Somewhat to our surprise, he was able to demonstrate that our estimation technique could achieve relatively good performance with a table of allowed Q_e having less than 32 entries. We subsequently worked jointly to optimize this single-rate system.

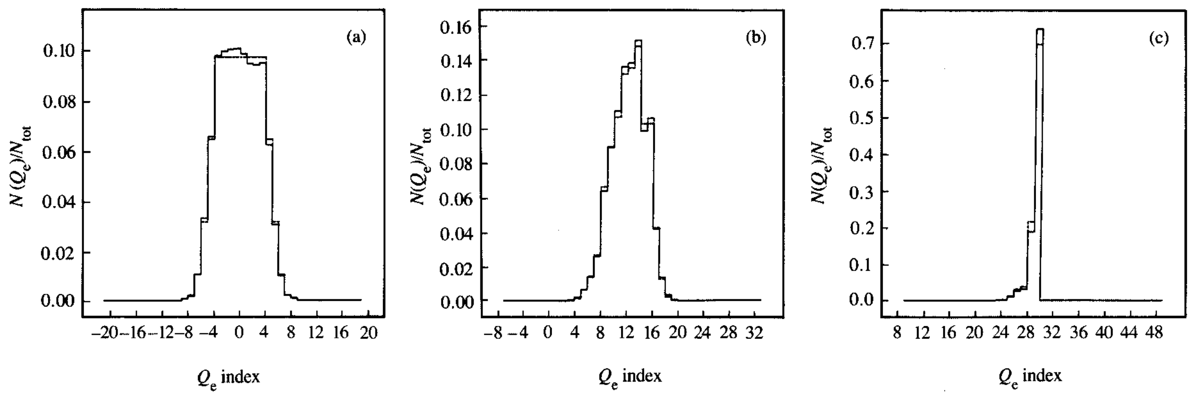


Figure 4

Distribution in Q_e usage for the 5-bit Q_e table (Table 1): (a) $q = 0.5$; (b) $q = 0.1$; (c) $q = 0.0002$. The dashed curve shows theoretical distributions; the solid curve, experimental distributions.

efficiency with a 6-bit Q_e is due to the reduced spread in the distribution. In both cases, the peak in the distribution is quite close to the desired Q_e. The coarser granularity of the 5-bit Q_e table is not the only factor. The changes in table position for the 5-bit Q_e are often larger as well. This increases the spread in the Q_e distribution.

The effect of the estimation process can be eliminated by choosing the Q_e so as to minimize the coding inefficiency. Such a procedure might be appropriate when the probability is known *a priori*. The coding inefficiency for this special case is shown in **Figure** 7 for two cases. The first case (solid line) is for the 12-bit integer representation of Q_e with no additional granularity introduced. This curve gives the lower bound for the coding inefficiency that can be achieved with this integer representation. Quantization of Q_e to the 12-bit integer representation causes the sequence of distinct minima which is noticeable for $q < 2^{-10}$. The sequence of distinct minima for q larger than about 2^{-3} reflects the fact that the bit rate stays constant over short intervals in Q_e. These intervals of constant bit rate result from the arithmetic approximation and renormalization used in the Q-Coder. The dashed line in Figure 7 represents the 12-bit integer representation and the granularity of the 5-bit Q_e table.

One feature of Tables 1 and 2 is the avoidance of Q_e values which renormalize to X′1000′. For the integer representation chosen for the tables this is the minimum A value allowed before MPS renormalization must occur. Consequently, when the Q_e value renormalizes to an A value too close to X′1000′, the probability of the MPS renorm becomes very large. If A is exactly X′1000′, any MPS will

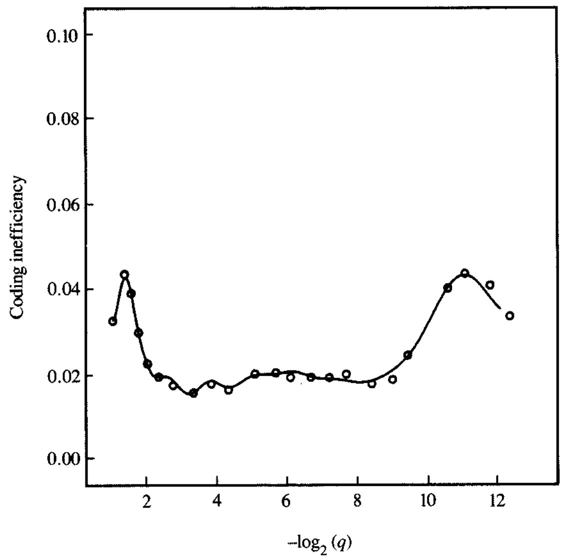


Figure 5

Coding inefficiency as a function of q, calculated and measured for a single context using Table 2.

immediately trigger an MPS renorm. In single-context coding, this creates a trap for the estimator. Note, however, that the smallest Q_e entry in the table must renormalize to

