Segmented Log Functions:

Application to Quantizers and Geometrically-Spaced Sequences

Peter Kabal

Department of Electrical & Computer Engineering

McGill University

v2 2026-03

http://WWW-MMSP.ECE.McGill.CA/MMSP

Revision History:

2019-10-03	v1	Initial version
2026-03-25	v2	Corrections, alias table discussion extended

Segmented Log Functions:
Application to Quantizers and Geometrically-Spaced Sequences

Abstract

This report examines log-based compressive functions. A generalized A-law function which subsumes both μ-law and A-law functions is formulated. Segmented versions of these functions are used to form non-uniformly-spaced quantizers. The use of the segmented compressive functions to form geometrically-spaced sequences is described. Table-driven approaches to implement the search for the quantizer interval corresponding to an input value are described. These apply to non-uniform quantization intervals. A new method motivated by Walker's alias method can reduce the number of comparisons needed to isolate a quantizer interval.

1 Introduction

Log-based functions are used as compressor-expander (compander) functions for creating quantizers with non-uniform step sizes. This allows for a quantizer to have small step sizes for small input levels and large step sizes for large input levels. The μ-law and A-law functions are commonly cited variants. Segmented approximations to these functions have been standardized for use in voice communications.

In this report, the μ-law function is used to create a geometrically-spaced sequence. This sequence is also the basis for defining a segmented approximation to the μ-law function which is used as the basis for the ITU-T G.711 standard-compliant μ-law quantizer.

The A-law function is another log-based compander function. A generalized A-law function is developed which subsumes both the μ-law and standard A-law functions. The segmented version of the A-law function requires a further small modification to match the ITU-T G.711 standard-compliant A-law quantizer.

An appendix describes quantization procedures which are tailored to the G.711 quantizers. In another appendix, table-driven quantization procedures are described. A new quantization scheme uses an auxiliary table to reduce the search complexity for finding a quantizer region.

2 Log Function

The log function goes from $-\infty$ at $x = 0$ to zero at $x = 1$, and then becomes positive for $x > 1$ as shown in Fig. 1. Also shown in the figure are samples of the log function with equal-size ordinate-increments. The sample points result in x-increments that have geometric spacing.

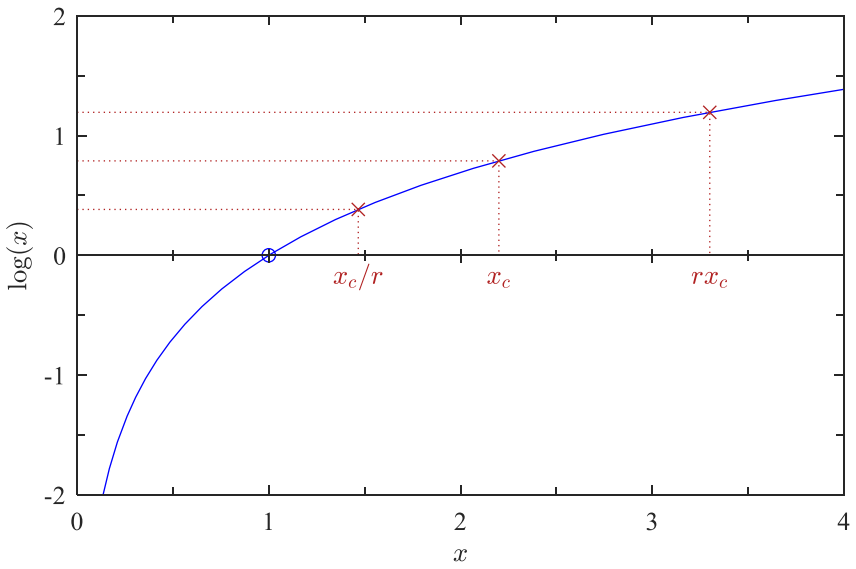

Fig. 1 The log function.

The geometrically-spaced x-increment property will be shown for a scaled and shifted version of the log function,

$$y = c\log(b + ax). \tag{1}$$

The inverse-log function is

$$x = \frac{\exp(y/c) - b}{a}. \tag{2}$$

Sample the log function at equally-spaced points on the y-axis,

$$y_{n+1} - y_n = \Delta_y. \tag{3}$$

The corresponding x-increment is

$$x_{n+1} - x_n = \frac{\exp\left(\frac{y_n}{c}\right)}{a}(r - 1), \tag{4}$$

where

$$r = \exp(\Delta_y/c). \tag{5}$$

The line joining $[x_n, y_n]$ to $[x_{n+1}, y_{n+1}]$ is a chord to the log function. The x-increment for the previous chord can be written as

$$x_n - x_{n-1} = \frac{\exp\left(\frac{y_n}{c}\right)}{a}(1 - \frac{1}{r}). \tag{6}$$

The ratio of the adjacent x-increments is

$$\frac{x_{n+1} - x_n}{x_n - x_{n-1}} = r. \tag{7}$$

The chords with equal-sized y-increments have x-increments which increase geometrically.

3 μ-Law Function

The μ-law function (see Panter and Dite [1], Smith [2]) is the shifted and scaled log function with the constants c and b set to create a normalized curve with both x and y in the range $[0, 1]$. Following convention, the parameter a is renamed to μ. Let $F(1) = 1$. Then $c = 1/\log(b + \mu)$. The log function is shifted so that the resulting function evaluates to zero when $x = 0$. This requirement sets $b = 1$. Using these values, the μ-law function is

$$F_\mu(x, \mu) = \frac{\log(1 + \mu x)}{\log(1 + \mu)}, \quad 0 \le x \le 1. \tag{8}$$

The inverse mapping is

$$F_\mu^{-1}(y, \mu) = \frac{(1 + \mu)^y - 1}{\mu}, \quad 0 \le y \le 1. \tag{9}$$

The slope of the μ-law function is

$$S_\mu(x) = \frac{\mu}{\log(1 + \mu)} \frac{1}{(1 + \mu x)}, \quad 0 \le x \le 1. \tag{10}$$

The ratio of the slope at $x = 0$ to the slope at $x = 1$ is

$$\begin{aligned} R_\mu &= \frac{S_\mu(0)}{S_\mu(1)} \\ &= 1 + \mu. \end{aligned} \tag{11}$$

Normally the parameter μ is taken to be positive. However, from the previous equation we can see that if R_μ is less than unity, then μ will take on values between -1 and zero. From Eq. (8), when μ is positive, increasing x moves up the log curve (decreasing slope), while when μ is negative, increasing x moves down the log curve (increasing slope). The μ-law function exhibits an x-y flipped symmetry,

$$F_\mu(x, \mu) = 1 - F_\mu\left(1 - x, -\frac{\mu}{1 + \mu}\right). \tag{12}$$

The μ-law function is plotted in Fig. 2. The paired top and bottom lines illustrate the x-y flipped symmetry associated with R_μ and $1/R_\mu$.

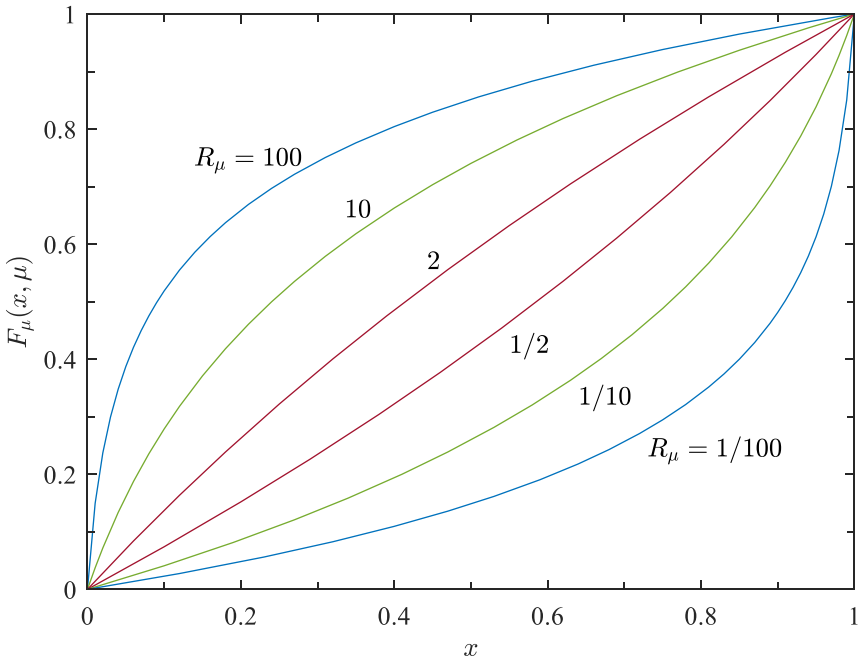

Fig. 2 The μ-law function for different values of $R_\mu = 1 + \mu$.

