# 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 $0.75 \le A < 1.5$, $A$ 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.

## 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 $q$, $\Delta A$ is the change in the interval ($0.75 > \Delta A \ge 0$), 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 $\Delta A$ is dependent on the type of renormalization. If the MPS renorm occurred last, $\Delta A$ is close to 0.75; if the LPS renorm occurred last, $\Delta A$ is typically somewhat smaller than 0.75. If the effective value of $\Delta A$ 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

[1] In unpublished work G. Goertzel and J. L. Mitchell explored and abandoned this possibility because they were unable to obtain good coding efficiencies.

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 $k$th allowed value of $Q_e$), balance of transition probabilities into and out of the $k$th 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 $k \geq k_{ex}$ and $X_k = 1 - q$ for $k < k_{ex}$. 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 $k_{ex} - 1$. 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 $k < k_{ex}$ 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 $k$th 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

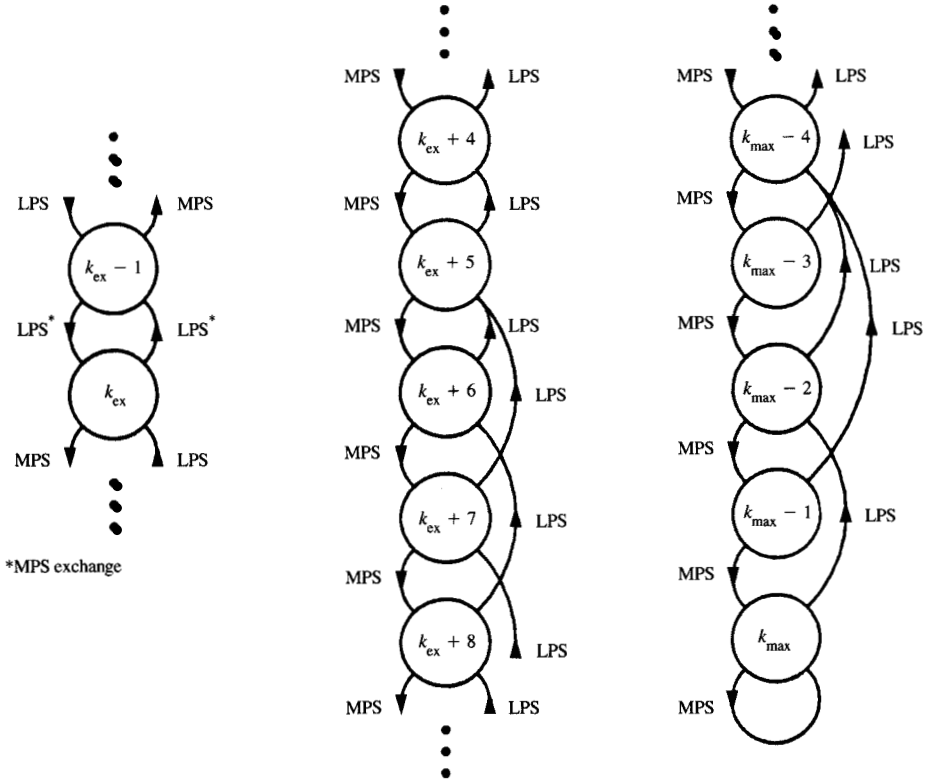


**Figure 1**

Sections of the state diagram for the probability estimator.

sequence continues until the interval decays below 0.75 for the $k$th state. The exponent $r_{kj}$ is the number of MPSs needed to just leave the $k$th 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 $k$th state is thus

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

where $j$ ranges over all MPS renormalizations which can occur following the LPS renorm. $B_{\mathrm{LPS},k}$ is the number of bits generated in renormalizing $Q_e[k]$ to the allowed interval range. $B_{\mathrm{MPS},j}$ is the number of bits generated by the $j$th 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 $j$th MPS renorm after the LPS event from state $k$.

The total coding rate in bits per symbol is therefore

$$R = \sum_k R_k . \tag{9}$$

## 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 $P_{l,k}$ and the MPS renormalization probability $P_{m,k}$:

$$P_{l,k} = q, \tag{10}$$

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

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 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

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

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

$$E = \frac{\sum_k n_k R_k - H}{H}, \tag{13}$$

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

$$H = -q \log_2(q) - p \log_2(p), \tag{14}$$

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

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

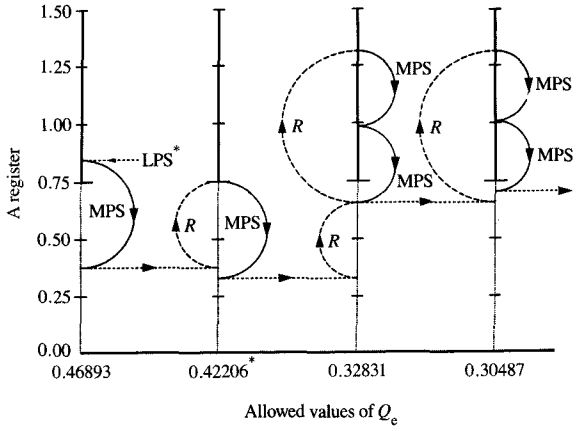


Figure 2

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

DOCKET ALARM

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.