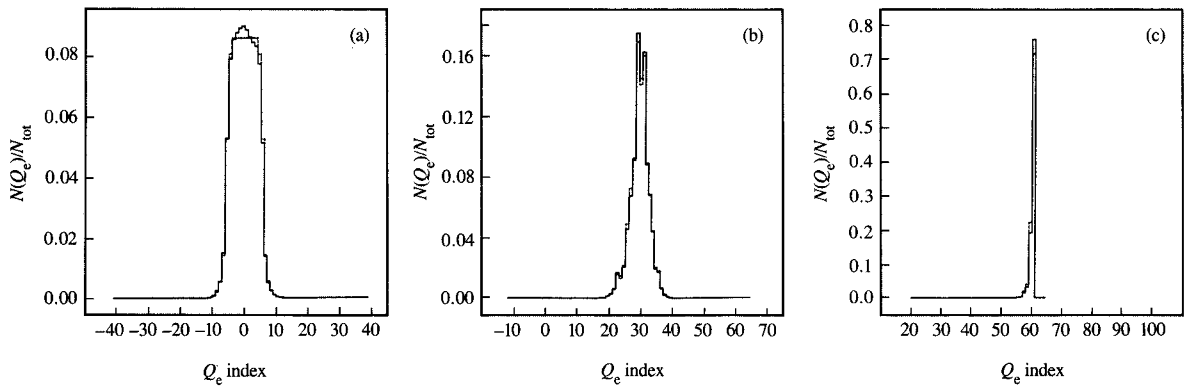


Figure 6

Distribution in Q_e usage for the 6-bit Q_e table (Table 2): (a) $q = 0.5$; (b) $q = 0.1$; (c) $q = 0.0002$. The dashed curve shows theoretical distributions; the solid curve, experimental distributions.

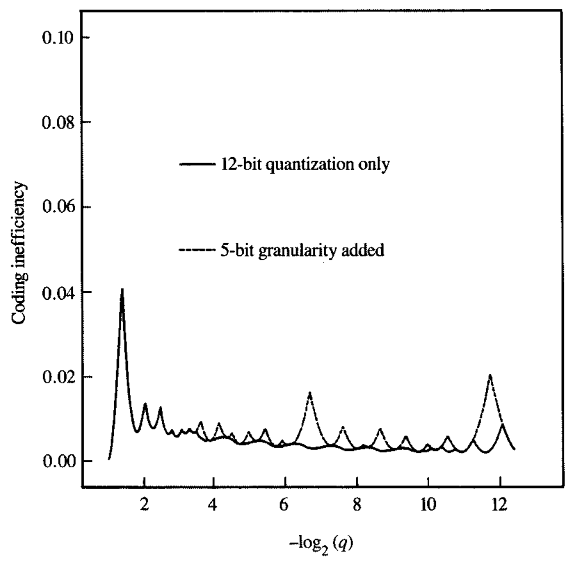


Figure 7

Coding inefficiency as a function of q. The solid curve is for a Q_e table which allows all values between X'AC1' and 1. The Q_e value used for any given q is the smallest value in the table that minimizes the coding inefficiency. The dashed curve is for the 5-bit Q_e table (Table 1), again with the smallest value of Q_e chosen that minimizes the coding inefficiency.

X'1000'. When the system is at that entry, the LPS renorm moves the Q_e pointer such that at least two MPS renorms in succession are needed to return to the smallest entry. This avoids the trap.[3]

The calculations and measurements described in this section show that coding inefficiency is strongly influenced by the Q_e table granularity and the amount of change in index following renormalization. Table granularity and the amount of change directly influence the rate at which the system adapts to a change in q. However, the faster the adaptation rate, the more the distribution in Q_e is spread, and the larger the coding inefficiency is for stationary statistics.

6. Multi-rate probability estimation

In real data sets the symbol probability can vary quite widely. Consequently, the adaptive nature of the probability estimator is extremely important in achieving good coding efficiencies. The two different Q_e table granularities discussed in the preceding section exhibit quite different coding efficiencies for fixed-probability coding. However, the coarser granularity of the 5-bit Q_e table allows the estimator to arrive

[3] G. G. Langdon suggested the particular integer representation (X'1000' corresponding to 0.75) used in developing these tables. His suggestion was motivated by simplicity in hardware implementation, in that this representation allowed a single-bit test to determine when renormalization of the interval was needed. However, this representation also guaranteed that the estimator would be trapped at the smallest integer value, $Q_e = 1$. We had explored conditional renormalization as a means of centering the estimator at the correct Q_e. Langdon suggested using it to avoid the trap.

at the appropriate Q_e value at a cost of about half as many bits in the compressed data string. This illustrates the trade-off that must be made between good coding efficiency for stationary probabilities and rapid estimation of changing probabilities.

A second-order process has been developed, however, which allows use of the 6-bit Q_e table (retaining most of the coding efficiency for stable statistics) and yet provides very fast estimation of changing probabilities. This second-order process is based on measurement of the correlation of renormalizations. If the probability of a given renormalization is about 0.5, the probability of two renormalizations in sequence being the same is also about 0.5. Only if the current estimate of Q_e is poor will the renormalization correlation probability significantly exceed 0.5.

Correlation of renormalizations is detected by comparing the previous renormalization (for a given context) to the current one. A 4-bit counter (R_{cr}), kept individually for each context, is used to determine the degree to which this correlation is occurring. The counter is incremented by one if the renormalization is the same as the last and decremented by two if the renormalization is different. The one exception to this rule is at the minimum value of Q_e. A spurious indication of correlation would result if R_{cr} were incremented in that state. If R_{cr} is zero, the schedule for change in the Q_e index shown in the 6-bit Q_e table is followed. As R_{cr} increases, an additional change in Q_e index is invoked according to the schedule in **Table 3**. Because the change in Q_e index increases with increasing R_{cr}, R_{cr} is a measure of the estimation rate.

This schedule of extra increments and decrements was arrived at experimentally, using the optimized 6-bit Q_e table and adjusting the schedule until the best overall performance was obtained for mixed-context coding. The coding performance of this multi-rate estimator is shown in **Figure 8** for the fixed-probability data sets. Over most of the range of probabilities the degradation in coding efficiency caused by the addition of the multi-rate structure is very small. As is seen in the next section, this is offset by a significant increase in the robustness of the estimation process in the coding of real files.