3.1 Segmented μ-Law Function

Create M uniform length segments for y in the interval $[0, 1]$,

$$\boldsymbol{y}^{(s)} = \left\{0, \frac{1}{M}, \frac{2}{M} \ldots, \frac{M-1}{M}, 1\right\}. \tag{13}$$

For the μ-law function, using $c = 1/\log(1+\mu)$ and $\Delta_y = 1/M$ in Eq. (5),

$$r^M = 1 + \mu. \tag{14}$$

For M segments, the abscissa values define intervals of lengths $L_i = Dr^i$, for $0 \le i \le M-1$,

$$\boldsymbol{L_x} = \left\{D, rD, r^2D, \ldots, r^{(M-1)}D\right\}. \tag{15}$$

Fitting these M segment lengths into the interval [0,1], gives

$$D = \frac{r-1}{r^M - 1}. \tag{16}$$

Setting $y_k^{(s)} = k/M$ in Eq. (8), the geometrically-spaced abscissa values on the μ-law function are

$$\boldsymbol{x}^{(s)} = \left\{0, \frac{r-1}{r^M-1}, \frac{r^2-1}{r^M-1}, \ldots, \frac{r^{M-1}-1}{r^M-1}, 1\right\}. \tag{17}$$

An $M = 4$ segment approximation to the μ-law curve is shown in Fig. 3 for $r = \{4, 2, 1/2, 1/4\}$ alongside the μ-law curves for $R_\mu = \{256, 64, 1/64, 1/256\}$ corresponding to $\mu = \{255,\ 63, -63/64, -255/256\}$.

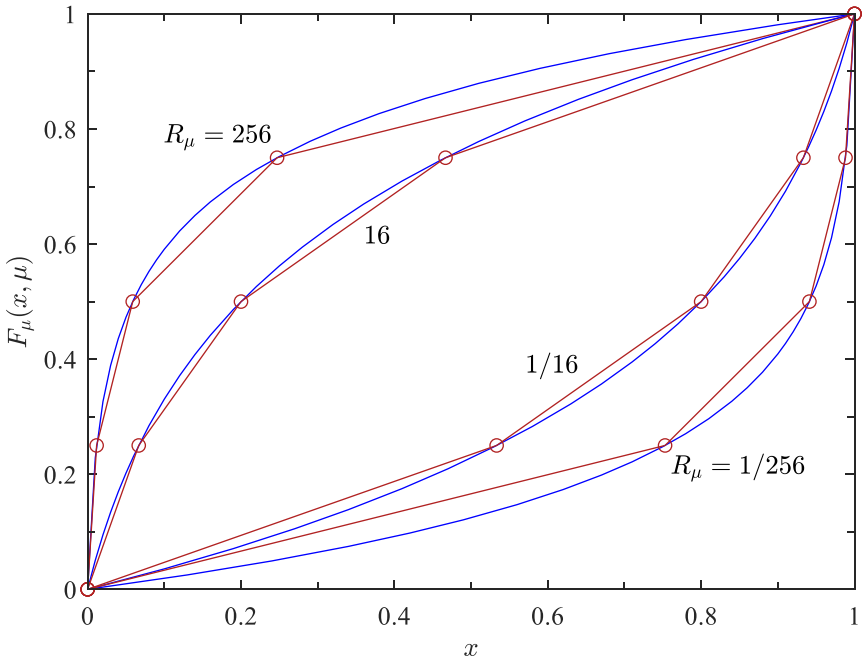

Fig. 3 The μ-law function for different values of $R_\mu = \mu + 1$. Four segment approximations are shown for $r = \{4, 2, 1/2, 1/4\}$.

3.1.1 Application to Geometrically-Spaced Sequences

The segmented μ-law function has application in creating an N-point sequence $\boldsymbol{x}^{(s)}$ with $M = N - 1$ geometrically-spaced intervals as in Eq. (15). Start with an N-point uniformly-spaced $\boldsymbol{y}^{(s)}$ covering $[0, 1]$ as in Eq. (13). Let the geometric spacing be determined by the ratio of the lengths of the last and first intervals,

$$R_s = r^{N-2}. \tag{18}$$

Using Eq. (14), the value of μ is

$$\begin{aligned} \mu &= r^{N-1} - 1 \\ &= \left(\sqrt[N-2]{R_s}\right)^{N-1} - 1. \end{aligned} \tag{19}$$

Since the segment spacing on the x-axis is inversely related to the slopes of the segments, we can make the identification[1],

$$R_\mu = \left(\sqrt[N-2]{R_s}\right)^{N-1}. \tag{20}$$

Then $\boldsymbol{x}^{(s)}$ can be calculated from the inverse μ-law function Eq. (9),

$$\begin{aligned} x_k^{(s)} &= F_\mu^{-1}(y_k^{(s)}, \mu) \\ &= \frac{(1+\mu)^{y_k^{(s)}} - 1}{\mu}, \\ &= \frac{R_\mu^{y_k^{(s)}} - 1}{R_\mu - 1}. \end{aligned} \qquad 0 \le k \le N-1. \tag{21}$$

Computationally, this calculation can be streamlined by first calculating an auxiliary set of values,

$$\begin{aligned} v_k &= R_\mu^{y_k^{(s)}}, \\ &= \exp\left(\log\left(\frac{R_s k}{N-2}\right)\right), \end{aligned} \qquad 0 \le k \le N-1. \tag{22}$$

Note that $v_{N-1} = R_\mu$, Then the values $x_k^{(s)}$ can be written as

$$x_k^{(s)} = \frac{v_k - 1}{v_{N-1} - 1}, \qquad 0 \le k \le N-1. \tag{23}$$

The N values $x_k^{(s)}$ span $[0, 1]$, but can be scaled and shifted to span other intervals.

Figure 4 shows a Matlab script to generate geometrically-spaced values. This routine works for both $R_s < 1$ and $R_s \ge 1$. Geometrically-spaced values will appear uniformly-spaced on a log axis.

3.2 Application to Quantizers

A quantizer maps a continuous-valued input x to an index representing one of N_q partitions of the x-values. Each index selects one of N_q possible output levels. A stepped quantizer characteristic with non-uniform step sizes is shown in Fig. 5.

[1] R_μ is defined as the ratio of the slope at $x = 0$ to that at $x = 1$, wheras R_s is defined as the ratio of the lengths of the *last* to *first* x-axis intervals. Since the y-axis intervals are of equal length, the slope is the inverse of the length of the x-axis intervals. For large N, R_s approaches R_μ.

```
function y = GeoSpace(ymin, ymax, Rs, N)
% GeoSpace Generate geometrically-spaced values
% y = GeoSpace(ymin, ymax, Rs, N)
%
% y    <-  Output geometrically-spaced values
% ymin  -> Lower limit of output values
% ymax  -> Upper limit of output values
% Rs    -> Ratio of last and first increments
% N     -> Number of output values, defaults to 100
%
% This function generates geometrically-spaced values between ymin and ymax
% (inclusive). The parameter Rs gives the ratio of the last and first
% increments,
%
%   Rs = [y(N) - y(N-1)] /  [y(2) - y(1)].
%
% The spacing between output values increases / decreases geometrically by
% the (N-2)'th root of Rs at each value.

if (nargin() == 3)
  N = 100;
else
  N = floor(double(N));
end

if (N == 1)
  y = ymax;
elseif (N == 2)
  y = [ymin, ymax];
elseif (Rs == 1)
  y = linspace(ymin, ymax, N);
else
  vk = exp(log(Rs) * ((0:N-1) / (N-2)));
  yn = (vk - 1) / (vk(N) - 1);
  y = yn * ymax + (1 - yn) * ymin;
end

end
```

Fig. 4 Matlab code for generating geometrically-spaced values.

The quantizer is specified by a set of $N_q - 1$ ordered decision levels $x_q[n]$ which delimit the N_q quantizer intervals. The $N_q - 1$ finite-valued decision levels are $\{x_q[0], \ldots, x_q[N_q - 2]\}$. These decision levels can be supplemented with the virtual decision levels $x_q[-1] = -\infty$ and $x_q[N_q - 1] = \infty$. The index of the interval in which a value x lies is determined as

$$I_q = n \text{ if } x_q[n-1] \leq x < x_q[n], \quad 0 \leq n \leq N_q - 1. \tag{24}$$

In this equation, a value falling exactly on the lower limit of an interval is included in the interval. This is one choice of how to handle a value on the boundary of an interval. The quantized levels are determined by assigning a level $y_q[n]$ to each quantizer interval,

$$x \underset{x_q(\cdot)}{\Longrightarrow} I_q \Rightarrow y_q[I_q] \tag{25}$$

Normally $y_q[I_q]$ is chosen to be inside quantizer interval I_q.

The μ-law characteristic will be used to provide geometrically-spaced segments. The μ-law characteristic described earlier is one-sided – it can be made two-sided by applying the quantizer to the magnitude of the input signal. The quantizer shown in Fig. 5 was formed by creating equally spaced values from -1 to $+1$ in steps of $1/N_q$. These were mapped to decision levels using inverse μ-law function with $\mu = 10$. The quantizer output levels are placed midway between the decision levels. Since the number of output levels is odd, the origin of the graph occurs mid-tread in the staircase function.

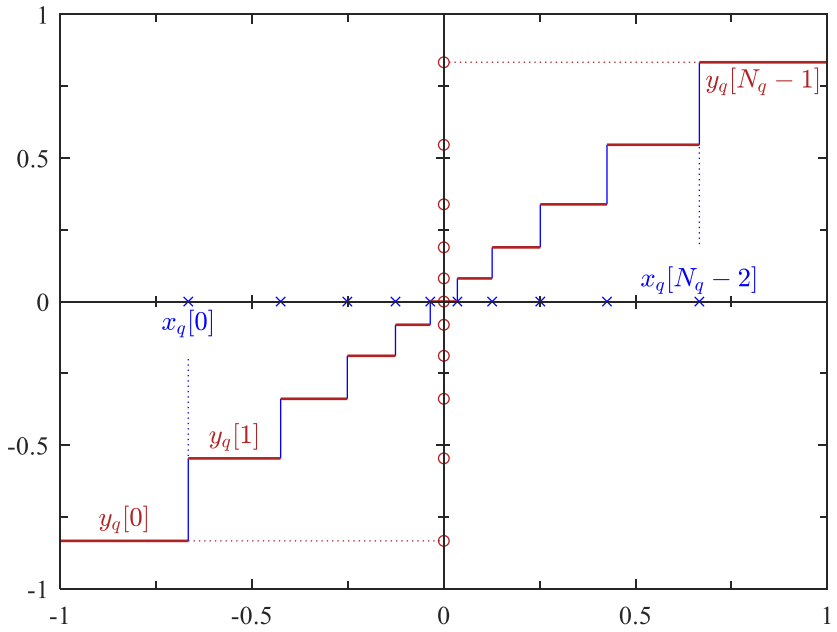

Fig. 5 A non-uniform step size quantizer ($N = 11$) designed using a continuous μ-law function, $\mu = 10$.

3.2.1 ITU-T Recommendation G.711 μ-law

The International Telecommunications Union Recommendation G.711 [3] uses a segmented μ-law characteristic to generate a quantizer with 256 output levels. Each polarity is implemented with $M = 8$ segments. Within a segment the quantizer characteristic has 16 uniformly-spaced output levels. Segment-to-segment, the step sizes increase by the factor $r = 2$, giving $\mu = 255$ (Eq. (14)). The index can be represented with 8-bits (one for the sign, 3 for the segment, and 4 for the level within a segment).

First consider a positive-valued x. Form the $M = 8$ geometrically-spaced segment boundaries for $\mu = 255$ using a uniformly-spaced $\boldsymbol{y}^{(s)}$ transformed to the geometrically-spaced $\boldsymbol{x}^{(s)}$. The $\boldsymbol{x}^{(s)}$ values are the quantizer decision levels at the segment boundaries. The decision levels between the $\boldsymbol{x}^{(s)}$ segment boundaries are uniformly spaced, defining 16 quantizer intervals between the segment boundaries. The total number of intervals is $N_q = 16M$. The $N_q + 1$ decision levels are the $x_q[n]$ values ($0 \le n \le N_q$). The last decision levels $x_q[N_q]$ is used only to define the last quantizer output level.[2] The quantizer output values are the mid-points of the intervals defined by the decision levels,

$$y_q[n] = \frac{x_q[n+1] + x_q[n]}{2}, \qquad 0 \le n \le N_q - 1. \tag{26}$$

Scaled values in G.711 μ-law

By convention in G.711, the decision levels and the output levels are scaled such they take on integer values. Set the smallest step size to be 2, with the end of the first segment being 2×16. From Eq. (17), the corresponding normalized decision level at the end of the first segment is $1/(r^M - 1) = 1/255$ with $r = 2$ and $M = 8$. This gives the scaling factor $32 \times 255 = 8160$ used in the G.711 specification.

To fit the scaled output levels for positive and negative values together, the scaled output levels are shifted down by 1 to put the first positive output level at zero. The positive decision levels are similarly moved down by one, and the lowest decision level is readjusted to 0 (instead of -1). After scaling and shifting, the decision levels at the positive segment boundaries are

[2] This virtual decision level delimits the "overload" region. Any inputs larger than the virtual decision level will result in an error larger than a half of the largest quantizer step size.

$$x_q[16k] = \begin{cases} 0, & k = 0, \\ 32\left(2^k - 1\right) - 1, & 1 \le k \le M. \end{cases} \tag{27}$$

The two-sided quantizer with 255 ordered decision levels and 256 output levels is formed from the one-sided quantizer as follows,

$$\begin{aligned} \boldsymbol{y}'_q &= \{-y_q[N_q - 1], \dots, -y_q[0], y_q[0], \dots, y_q[N_q - 1]\}, \\ \boldsymbol{x}'_q &= \{-x_q[N_q - 1], \dots, -x_q[1], x_q[0], x_q[1], \dots, x_q[N_q - 1]\}. \end{aligned} \tag{28}$$

The resulting quantizer has two identical output levels at zero, one for $-1 \le x < 0$, and the other for $0 \le x < 1$. Note also that $x_q[0] = 0$ and $y_q[0] = 0$. Both $\boldsymbol{y}'_q$ ($2N_q$ values) and $\boldsymbol{x}'_q$ ($2N_q - 1$ values) are anti-symmetric. The step sizes for positive inputs are $\{2, 4, \dots, 128, 256\}$; the largest output level is 8031; and the overload point is 8159 (calculated as $8031 + 256/2$).

Scaled values for 16-bit inputs

For 16-bit integer input values (values in $[-32{,}768, +32{,}767]$), the decision levels and output levels defined by G.711 will be scaled up by a factor of 4. The segment boundaries (cf. Eq. (27)) become

$$x_q[16k] = \begin{cases} 0, & k = 0, \\ 128\left(2^k - 1\right) - 4, & 1 \le k \le M. \end{cases} \tag{29}$$

Using this scaling, the steps sizes are $\{16, 32, \dots, 512, 1024\}$; the largest output level is 32,124; and the overload point is 32,636 (calculated as $32{,}124 + 1024/2$).

The G.711 μ-law quantizer is shown in Fig. 6. The origin of the graph occurs mid-tread in the staircase function. This means that inputs smaller than a half of the smallest step size will result in a zero-valued output levels.