An approximate model has been developed for this multi-rate structure which helps to explain the structure described above. If the probability of a renormalization being the same as the previous one is characterized by a single average value P_{cr}, the probability that the counter R_{cr} has any particular value can be modeled by the finite-state machine shown in **Figure 9**. Transitions to the right occur with probability P_{cr}, and transitions to the left occur with probability $1 - P_{cr}$. For stable symbol probabilities, P_{cr} should be of the order of 0.5. Since for that case the occupancy probability should be large for small R_{cr}, the decrement of R_{cr} must be larger than the increment. If the increment and decrement were equal, the occupation probability of all the states would be identical.

Table 2 Probability estimates for 6-bit Q_e.

Q_e index	*Q_e value*	*Decimal Q_e value*	*Decr (LPS)*	*Incr (MPS)*	*MPS exch*
0	X'0A81'	0.49237	1	1	1
1	X'0A01'	0.46893	1	1	0
2	X'0981'	0.44550	1	1	0
3	X'0901'	0.42206	1	1	0
4	X'08A1'	0.40448	1	1	0
5	X'07C1'	0.36346	1	1	0
6	X'0761'	0.34589	1	1	0
7	X'0701'	0.32831	1	1	0
8	X'06C1'	0.31659	1	1	0
9	X'0681'	0.30487	1	1	0
10	X'0641'	0.29315	1	1	0
11	X'0601'	0.28143	1	1	0
12	X'0581'	0.25800	1	1	0
13	X'0501'	0.23456	2	1	0
14	X'04C1'	0.22284	1	1	0
15	X'04A1'	0.21698	1	1	0
16	X'0481'	0.21112	2	1	0
17	X'0461'	0.20526	1	1	0
18	X'0441'	0.19940	2	1	0
19	X'0421'	0.19354	2	1	0
20	X'03C1'	0.17596	1	1	0
21	X'0381'	0.16425	1	1	0
22	X'0341'	0.15253	1	1	0
23	X'0301'	0.14081	1	1	0
24	X'02E1'	0.13495	2	1	0
25	X'02C1'	0.12909	1	1	0
26	X'02A1'	0.12323	1	1	0
27	X'0281'	0.11737	2	1	0
28	X'0261'	0.11151	1	1	0
29	X'0241'	0.10565	2	1	0
30	X'0221'	0.09979	2	1	0
31	X'01E1'	0.08807	1	1	0
32	X'01A1'	0.07635	2	1	0
33	X'0181'	0.07050	1	1	0
34	X'0161'	0.06464	2	1	0
35	X'0141'	0.05878	1	1	0
36	X'0131'	0.05585	2	1	0
37	X'0121'	0.05292	2	1	0
38	X'00F1'	0.04413	1	1	0
39	X'00E1'	0.04120	2	1	0
40	X'00C1'	0.03534	1	1	0
41	X'00A1'	0.02948	2	1	0
42	X'0091'	0.02655	2	1	0
43	X'0079'	0.02216	1	1	0
44	X'0071'	0.02069	2	1	0
45	X'0061'	0.01776	1	1	0
46	X'0053'	0.01520	2	1	0
47	X'0049'	0.01337	2	1	0
48	X'0039'	0.01044	1	1	0
49	X'0033'	0.00934	1	1	0
50	X'0025'	0.00677	2	1	0
51	X'0023'	0.00641	2	1	0
52	X'0019'	0.00458	1	1	0
53	X'0013'	0.00348	2	1	0
54	X'0011'	0.00311	2	1	0
55	X'000B'	0.00201	2	1	0
56	X'0009'	0.00165	2	1	0
57	X'0007'	0.00128	2	1	0
58	X'0005'	0.00092	2	1	0
59	X'0003'	0.00055	2	1	0
60	X'0001'	0.00018	2	0	0

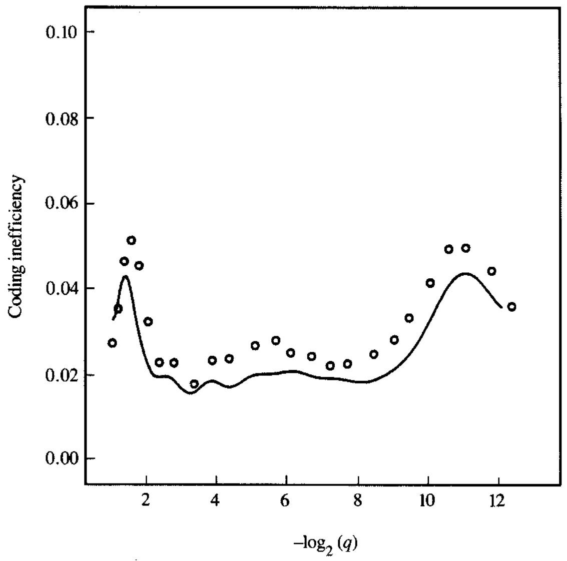


Figure 8

Coding inefficiency with multi-rate probability estimation. The circles are measured for pseudorandom data sets of fixed q. The 6-bit Q_e table (Table 2) is used with this multi-rate system. Consequently, the coding efficiency of the single-rate 6-bit Q_e estimator is included for comparison (solid line).

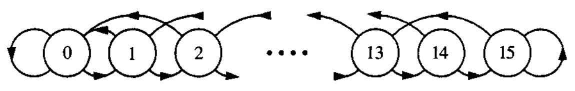


Figure 9

State diagram for the approximate calculation of the distribution of the rate counter R_{cr}.

Table 3 Schedule of extra increments and decrements.