Some properties of the segmented μ-law quantizer:

- The two sided μ-law quantizer is sometimes referred to as a 15-segment approximation since the negative and positive segments around the origin are collinear.
- Recommendation G.711 is silent about how to handle an input which lies on a decision level. The reference implementation in Recommendation G.191 has an input value on a decision level is included with the interval above the decision level.
- Having step sizes w**h**ich double ($r = 2$) segment-to-segment allows the 8-bit index to be created using bit-wise operations as described in Appendix A. Appendix B describes table-driven methods to implement non-uniformly-spaced quantizers.

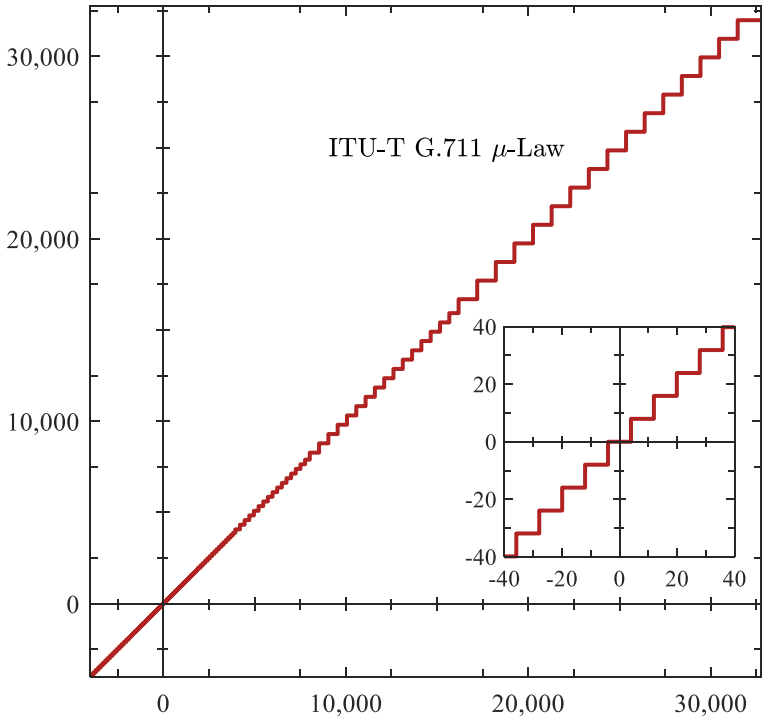

Fig. 6 μ-law quantizer specified in ITU-T Recommendation G.711. The inset shows an expanded region around the origin.

- The G.711 standard defines an 8-bit code which is the bit complement of the 8-bit quantizer index.
- The output value can be obtained using a table lookup of the 8-bit code.

4 *A*-Law Function

The A-law function has a linear part for small values and a logarithmic part for larger values. The scaled log function used in the A-law function is,

$$F(x) = c\log(ax). \tag{30}$$

This is a scaled log function with the offset b in Eq. (1) set zero. Setting the function value for $x = 1$ to be 1, the constant $c = 1/\log(a)$. The logarithmic part is then

$$\begin{aligned} F(x) &= \frac{\log(ax)}{\log(a)} \\ &= 1 + \log_a(x)\,. \end{aligned} \tag{31}$$

The A-law function is most often written with $A = a/e$, giving

$$F(x) = \frac{1 + \log(Ax)}{1 + \log(A)}. \tag{32}$$

The linear part of the A-law function starts at zero for $x = 0$. The line from the origin meets the log function at a point of tangency,

$$[x_t, y_t] = \left[\frac{1}{A}, \frac{1}{1 + \log(A)}\right]. \tag{33}$$

The full A-law function is

$$F_A(x, A) = \begin{cases} \dfrac{Ax}{1 + \log(A)}, & 0 \le x \le \dfrac{1}{A}, \\ \dfrac{1 + \log(Ax)}{1 + \log(A)}, & \dfrac{1}{A} \le \mathrm{x} \le 1. \end{cases} \tag{34}$$

The A-law companding function first appears in Cattermole [5].

The inverse A-law function is

$$F_A^{-1}(y, A) = \begin{cases} \dfrac{1 + \log(A)}{A} y, & 0 \le y \le \dfrac{1}{1 + \log(A)}, \\ \exp((1 + \log(A))(y - 1))\,, & \dfrac{1}{1 + \log(A)} \le \mathrm{y} \le 1. \end{cases} \tag{35}$$

An example of the A-law function is shown in Fig. 7 for $A = 8$. The solid line from the origin meets the log curve at the point of tangency marked with an ×.

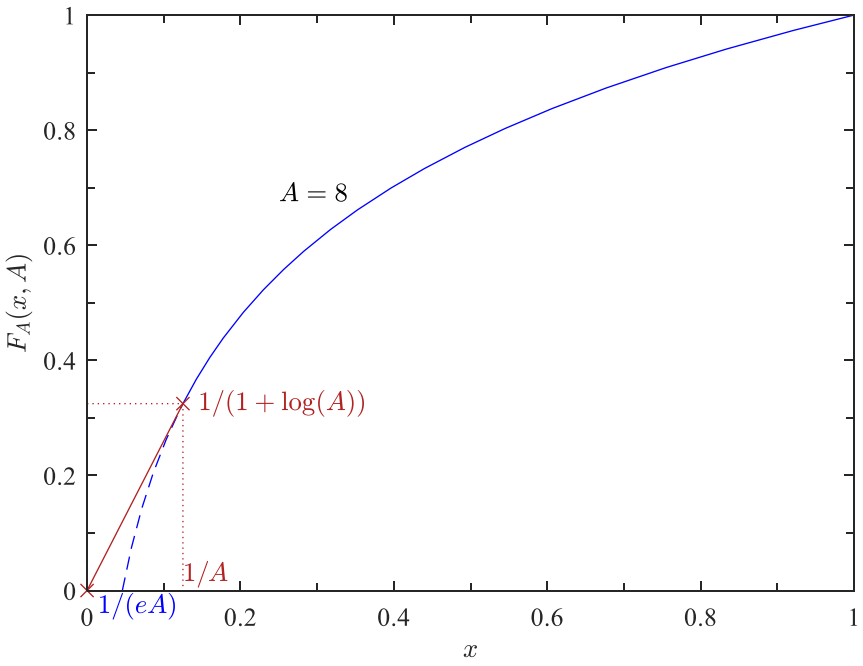

Fig. 7 A-law function for $A = 8$.

The slope of the A-law characteristic is

$$S_A(x) = \begin{cases} \dfrac{A}{1 + \log(A)}, & 0 \le x \le \dfrac{1}{A}, \\ \dfrac{1/x}{1 + \log(A)}, & \dfrac{1}{A} \le \mathrm{x} \le 1. \end{cases} \tag{36}$$

The ratio of the slope at $x = 0$ to the slope at $x = 1$ is

$$\begin{aligned} R_A &= \frac{S_A(0)}{S_A(1)} \\ &= A. \end{aligned} \tag{37}$$

4.1 Generalized *A*-Law Function

The general shifted/scaled log function of Eq. (1) is

$$y = c \log(b + ax), \tag{38}$$

In can be noticed that the μ-law function has an offset $b = 1$, while the A-law function has an offset $b = 0$. For a generalized version of the A-law function, the logarithmic portion can be formed by allowing b to take on values $b \le 1$ (including negative values.). The scaling factor c is chosen so that $x = 1$ maps to $y = 1$, giving $c = 1/\log(b + a)$

The logarithmic function has the lower end at $[x_z, 0]$, where $x_z = (1-b)/a$. The slope of the logarithmic function is $ac/(b+ax)$. The ratio of the slope at $[x_z, 0]$ to the slope at [1,1] is

$$R_{\log} = b + a. \tag{39}$$

The generalized A-law function has a linear portion which starts at the origin and meets the log function at a point of tangency $[x_t, y_t]$. Let the ratio of the slope at $[x_t, y_t]$ to the slope at $[1,\ 1]$ be

$$A = \frac{b + ax_t}{b + a}. \tag{40}$$

Let γ be a measure of the reduction in the range of slopes for the generalized A-law frunction relative to the range of slopes for the logarithmic function,

$$\begin{aligned} \gamma &= \frac{R_{\log}}{A}, \\ &= b + ax_t. \end{aligned} \tag{41}$$

Then the point of tangency is

$$\begin{aligned} x_t &= \frac{\gamma\ - b}{a}, \\ y_t &= c\log(\gamma)\,. \end{aligned} \tag{42}$$

Another expression for the point of tangency is found by equating the slope of the linear portion to the slope of logarithmic portion at the point of tangency,

$$\frac{y_t}{x_t} = \frac{ac}{b + ax_t}. \tag{43}$$

This equation along with Eq. (42) gives an expression for b,

$$b = \gamma(1 - \log(\gamma)). \tag{44}$$

Then from Eq. (39)

$$a = \gamma(A + \log(\gamma) - 1). \tag{45}$$

Substituting for b and a in Eq. (42) gives the point of tangency as

$$\begin{aligned} x_t &= \frac{\log(\gamma)}{A + \log(\gamma) - 1}, \\ y_t &= \frac{\log(\gamma)}{\log(\gamma) + \log(A)}. \end{aligned} \tag{46}$$

With these formulations, the generalized A-law function can be specified entirely in terms of A and γ,

$$F_A(x,A,\gamma)=\begin{cases}\dfrac{A-1+L_\gamma}{L_\gamma+\log(A)}x, & 0\le x\le\dfrac{L_\gamma}{A-1+L_\gamma},\\ \dfrac{L_\gamma+\log\left(1-L_\gamma+(A-1+L_\gamma)x\right)}{L_\gamma+\log(A)}, & \dfrac{L_\gamma}{A-1+L_\gamma}\le x\le 1,\end{cases} \tag{47}$$

where $L_\gamma=\log(\gamma)$. The generalized A-law function becomes the μ-law function when $\gamma=1$ (with $\mu=A-1$) and the conventional A-law function when $\gamma=e$.

4.2 Segmented A-Law Function

Consider sampling the *logarithmic* part of generalized A-law at equally-spaced ordinate values $y_n=k/M$. Then $\Delta_y=1/M$ and using Eq. (5),

$$r^M=b+a. \tag{48}$$

Using the inverse log function of Eq. (2), the x_n values corresponding to the equally spaced y_n values are

$$x_n=\frac{r^n-b}{r^M-b}. \tag{49}$$

Consider including the linear part of the generalized A-law function as part of the segmented function. From geometric considerations, the tangent line has a slope larger than any chord above the point of tangency and a slope smaller than any chord below the point of tangency. This means that the ratio of the tangent line slope to slope of any of the chords cannot fit into the geometric sequence of the slopes of the chords.

In the next section, a modified geometrically spaced function is described. It will be shown that this sequence can be generated by sampling a modified generalized A-law function. The modification involves replacing the tangent line by another line emanating from the origin that intersects the log curve at a point to the left of the tangent point.

4.2.1 Application to Modified Geometrically-Spaced Sequences

The standard A-law quantizer has segment sizes which follow a modified geometric spacing. For uniformly-spaced ordinate values, the abscissa values have intervals of the form

$$\boldsymbol{L}_x=\{D,D,rD,r^2D\ \ldots,r^{M-2}D\}. \tag{50}$$

The lengths of the first two segments are equal before the geometric spacing applies. Fitting these lengths into the $[0,1]$ interval gives

$$D = \frac{r(r-1)}{r^M - r(2-r)}. \tag{51}$$

The segment boundaries are

$$x_k^{(s)} = \begin{cases} 0, & k = 0, \\ \dfrac{r^k - r(2-r)}{r^M - r(2-r)}, & k = 1, \dots, M. \end{cases} \tag{52}$$

A segmented compressive function which maps the non-uniformly spaced x values to uniformly spaced y values is formed when the y segment boundaries are set to

$$\boldsymbol{y}^{(s)} = \left\{0, \frac{1}{M}, \frac{2}{M}, \dots, \frac{M-1}{M}, 1\right\}. \tag{53}$$

This compressive function has been determined without reference to a log function.

Comparing the segment boundaries of Eq. (52) to Eq. (49), the appropriate value of the offset is $b = r(2-r)$. Then in terms of M and r,

$$\begin{aligned} b &= r(2-r), \\ a &= r^M - b. \end{aligned} \tag{54}$$

Equating the value of b as a function of r above and the value of b as a function of γ in Eq. (44), determines γ as a function of r. There is a distinct value of γ for each r. The values γ and $A = \gamma r^M$ specify the log part of a generalized A-law function. The linear part of the linear/log function extends from the origin to $[x_1^{(s)}, y_1^{(s)}]$. An example for the case of $r = 2$ appears in the next subsection.

Modified geometrically-spaced Sequence for $r = 2$

Restrict r to be 2. Then from Eq. (54) $b = 0$ and $a = r^M$. Then from Eq. (44) $\gamma = e$, corresponding to the log part of a standard A-law function. The formula for the x-segment boundaries in Eq. (52) simplifies to,

$$x_k^{(s)} = \begin{cases} 0, & k = 0, \\ \dfrac{1}{2^{M-k}}, & k = 1, \dots, M. \end{cases} \tag{55}$$

For all except the first boundary, the x and y segment boundaries are points on the log part of a (standard) A-law function.

$$y_k^{(s)} = \frac{1 + \log(Ax_k^{(s)})}{1 + \log(A)}, \quad k = 1, \ldots, M, \tag{56}$$

with

$$2^M = Ae; \quad A = 2^M/e. \tag{57}$$

The first segment goes from the origin to a point on the log function at $[1/2^{M-1}, 1/M]$, the next segment (collinear to the first) is a chord to the log function. The initial linear portion can also be interpreted as the double length line from the origin to the point $[1/2^M, 2/M]$.

A plot of the log function and the segmented approximation is shown in Fig. 8. The continuous log curve is for $A = r^M/e$ with $r = 2, M = 4$. The initial segments do not match the tangent line portion of the continuous A-law function. The tangent line extends from the origin to meet the log curve at the point of tangency marked with an x in the figure.

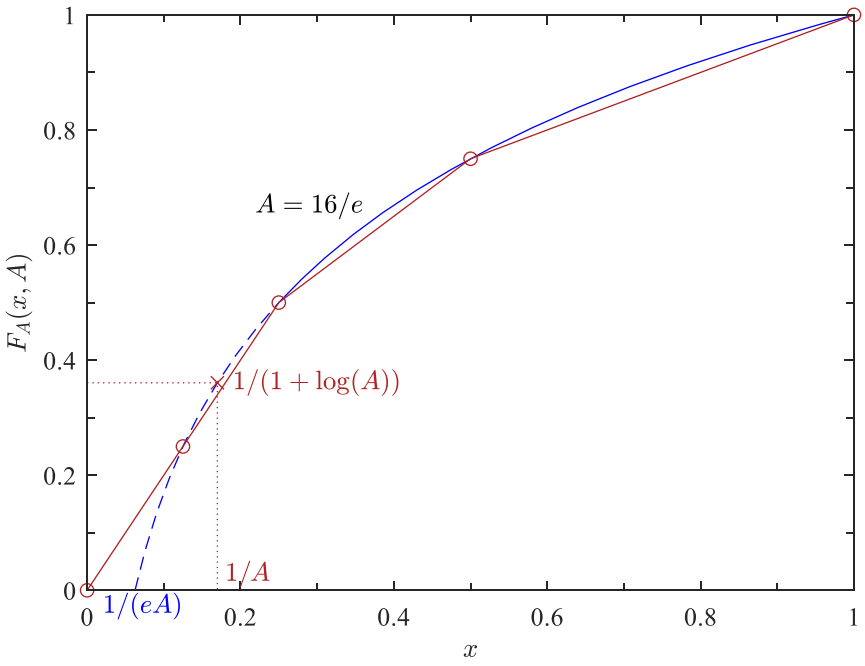

Fig. 8 A-law function for $A = 16/e$ (5.89) and a 4-segment approximation.

4.3 Application to Quantizers

4.3.1 ITU-T Recommendation G.711 – A-law quantization

The International Telecommunications Union Recommendation G.711 [3] uses a segmented A-law characteristic to generate a quantizer with 256 output levels. Each polarity is implemented with $M = 8$ segments. The first two segments have the same step size; for subsequent segments, the step sizes increase by the factor $r = 2$. Except for the first boundary, the segments boundaries lie on the log part of an A-law function with $A = 94.2$ (calculated as $2^M/e$). Within a segment the quantizer characteristic has 16 output levels. The index can be represented with 8-bits (one for the sign, 3 for the segment, and 4 for the level within a segment).