R_{cr}	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Decrement	0	0	1	1	2	2	3	4	5	7	9	11	13	14	15	15
Increment	0	0	0	1	1	1	2	2	3	3	4	5	5	5	5	5

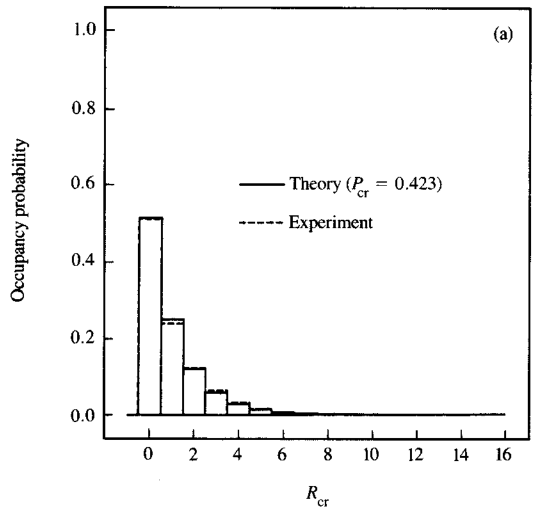
A comparison between the predictions of this model and measured R_{cr} distributions is shown in **Figure 10**. Figure 10(a) shows the R_{cr} distribution when the extra changes in Table 3 are invoked. Figure 10(b) shows the R_{cr} distribution when the table of extra changes is set to zero. The basic model for the correlation in renormalization appears to be fairly good if the renormalization correlation probability P_{cr} is treated as an adjustable parameter.

If the schedule of increments and decrements listed in Table 3 is set to zero, the renormalization correlation probability can be directly calculated using the same concepts discussed in Section 3. **Figure 11** shows a comparison between calculated and measured renormalization correlations. The measurements were done either by direct counting of correlation, or by a best fit (least squares) of calculated to measured R_{cr} distributions. A third set of measurements shows the effect on the renormalization correlation of allowing the extra changes shown in Table 3. The main purpose in doing these calculations and measurements was to determine whether the schedule of extra changes should have a dependence on Q_e. The relatively slight dependence of P_{cr} on q indicates that little can be gained with this additional complexity.

7. Mixed-context coding

Controlled experiments on mixed contexts are more difficult to design. One attempt is shown in **Figures 12** and **13**, in which pseudorandom data sets were coded as if they comprised 32 intermixed contexts, each with the same probability. These two figures also show the prediction of coding efficiency given by the random-interval model. Although agreement is poor for q near 0.5, at smaller q the random-interval model predicts the behavior quite well. The predicted coding efficiency for a single context is also shown for comparison (dashed line). Qualitatively, mixing contexts appears to increase the coding inefficiency by about 1%. However, this experiment is not representative of most mixed-context data sets in that the various contexts usually have quite different Q_e.

The assumption of a uniform interval distribution, the basis for the random-interval model, is not very good, especially near $q = 0.5$. However, there are enough other difficulties in the modeling of mixed contexts that nothing more successful has yet been found. One example of these difficulties is the interaction between contexts with q near 0.5 and contexts with very small q. As noted earlier, near $q = 0.5$ there is a significant chance of getting a 2-bit MPS renorm. However, below $q = 0.375$, this 2-bit MPS renorm cannot occur. The randomizing effect of mixed contexts tends to shift the interval distribution to larger values, thereby reducing the probability of the 2-bit MPS renorm and the bit rate for coding of contexts with q near 0.5. Unfortunately, the shift in the interval distribution can only occur because of MPS renormalization during coding of other contexts.



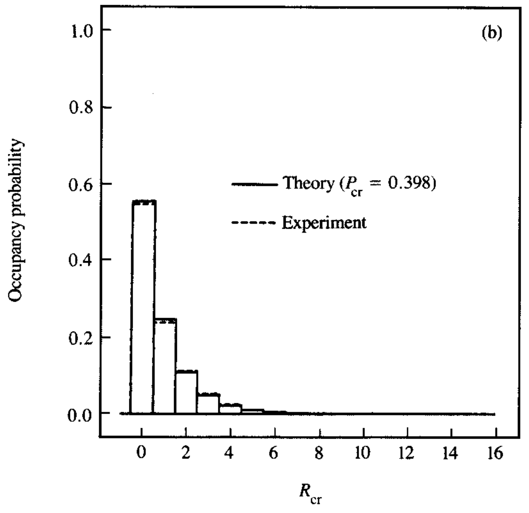


Figure 10

A typical R_{cr} distribution when extra increments and decrements of the Q_e index are (a) allowed; (b) not allowed. q = 0.1.

Consequently, the decrease for high-q contexts is offset by increased bit rate for low-q contexts. This effect was observed experimentally during optimization of the 5-bit Q_e table. Minor changes made to the Q_e table at values near 0.5 caused a significant increase in the bit rate for some contexts with very small Q_e.

Figure 14 shows the effect of multiple contexts for the 12-bit-per-context multi-rate estimator. Again, the coding inefficiency is increased by about 1%. The increase in coding inefficiency is caused primarily by a broadening of the distribution of Q_e rather than by a poor average estimate of the probability.