In the literature on quantizers, the segmented function with $M = 8$ is described as approximating an A-law function for $A = 87$.[3] The piecewise linear companding law given by the segmented A-law function used in G.711 was compared to the standard A-law function for various values of A. This was done by generating 1,000,000 uniformly distributed x-values and passing these through the two companding functions. The maximum of the absolute difference was calculated for each value of A. The smallest error (about 1% of full scale) occurred for $A = 87.56$.[4]

First consider positive-valued x. Form the $M = 8$ segment geometrically-spaced values for $r = 2$ using a uniformly-spaced $\boldsymbol{y}^{(s)}$ transformed to a modified geometrically-spaced $\boldsymbol{x}^{(s)}$ values. The $\boldsymbol{x}^{(s)}$ serve as the quantizer decision levels at the segment boundaries. Use linear interpolation between the $\boldsymbol{x}^{(s)}$ values to fill in uniformly spaced decision levels defining a total of 16 intervals between the segment boundaries. The total number of intervals is $N_q = 16M$. The $N_q + 1$ decision levels are the $x_q(n)$ values ($0 \le n \le N_q$). The last decision level ($x_q(N_q)$) is a virtual decision level used only to define the last quantizer output value. The quantizer output values are the mid-points of the intervals defined by the decision levels,

$$y_q(n) = \frac{x_q(n+1) + x_q(n)}{2}, \qquad 0 \le n \le N_q - 1. \tag{58}$$

[3] In [6], "The ITU chose the values A=87.56 and $\mu = 255$ for the G.711, standard…".

[4] If the segmented G.711 μ-law quantizer is compared on the same basis, the best fit is to a standard μ-law function with $\mu = 229.3$.

Scaled values in G.711 A-law

By convention in G.711, the decision levels and output values are scaled such they take on integer values. Set the smallest step size to be 2, with the end of the second segment being 2×32. From Eq. (55), the corresponding normalized decision level at the end of the first segment is $1/2^7$ (for $r = 2$ and $M = 8$). The appropriate scaling factor is $32 \times 128 = 4096$.

After scaling the decision levels at the positive segment boundaries are

$$x_q(16k) = \begin{cases} 0, & k = 0, \\ 16 \times 2^k, & k = 1, \dots, M. \end{cases} \tag{59}$$

The two-sided quantizer with 255 ordered decision levels and 256 output levels can be formed from the one-sided quantizer as follows,

$$\begin{aligned} \boldsymbol{y}'_q &= \{-y_q[N_q - 1], \dots, -y_q[0], y_q[0], \dots, y_q[N_q - 1]\}, \\ \boldsymbol{x}'_q &= \{-x_q[N_q - 1], \dots, -x_q[1], x_q[0], x_q[1] \dots, x_q[N_q - 1]\}. \end{aligned} \tag{60}$$

The step sizes for positive inputs are $\{2,\ 2,\ 4, \dots . 64, 128\}$; the largest output level is 4032; and the overload point is 4096 (calculated as $4032 + 128/2$).

Scaled values for 16-bit inputs

For 16-bit integer input values (values in $[-32{,}768, +32{,}767]$), the decision levels and output levels defined by G.711 will be scaled up by a factor of 8. The segment boundaries (cf. Eq.(59)) become

$$x_q[16k] = \begin{cases} 0, & k = 0, \\ 128 \times 2^k\,, & 1 \le k \le M. \end{cases} \tag{61}$$

Using this scaling, the steps sizes are $\{16, 16, 32, \dots, 512, 1024\}$; the largest output level is 32,256; and the overload point is 32,768 (calculated a $32{,}256 + 1024/2$).

The G.711 A-law quantizer is shown in Fig. 9. The origin of the graph occurs mid-riser in the staircase function. Than means that the low level fluctuations (noise) will cause the quantizer output to swing between the two inner levels. Compared with the G.711 μ-law quantizer shown in Fig. 6, the step size for small input levels is twice as large.

Some properties of the segmented A-law quantizer:

- The two sided A-law quantizer can be considered to be a 13-segment approximation, since the two negative and the two positive segments around the origin are collinear.

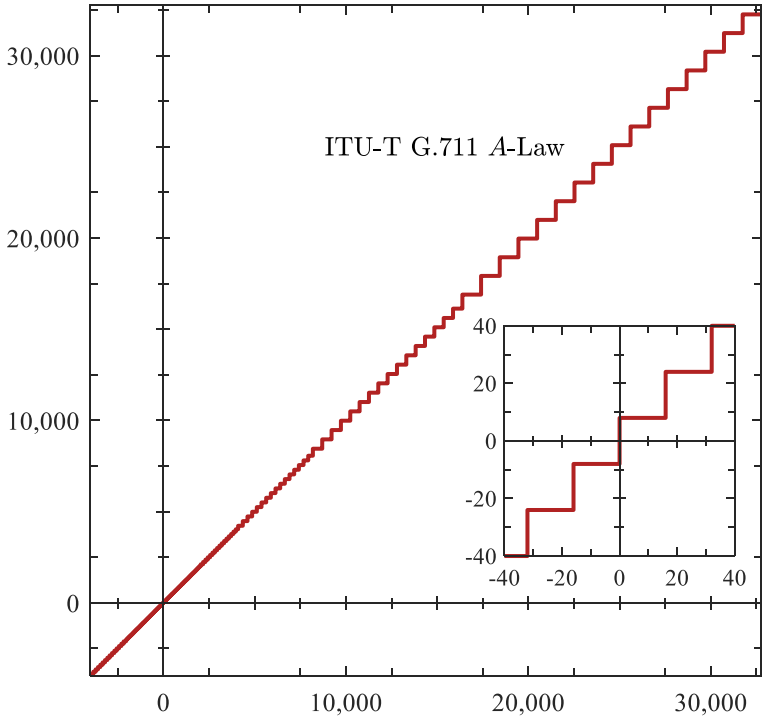

Fig. 9 A-law quantizer specified in ITU-T Recommendation G.711. The inset shows an expanded region around the origin.

- Recommendation G.711 is silent about how to handle an input which lies on a decision level. In the reference implementation in Recommendation G.191, a value on a decision level belongs to the interval above the decision level.
- Having step sizes which double ($r = 2$) with the segments allows the 8-bit index to be created using bit-wise operations as described in the Appendix A. Appendix B describes table-driven methods to implement non-uniformly-spaced quantizers.
- The G.711 standard defines an 8-bit code created using an exclusive-or operation on the 8-bit index value.
- The output value can be obtained using a table lookup of the 8-bit code.

5 Summary and Discussion

This report has described log compressive functions, culminating in a generalized A-law function which subsumes the μ-law and A-law functions. The generalized A-law function has two parameters A and γ. Effectively γ controls the length of the linear portion. With γ chosen such that there is no linear portion, the resulting μ-law function can be segmented. It is shown that the generalized A-law function with a non-zero linear portion is not directly amenable to segmentation. Segmentation of the log-portion of the generalized A-law function, with a modified linear segment does, however, work.

Appendix A describes on standards-based algorithms for implementing segmented μ-law and segmented A-law quantizers.

Appendix B describes table driven approaches for implementing non-uniformly-spaced quantizers. A new alias table method is described. This method uses an auxiliary table to reduce the search space.

References

1. P. F. Panter and W. Dite, "Quantization Distortion in Pulse-Count Modulation with Nonuniform Spacing of Levels", *Proc. IRE*, Vol. 37, No. 1, pp. 44-48, Jan. 1951.
2. B. Smith, "Instantaneous Companding of Quantized Signals", *Bell Sys. Tech, J.*, Vol. 36, No. 3, pp. 653-708, May 1957 (Wiley Online Library).
3. ITU-T, "Pulse Code Modulation (PCM) of Voice Frequencies", *Recommendation G.711*, Nov. 1988 (ITU-T G Series).
4. K. W. Cattermole, *Principle of Pulse Code Modulation*, Iliffe, Second (revised) printing, 1973 (ISBN 978-0-592-028348).
5. K. W. Cattermole, "Contribution to Discussion of Purton", *Proc. IEE – Part B: Electronic and Communication Engineering*, Vol. 109, No. 48, p. 486. Nov. 1962 (IET Digital Library).
6. ITU-T, "ITU-T Software Tool Library 2019 User's Manual", *Recommendation G.191*, Feb. 2019 (ITU-T G Series).
7. ITU-T, "Wideband Embedded Extension for ITU-T G.711 Pulse Code Modulation", *Recommendation G.711.1,* Sept. 2012 (ITU-T G Series).
8. P. Kabal, *Generating Gaussian Pseudo-Random Variates*, Technical Report, Dept. Electrical & Computer Engineering, McGill University, Feb. 2026, on-line document http://www-mmsp.ece.mcgill.ca/MMSP/Documents/Reports.
9. P. Kabal, P. Kabal, *Minimum Mean-Square Error Quantizers*, Technical Report 80-09, INRS-Telecommunications, University of Quebec, May 1980, on-line document http://www-mmsp.ece.mcgill.ca/MMSP/Documents/Reports.
10. P. Kabal, *Quantizer Design,* MATLAB Central File Exchange, retrieved March 24, 2026.
11. Wikipedia contributors, "Rounding", *Wikipedia*, (accessed 2026-03-24).

Appendix A Implementing μ-law and A-law Quantizers

The quantization strategy often used to implement a (segmented) μ-law or A-law quantizer takes advantage of the fact that step size is constant for a segment and that step sizes double from segment to segment. The process identifies the sign (1 bit), the segment index (3 bits), and the level within the segment (4 bits). The result can be combined into an 8-bit code which identifies the quantizer interval

The following description describes the reference implementations in ITU-T G.191, modified for 16-bit integer input values.

A.1 Segmented μ-Law Quantizer

The steps to determine the quantizer interval index are as follows:

1. Record the sign of the input value. If the value is negative, convert it to a positive value.
   a. Twos-complement negation. If the input value is the most negative value ($-32{,}768$), change it to the most positive value (32767). For all other negative values, apply twos-complement negation. Twos-complement negation is used in the reference code for the G.711.1 specification [7]. The effect is that for positive input values, a value on a decision level gives an output value above the decision level (larger magnitude), and for negative input values, a value on a decision level gives an output value below the decision level (larger magnitude).
   b. Ones-complement negation. The magnitude of the positive value is one smaller than the magnitude of the negative value. Ones-complement negation is used in the reference code in the G.191 specification [6]. The effect is that for both positive and negative input values, a value on a decision level gives an output value above the decision level.
2. Saturate the (positive) value to one less than the overload point (to 32,635).
3. The decision levels at the segment boundaries are of the form $128\left(2^k - 1\right) - 4$ (see Eq. (29)). Adding 132 to this gives decision levels which are powers of two, 128×2^k. Adding the same 132 value to the inputs facilitates identifying the segment number – find the position of the most significant one-bit. Finding the segment number is an operation of the form $\lfloor \log_2(x) \rfloor$, but is implemented as a shift loop to search for the leftmost one-bit.

4. Pick up 4 bits after the segment identifying bit. This is the index of the quantization interval within the segment.
5. Form the overall index by concatenating the sign (0 for positive, 1 for negative), segment number (0 to 7), and quantizer level within the segment (0 to 15).
6. Form a code from the index by complementing the bits of the index.

The table below shows the bit patterns used in this algorithm.

Table 1 Bit Patterns for μ-law coding

Segment	$x + 132$ Input Level	Index	$y + 132$ Output Level
0	0000 0000 1abc d...	000 abcd	0000 0000 1abc d100
1	0000 0001 abcd	001 abcd	0000 0001 abcd 1000
2	0000 001a bcd.	010 abcd	0000 001a bcd1 0000
3	0000 01ab cd..	011 abcd	0000 01ab cd10 0000
4	0000 1abc d...	100 abcd	0000 1abc d100 0000
5	0001 abcd	101 abcd	0001 abcd 1000 0000
6	001a bcd.	110 abcd	001a bcd1 0000 0000
7	01ab cd..	111 abcd	01ab cd10 0000 0000

A.2 Segmented *A*-Law Quantizer

The steps to generate the quantizer interval index are as follows:

1. Record the sign of the input value. If the value is negative, convert it to a positive value.
   a. Twos-complement negation. If the input value is the most negative value (−32,768), change it to the most positive value (32767). For all other negative values, apply twos-complement negation. Twos-complement negation is used in the reference code for the G.711.1 specification [7]. The effect is that for positive input values, a value on a decision level gives an output value above the decision level (larger magnitude), and for negative input values, a value on a decision level gives an output value below the decision level (larger magnitude).
   b. Ones-complement negation. The magnitude of the positive value is one smaller than the magnitude of the negative value. Ones-complement negation is used in the reference code in the G.191 specification [6]. The effect is that for both positive and negative input values, a value on a decision level gives an output value above the decision level.

2. The decision levels at the segment boundaries are of the form 128×2^k (see Eq. (61)). The segment number is determined from the position of the most significant one-bit. Finding the segment number is an operation of the form $\lfloor \log_2(x) \rfloor$, but is implemented as a shift loop to search for the leftmost one-bit.
3. Pick up 4 bits after the segment identifying bit. This is the index of the quantization interval within the segment.
4. Form the overall index by concatenating the sign (0 for positive, 1 for negative), segment number (0 to 7), and quantizer level within the segment (0 to 15).
5. Form a code from the index by complementing selected bits of the index (exclusive OR with `0xD5`).

The table below shows the bit patterns used in this algorithm.

Table 2 Bit Patterns for *A*-law coding

Segment	x Input Level	Index	y Output Level
0	0000 0000 abcd	000 abcd	0000 0000 abcd 1000
1	0000 0001 abcd	001 abcd	0000 0001 abcd 1000
2	0000 001a bcd.	010 abcd	0000 001a bcd1 0000
3	0000 01ab cd..	011 abcd	0000 01ab cd10 0000
4	0000 1abc d...	100 abcd	0000 1abc d100 0000
5	0001 abcd	101 abcd	0001 abcd 1000 0000
6	001a bcd.	110 abcd	001a bcd1 0000 0000
7	01ab cd..	111 abcd	01ab cd10 0000 0000

A.3 Decoding

The output value can be created from the code by running the algorithms described above in reverse. A simpler approach is to use the 8-bit code as an index to a table of 256 output levels.

A.4 Rounding Considerations

The bit-wise operations described above start with a 16-bit integer value. If the input is of higher precision (floating point, for instance), then converting to a 16-bit operation is itself a quantization step. To be fully compatible (preserve the correct decision levels), this conversion must use the appropriate form of conversion to integer values.

For case 1a) in the procedures above, twos-complement negation requires that the conversion from the higher precision drop the fractional part. Discarding the fractional part

means that resulting magnitude is less than or equal to the value. For case 1b), the conversion should use the floor function.

If rounding is used, the effective quantizer decision levels are modified. Consider the integer-valued decision level $x_q[i]$. When rounding is used, all values within ½ unit of the decision level will take on the value of the decision level, effectively moving the decision level by ½ a unit.

Appendix B Implementing Non-Uniform Quantizers

Consider a non-uniformly-spaced quantizer defined by a set of $N_q - 1$ finite decision levels. There are N_q quantizer intervals, indexed from 0 to $N_q - 1$. Quantization involves identifying which interval a value x falls in.

B.1 Quantizer Index Search

A simple strategy is to search for the quantization interval. This approach involves up to $N_q - 1$ comparisons. A more efficient strategy is to use a binary search. A snippet of Matlab code for this approach is shown in Fig. 10. The number of iterations needed to identify the quantizer interval is

$$\lfloor \log_2(N_q) \rfloor \le N_i \le \lceil \log_2(N_q) \rceil. \tag{62}$$

```
function index = QuantIndexBS(x, Xq)
% Binary search for the quantizer interval

iL = 0;
iU = length(Xq) + 1;
while (iU > iL + 1)
  i = floor((iL + iU) / 2);
  if (x < Xq(i))
    iU = i;
  else
    iL = i;
  end
end
index = iL;

end
```

Fig. 10 Matlab code for a binary search for quantizer intervals.