Multiple-context interval randomization does not strongly influence the transient behavior of the estimation process. **Figures 15–17** show the transient coding inefficiency for the 6-, 7-, and 12-bit-per-context estimators. Decision strings with $q = 0.01$ for the zero symbol were used, and the coder was initialized with Q_e = X′0AC1′ ($Q_e \simeq 0.5$), MPS = 0, and $A(0)$ = X′1000′ (0.75). This is the initialization convention used for the image data set experiments described later in this section. Only the data for $q = 0.01$ are shown, as the behavior for other q values is qualitatively similar. To generate these plots, the results from 25600 different pseudorandom decision strings were averaged for the single-context case; results from 800 strings were averaged for the 32-context case. The comparison of single-context and 32-

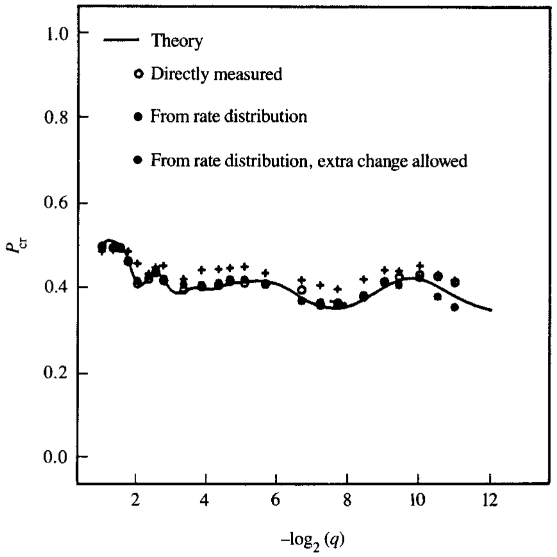


Figure 11

Comparison between the calculated and measured correlated renormalization probabilities.

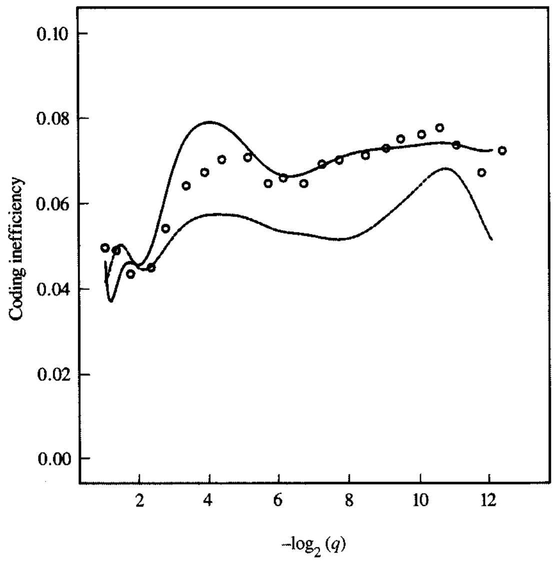


Figure 12

Coding of mixed contexts for the 6-bit-per-context estimator. 32 contexts, all with the same q, were coded in a cyclical fashion. The prediction of the random-interval model is shown with the solid line. For comparison, the single-context exact calculation is shown with a dashed line.

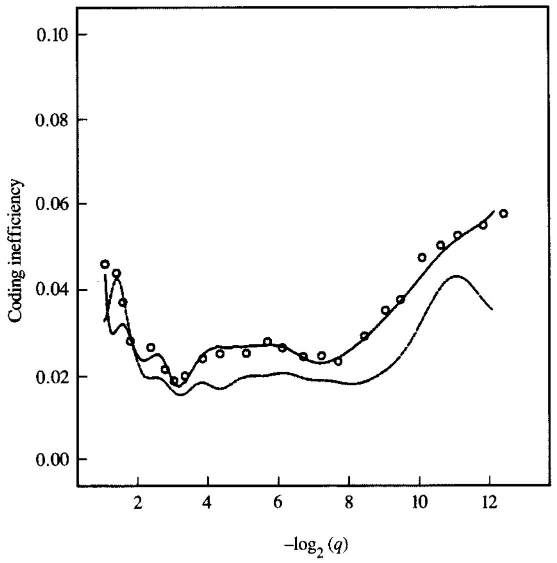


Figure 13

Coding of mixed contexts for the 7-bit per-context estimator. 32 contexts, all with the same q, were coded in a cyclical fashion. The prediction of the random-interval model is shown with the solid line. For comparison, the single-context exact calculation is shown with a dashed line.

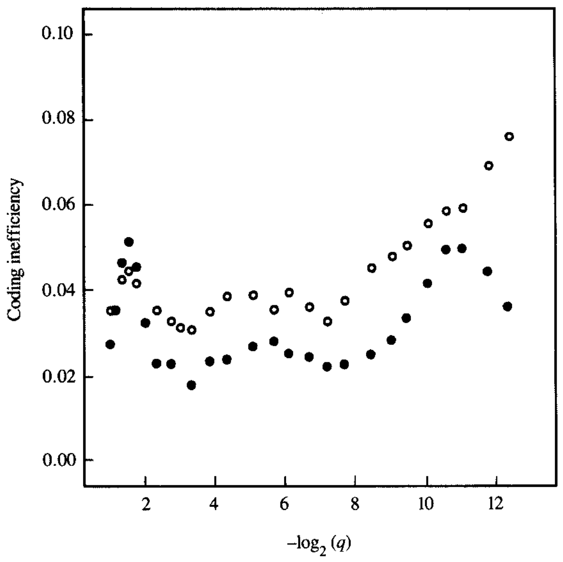


Figure 14

Coding of mixed contexts for the 12-bit-per-context estimator. 32 contexts (○), all with the same q, were coded in a cyclical fashion. For comparison, single-context measurements (●) are also shown.

context data shows that except for the detailed structure seen in the single-context case, the transient behavior is quite similar for single- and mixed-context coding. The detailed structure in the single-context data is real and can be verified approximately by calculating the coding rate for the most likely sequence—an LPS followed by a string of MPSs. The randomizing of the A register by context mixing removes this fine structure.

Figure 18 shows the transient behavior of the estimation-rate counter R_{cr}. In this case, context mixing does affect the behavior, although not strongly. Context mixing tends to broaden the Q_e distribution; this increases the renormalization correlation, which in turn increases the value of R_{cr}.

Most real coding problems involve mixed contexts. Consequently, while initial work was done with pseudorandom fixed-probability data sets, great emphasis was placed on obtaining the best coding efficiencies for two sets of mixed-context files, one generated by a gray-scale compression model [16] and the other by a 7-pel predictor facsimile model [11]. Data from a single file derived from a modified-neighborhood predictor [17] are also included. Results for these files are shown in **Table 4** for four variants of the estimator: the 6-bit-per-context (5-bit Q_e, one bit to define the sense of the MPS), the 7-bit-per-context (6-bit Q_e), the 12-bit-per-context multi-rate estimator, and the 12-bit-

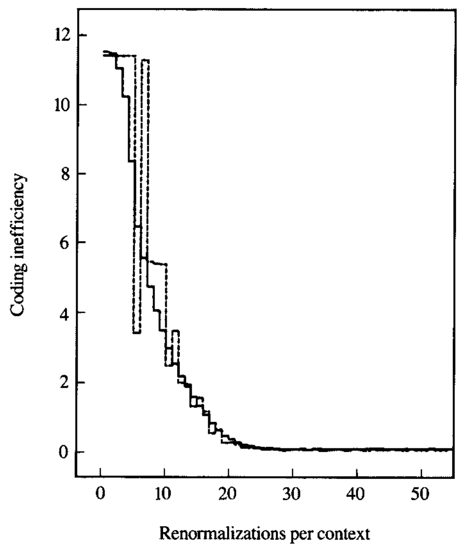


Figure 15

Decay of coding inefficiency as a function of the average renormalization count per context for the 6-bit-per-context estimator. $q = 0.01$; the solid curve represents 32 contexts, the dashed curve a single context.

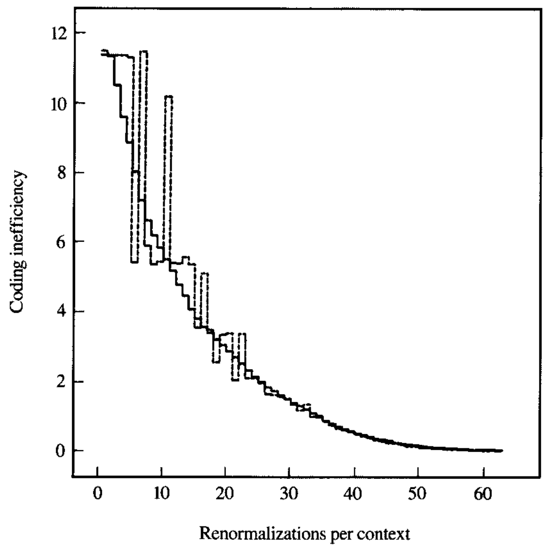


Figure 16

Decay of coding inefficiency as a function of the average renormalization count per context for the 7-bit-per-context estimator. $q = 0.01$; the solid curve represents 32 contexts, the dashed curve a single context.

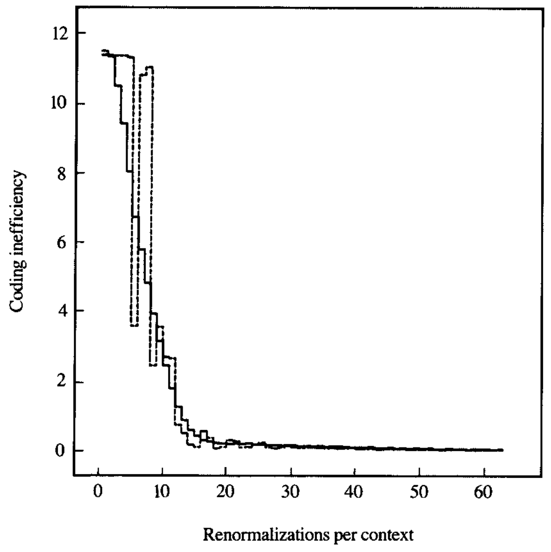


Figure 17

Decay of coding inefficiency as a function of the average renormalization count per context for the 12-bit-per-context estimator. $q = 0.01$; the solid curve represents 32 contexts, the dashed curve a single context.

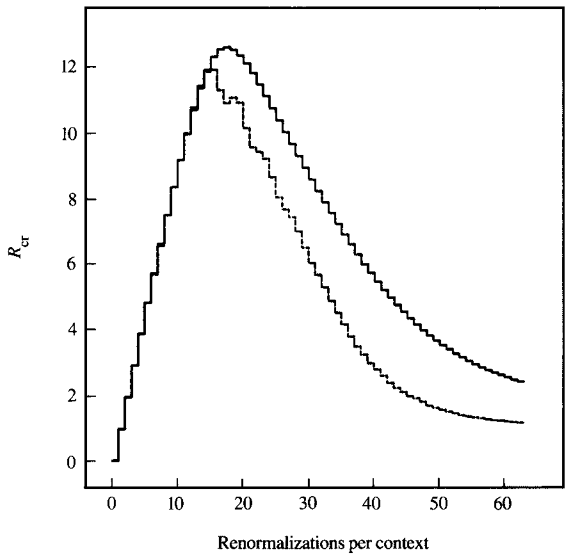


Figure 18

Variation of the average rate counter, R_{cr}, as a function of the average renormalization count per context. $q = 0.01$; the solid curve represents 32 contexts, the dashed curve a single context.

Table 4 Q-Coder coding performance.