The code for the binary search is universal – the same code can be used for any quantizer by using the appropriate table of decision levels. In addition, the code does not have special tests for values below the first quantizer boundary or above the last quantizer boundary.

B.2 Quantizer Alias Table

Another approach uses a method introduced in [8]. This approach is motivated by Walker's alias method which can be used to generate discrete random variables with given probabilities. The alias table is an auxiliary table that is used along with the table of quantizer decision

levels. The alias table provides a lower limit for the quantizer index thus reducing the range of quantizer intervals that need to be searched.

B.2.1 Alias Table

The alias table provides a starting index for a search as follows.

1. Let v take on values from 0 to $N-1$ as x takes on values from $x_q[0]$ to $x_q\left[N_q-2\right]$,

$$v = a(x - x_q[0]). \tag{63}$$

where

$$a = \frac{N-1}{x_q\left[N_q-2\right] - x_q[0]}. \tag{64}$$

As v takes on values $\{0, 1, \dots, N-1\}$, x takes on values $\left\{x_q[0], \dots [N_q-2]\right\}$ in steps of $1/a$.

2. Create an alias table $T_A[\cdot]$ with N integer-valued elements. Table element $T_A[k]$ contains the quantizer index for the value $x_k = x_q[0] + k/a$. The size of the alias table N will determine the maximum number of decision levels $p_{\max}$ that have to be tested. One can choose N to be large enough such that at most $p_{\max}$ decision levels are included in any of the blocks. Conversely, if N is specified, the maximum number of quantizer intervals that need to be searched is

$$p_{\max} = \max_n (T_A[n+1] - T_A[n]\,). \tag{65}$$

The later examples create alias tables for increasing values of N until a specified value of $p_{\max}$ is achieved.

Quantization procedure

1. If $x < x_q[0]$, set $I_q = 0$; or if $x \geq x_q\left[N_q-2\right]$, set $I_q = N_q - 1$. If neither case occurs, continue with the following steps.
2. Identify upper and lower bounds for the quantizer index using $T_{A[}[n]$, where

$$n = \left\lfloor a(x - x_q[0]) \right\rfloor. \tag{66}$$

The quantizer index is bounded by

$$T_A[n] \leq I_q \leq T_A[n+1]. \tag{67}$$

3. Set I_q to $T_A[n]$.Test $x \geq x_q[I_q + 1]$. If so, increment I_q. Continue testing while the test succeeds for $p_{\max}$ steps.

Figure 11 shows Matlab code for the quantization procedure. This code works for any value of $p_{\max}$. For a given value of $p_{\max}$, the `while` loop can be unwrapped to eliminate the last comparison..

```
function index = QuantAlias(x, TA, Xq)
% index <-  Index of the quantizer interval which contains x, 0 <= index <= Nq.
% x      -> Value to be quantized
% TA     -> Alias table of size N. TA records the quantizer regions associated
%           with the bottom edge of the alias table intervals.
% Xq     -> Quantizer break points, Nq-1 values

Nq = length(Xq) + 1;

if (x < Xq(1))
  index = 0;
elseif (x >= Xq(end))
  index = Nq - 1;
else
  %  Find the alias table interval index containing x
  N = length(TA);
  Ds = N / (Xq(end) - Xq(1));
  i = floor((x - Xq(1)) * Ds);

  % Find the index of the quantizer interval corresponding to the bottom edge
  % of the alias table segment. Then x >= Xq(index).
  index = TA(i+1);

  % Test whether x lies above the _next_ quantizer interval
  while (x >= Xq(index + 1))
    index = index + 1;
  end
end
```

Fig. 11 Matlab code for quantizing using an alias table.

B.2.2 Example

Consider a quantizer designed to minimize the mean-square error for an input with a Gaussian probability density. The full quantizer is designed with 256 quantization intervals.[5] The quantizer intervals increase in length on either side of zero. The ratio of the largest interval between the decision boundaries to the smallest interval is about 20.

Table 3 shows the alias table size for different values of $p_{\max}$. For each value of $p_{\max}$, the number of intervals with $0, 1, \ldots, p_{\max}$ decision levels is roughly the same. The maximum number of comparisons needed to isolate the quantizer intervals is $p_{\max}$. This compares with 8 for the binary search algorithm described earlier.

[5] The quantizer design strategy is described in [9]. A Matlab implementation is given in[10].

Table 3 Alias table size, $N_q = 256$

$p_{\max} = 1$	$p_{\max} = 2$	$p_{\max} = 3$
507	261	169

B.2.3 Compression

Since the quantizer table under consideration is symmetric about zero, consider quantizing the magnitude of the input. This half-length quantizer has 128 intervals starting at zero and is defined by 127 decision boundaries. This half-length quantizer has intervals which start off small and increase in length. A compressive function can be applied to the quantizer to reduce the ratio of the smallest interval to the largest interval and thereby reduce the size of the alias table.

Here the compressive functions map an input in [0, 1] to an output in [0, 1].The compressive function must be monotonically increasing and have a slope larger than 1 at $x = 0$ and a slope smaller than 1 at $x = 1$. As a pre-processing step, compression is applied to the quantizer decision level after being shifted and scaled to the interval [0, 1]. During quantization the input value will be similarly shifted/scaled and compressed.

The compressive functions considered use only simple arithmetic operations. The parameters of the functions will be optimized to maximize the size of the smallest quantizer interval.

Quadratic compressive function

A quadratic compressive function is

$$F(x, a) = ax^2 + (1 - a)x. \tag{68}$$

This function is monotonically increasing for $-1 \leq a \leq 0$. For the half-length Gaussian quantizer, after compression the relative size of the smallest interval is obtained for $a = -1$. The slope of the compressor at 0 is 2 and the slope at 1 is zero. This compression functions is simple to compute (1 multiply and 1 add for $a = -1$).

Inverse law compressive function

A compressive function can be derived from the inverse law relationship $v = 1/u$. Let u be a shifted/scaled version of x and let the output be a shifted/scaled version of v. Then the compressor which maps x in [0, 1] to an output in [0, 1] is

$$F(x,r) = \frac{r}{r-1}\left(1 - \frac{1}{x(r-1)+1}\right). \tag{69}$$

The slope at $x = 0$ is r and the slope at $x = 1$ is $1/r$. The values of r for the half-length Gaussian quantizer which give largest smallest quantizer interval is $r = 2.36$. The compressor output can be computed with 2 additions/subtractions, 1 multiplication, and 1 division. If $r = 2$, the multiply can be avoided.

Table 4 shows the table sizes with and without compression is applied. The results show that the alias table size can be reduced with compression, but at the cost of additional computational steps.

Table 4 Alias table size, $N_q = 128$

Companding	$p_{\max} = 1$	$p_{\max} = 2$	$p_{\max} = 3$
None	247	124	83
Quadratic	167	70	47
Inverse	158	79	54

Rounding Values on Quantizer Decision Levels

A quantizer boundary $x_q[n]$ can be included either in quantizer interval below the boundary or in the interval above the boundary. Choosing which case applies is related to the problem of rounding the point mid-way between integer values [11]. Some choices are: a value on the mid-point is rounded up (towards $+\infty$), rounded down (towards $-\infty$), rounded towards zero, or rounded upwards away from zero. Other choices include rounding alternate decision levels up/down.

In this section, the definition of the quantization interval includes the lower boundary of a quantization interval in the interval (see Eq. (24)). In the terminology of rounding, this implements round up. If the input value is a floating-point value and the quantizer boundaries are also floating point, a quantizer boundary can be nudged up to change that boundary to effectively implement round down. This is accomplished by moving the boundary to the next larger floating-point value. A *nextafter* function is available in some computer languages. In Matlab, *nextafte*r for floating-point values can be implemented as `nextafter = @(x)(x+eps(x))`. It the input value is integer-valued and the quantizer boundaries are integer-values, a boundary value can be incremented to effectively implement a round up at the boundary value.