	Present experiments				*Reference*		*Entropy*
Bits per context	*6 bits*	*7 bits*	*12 bits*	*24 bits*			
Q_e	*5 bits*	*6 bits*	*6 bits*	*6 bits*			
(file)	(bits)	(bits)	(bits)	(bits)	(bits)		(bits)
Gray-scale, teleconferencing model							
t2	171 856	171 296	170 968	170 544	172 608	[16]	166 373
courierf	94 432	94 720	93 608	93 376	98 560	[16]	95 327
courier	171 648	171 264	170 488	170 616	173 536	[16]	170 335
ieeef	175 536	175 856	174 704	174 272	176 560	[16]	173 443
ieee	232 952	233 040	232 104	231 792	232 704	[16]	228 819
handwrtf	98 832	98 664	98 240	98 336	100 224	[16]	97 563
topletf	101 784	102 008	101 088	101 040	104 448	[16]	101 272
densetf	248 232	249 016	247 328	246 928	253 792	[16]	245 042
marcosf	110 536	110 440	109 240	108 800	110 512	[16]	108 278
atomsf	171 408	170 392	170 720	170 504	170 288	[16]	164 753
fruitf	141 248	140 344	140 328	140 296	141 408	[16]	137 530
t4f	237 560	236 704	236 256	236 008	240 960	[16]	229 059
t5f	164 112	162 120	161 752	161 784	163 264	[16]	156 253
Total:	2 120 136	2 115 864	2 106 824	2 104 296	2 138 864		2 074 047
Facsimile, 7-pel predictor							
ccitt1	119 752	123 632	119 816	119 000	122 021	[11]	130 623
ccitt2	71 568	72 232	71 400	70 696	71 932	[11]	71 884
ccitt3	187 728	192 224	187 400	185 952			200 927
ccitt4	446 816	461 200	447 304	444 184			494 249
ccitt5	215 888	220 176	215 960	214 120			230 345
ccitt6	112 256	113 992	111 808	110 544			117 002
ccitt7	468 232	465 840	466 304	462 840	466 907	[11]	465 156
ccitt8	124 768	125 280	123 840	123 056			131 735
Total:	1 747 008	1 774 576	1 743 832	1 730 392			1 841 921
Digital halftone, 7-pel predictor							
budking	966 480	1 051 592	885 368	879 176	101 1952	[11]	1 267 429
boat2x	154 744	155 464	149 656	147 680	152 258	[11]	153 029
jphmesh	184 016	213 680	146 136	144 192			257 665
Total:	1 305 240	1 420 736	1 181 160	1 171 048			1 678 123
Digital halftone, modified predictor model [17]							
jphmesh5	91 088	90 312	90 584	89 792			93 526

per-context multi-rate estimator when used with an independent "interval register" for the estimation process for each context (24 bits per context). Using an independent interval register per context for the estimation process removes the effect of context mixing from the coding efficiency.

When the 6- and 7-bit-per-context single-rate estimator results were compared, the 6-bit-per-context estimator worked significantly better for the facsimile files and the digital halftone files analyzed using the 7-pel predictor. However, the gray-scale files and the file from the modified predictor model compressed slightly better with the 7-bit-per-context estimator.

Still better results were obtained with the 12-bit-per-context multi-rate estimator, but the best results were obtained with the 24-bit-per-context estimator. Unfortunately, relative to the 12-bit-per-context estimator, a substantial penalty is incurred in achieving slightly improved compression in terms of context storage and computational complexity. The compression improvement between 24-bit-per-context and 12-bit-per-context estimators ranged from approximately −0.2% to 1.5%, with the largest improvement occurring for the facsimile and digital halftone files. Since the gray-scale files had an average probability per decision of about 0.25, and the facsimile and digital halftone files had an average probability per decision an order of magnitude smaller, this is somewhat consistent with the data in Figure 14. The 12-bit-per-context multi-rate estimator gave the second best overall performance, but for facsimile files the performance of the 6-bit-per-context version was essentially as good.

The results for the digital halftone files are particularly interesting, for these files require an estimator which can track rapidly changing probabilities. This is evident both from relative performances achieved with the three different estimators, and from comparison of the results with the entropies. The robustness of the multi-rate estimator is particularly apparent with these files. Note that the numbers quoted for the Q-Coder and in [8] are for fully decodable files. They include the overhead for bit stuffing to block carry propagation, flushing of the final bits in the code register, and rounding to a byte boundary.

8. Summary

Theoretical modeling and experimentation have been used to show that estimation of symbol probabilities solely from interval-register renormalization is a practical and robust estimation technique. The technique is readily implemented in either hardware or software. The computations involved are almost trivial, and scale with the number of compressed bits generated.

The single-rate estimator is particularly simple, and requires minimal storage for the Q_e table and contexts. Its simplicity—in both hardware and software—makes it the estimation technique of choice for the Q-Coder. While the multi-rate estimator does give significantly better performance under some conditions, it requires either substantially more computations per estimation or much larger finite-state machine tables. This additional complexity is a deterrent to the use of the multi-rate estimator in current-generation hardware and software environments.

The coding efficiency attained with these estimation techniques is clearly influenced by the nature of the data sets chosen for the optimization. However, given that choice, the coding efficiencies attained for both single- and multi-rate variations compare favorably with results from any technique known. Although complexity is currently a barrier to its use, the multi-rate version has an additional robustness that guarantees excellent coding efficiency for the entire range of models considered.

Adaptive arithmetic coding often gives lower coding rates than predicted from the stationary entropy, and thus raises an interesting question. What is the lower bound for the coding rate for statistically unstable data? One promising approach to a theoretical bound is J. Rissanen's MDL (minimum descriptor length) principle [18]. J. Rissanen and K. Mohiuddin [10, 14] used that principle as the basis for their probability-estimation technique. However, on the basis of very limited comparisons, the Q-Coder outperforms that estimator. The true theoretical bound for the coding of nonstationary systems is not known.

Acknowledgments

As noted in the Introduction, this paper is part of a collaborative effort between IBM researchers at the IBM Almaden Research Center in San Jose, California, and the IBM T. J. Watson Research Center in Yorktown Heights, New York, to develop an arithmetic coding implementation which is computationally efficient in both software and hardware. We have benefited greatly from the continuing interaction with J. Rissanen and R. Arps of the Almaden facility, and G. Langdon, formerly at Almaden and now retired from IBM. We have also benefited from interactions with C. Gonzales and G. Goertzel at the Yorktown facility, and from the continuing support and encouragement of our manager, K. Pennington.

References

1. J. J. Rissanen, "Generalized Kraft Inequality and Arithmetic Coding," *IBM J. Res. Develop.* **20**, 198 (1976).
2. R. C. Pasco, "Source Coding Algorithms for Fast Data Compression," Ph.D. Thesis, Department of Electrical Engineering, Stanford University, Stanford, CA, 1976.
3. G. G. Langdon and J. J. Rissanen, "A Simple General Binary Source Code," *IEEE Trans. Info. Theory* **IT-28**, 800 (1982).
4. Glen G. Langdon, Jr., "An Introduction to Arithmetic Coding," *IBM J. Res. Develop.* **28**, 135 (1984).
5. W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, Jr., and R. B. Arps, "An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder," *IBM J. Res. Develop.* **32**, 717 (1988, this issue).
6. J. L. Mitchell and W. B. Pennebaker, "Software Implementations of the Q-Coder," *IBM J. Res. Develop.* **32**, 753 (1988, this issue).
7. J. L. Mitchell and W. B. Pennebaker, "Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder," *IBM J. Res. Develop.* **32**, 727 (1988, this issue).
8. R. B. Arps, T. K. Truong, D. J. Lu, R. C. Pasco, and T. D. Friedman, "A Multi-Purpose VLSI Chip for Adaptive Data Compression of Bilevel Images," *IBM J. Res. Develop.* **32**, 775 (1988, this issue).
9. J. L. Mitchell, W. B. Pennebaker, C. A. Gonzales, and C. F. Touchton, "A Predictive Image Coder/Decoder Using Adaptive Binary Arithmetic Coding," *Research Report RC-13423*, IBM T. J. Watson Research Center, Yorktown Heights, NY, January 1988.
10. J. J. Rissanen and K. M. Mohiuddin, "Multiplication-Free Multi-Alphabet Arithmetic Coder," *IEEE Trans. Commun.*, to be published.
11. G. G. Langdon and J. J. Rissanen, "Compression of Black-White Images with Arithmetic Coding," *IEEE Trans. Commun.* **COM-29**, 858 (1981).
12. W. B. Pennebaker and J. L. Mitchell, "Adaptive Arithmetic Coding in the Logarithm Domain," unpublished work; W. B. Pennebaker, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598.
13. G. Goertzel and J. L. Mitchell, "Symmetrical Optimized Adaptive Data Compression/Transfer/Decompression System," U.S. Patent 4,633,490, December 30, 1986.
14. K. Mohiuddin, J. J. Rissanen, and M. Wax, "Adaptive Model for Nonstationary Sources," *IBM Tech. Disclosure Bull.* **28**, 4798 (1986).
15. D. R. Helman, G. G. Langdon, N. Martin, and S. J. P. Todd, "Statistics Collection for Compression Coding with Randomizing Feature," *IBM Tech. Disclosure Bull.* **24**, 4917 (1982).
16. D. Anastassiou, J. L. Mitchell, and W. B. Pennebaker, "Gray-Scale Image Coding for Freeze-Frame Videoconferencing," *IEEE Trans. Commun.* **COM-34**, 382 (1986).
17. K. Toyokawa, "New Arithmetic Coding Technique for Dither Halftone Image," presented at the Annual Meeting of the Information Society of Japan, Sapporo, September 28–30, 1987.
18. J. J. Rissanen, "Stochastic Complexity and Modeling," *Ann. Statist.* **14**, 1080 (1986).

Received August 11, 1987; accepted for publication August 17, 1988

William B. Pennebaker *IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Pennebaker is a Research Staff Member at the IBM T. J. Watson Research Center and currently manages a group doing research in areas related to image processing and compression. He joined IBM's Research Division in 1962 and has worked in areas related to low-temperature physics, thin films, display technology, printing technology, and image processing. Dr. Pennebaker has received an Outstanding Contribution Award for work on strontium titanate films, an Outstanding Invention Award for work on silicon nitride films, and an Outstanding Innovation Award for work on image processing and compression. He has received ten IBM Invention Achievement Awards. Dr. Pennebaker received his B.S. in engineering physics from Lehigh University, Bethlehem, Pennsylvania, in 1957, and his Ph.D. in physics from Rutgers University, New Brunswick, New Jersey, in 1962. He is a member of the American Association for the Advancement of Science, the American Institute of Physics, the Institute of Electrical and Electronics Engineers, and the Society for Information Display.

Joan L. Mitchell *IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown heights, New York 10598.* Dr. Mitchell graduated from Stanford University with a B.S. in physics in 1969. She received her M.S. and Ph.D. degrees in physics from the University of Illinois at Champaign-Urbana in 1971 and 1974, respectively. She joined the Exploratory Printing Technologies group at the IBM T. J. Watson Research Center immediately after completing her Ph.D., and since 1976 has worked in the field of data compression. Dr. Mitchell received IBM Outstanding Innovation Awards for two-dimensional data compression in 1978, for teleconferencing in 1982, and for the Image View Facility and resistive ribbon thermal transfer printing technology in 1985. She is co-inventor on fifteen patents.