Representation of Numbers

Let us represent a number, for example, 8.25, in a fixed-point format:

$$X = \sum_{i=-A}^{B} b_i \, r^{-i} \qquad 0 \le b_i \le r-1$$

where:

b_i represents the digit — A is the number of integer digits

r is the radix or base — B is the number of fractional bits

Example:

$$(101.01)_2 = [1x2^2 + 0x2^1 + 1x2^0] + [0x2^{-1} + 1x2^{-2}] = 5.25$$

This format is often referred to as a Q-point representation. Operations on such numbers is called Q-point arithmetic. This is a pseudo-floating point representation. Efficient implementations exist for DSPs.

Formats for positive fractions: $X = 0.b_1 b_2 \ldots b_B = \sum_{i=1}^{B} b_i z^{-i}$

What about negative fractions: $X = -0.b_1 b_2 \ldots b_B = -\sum_{i=1}^{B} b_i z^{-i}$

Negative numbers are typically represented in one of three formats:

- Sign-Magnitude Format: Most significant bit (MSB) set to 1

$$X_{sm} = 1.b_1 b_2 \ldots b_B \qquad (X \le 0)$$

- One's-Complement Format: $\overline{b_i} = 1 - b_i$

$$X_{1C} = 1.\overline{b_1}\,\overline{b_2} \ldots \overline{b_B}$$

- Two's-Complement Format: $X_{2C} = X_{1C} + 2^{-B} = 2 - |X|$

$$X_{2C} = 1.\overline{b_1}\,\overline{b_2} \ldots \overline{b_B} + 0\ 0\ \ldots\ 1$$

Binary Floating-Point Representation of Numbers

- A fixed-point representation of numbers allows us to cover a range $x_{max} - x_{min}$ with a resolution:

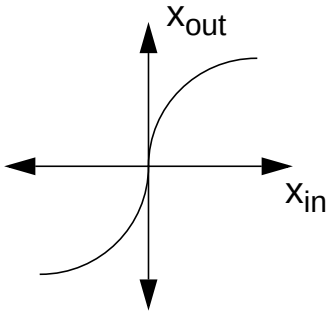
$$\Delta = \frac{x_{max} - x_{min}}{m - 1}$$

where $m = 2^b$ is the number of levels and b is the number of bits.

Drawbacks:

- resolution is fixed
- Δ increases in direct proportion to the increase in dynamic range

- A nonlinear representation overcomes this by allowing precision to be matched to the statistics of the signal (μ-law is an example of this):



- One popular implementation of this strategy is an exponential representation commonly used in digital computers (IEEE Floating Point is the most popular standard; many DSPs use similar formats):

$$X = M \bullet 2^E$$

M is the mantissa (a fixed point number in the range $\frac{1}{2} \le M < 1$), E is the exponent, and $(M + E + 2)$ bits are required.

Why is this a popular representation?

Note that:

$$X_1 X_2 = M_1 M_2 2^{E_1 + E_2}$$

$X_1 + X_2$ requires the exponents of each number to be equal (pre-shift)

Analysis of Sensitivity to Quantization Errors

Recall the general definition of an IIR filter:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \quad \Rightarrow \quad \bar{H}(z) = \frac{\sum_{k=0}^{M} \bar{b}_k z^{-k}}{1 + \sum_{k=1}^{N} \bar{a}_k z^{-k}}$$

where $\tilde{a}_k = a_k + \Delta a_k$ and $\tilde{b}_k = b_k + \Delta b_k$ represent the quantization errors.

We can write the denominator as:

$$\bar{D}(z) = \prod_{k=1}^{N} (1 - \bar{p}_k z^{-1})$$

The perturbation error, Δp_i, can be expressed in terms of partials:

$$\Delta p_i = \sum_{k=1}^{N} \frac{\partial p_i}{\partial a_k} \Delta a_k$$

We can obtain the partial derivatives, $\partial p_i / \partial a_k$, by differentiating $D(z)$:

$$\left(\frac{\partial D(z)}{\partial a_k}\right)_{z=p_i} = \left(\frac{\partial D(z)}{\partial z}\right)_{z=p_i} \left(\frac{\partial p_i}{\partial a_k}\right)$$

Therefore,

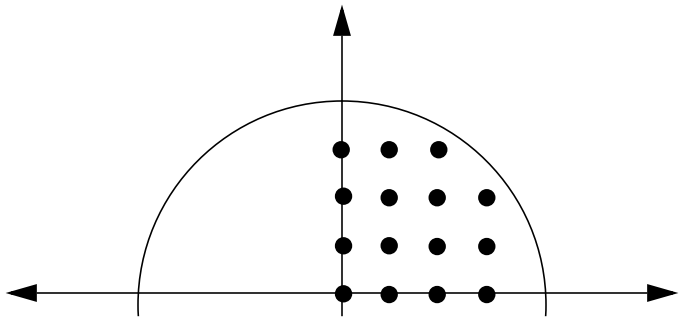
$$\frac{\partial p_i}{\partial a_k} = \left(\frac{\partial D(z)}{\partial a_k}\right)_{z=p_i} \Big/ \left(\frac{\partial D(z)}{\partial z}\right)_{z=p_i}$$

With a little work,

$$\Delta p_i = -\sum_{k=1}^{N} \frac{p_i^{N-k}}{\prod_{\substack{l=1 \\ l \neq i}}^{N} (p_i - p_l)} \Delta a_k$$

Note that when the poles are close, we are in trouble!

The combined effect of representing filter coefficients in finite precision is that we constrain the pole/zero locations of the filter:



Note that the sampling need not be uniformly spaced. In general it is nonuniformly spaced, because the relationship between a pole and a coefficient for a IIR filter is complex.

Finite precision makes it difficult to achieve particular sharp (high-Q) filter designs since the pole/zero locations are constrained to fixed frequencies. Most modern-day filter design programs will follow a process where the initial filter design is done in floating point, and then optimized in fixed-point for the amount of precision selected by the user.

In general, quantization effects are best investigated by numerical simulations, as the particular degradations introduced will depend on the nature of the signal. Signal-to-quantizing noise ratio (SQNR) is often used to evaluate the performance of a given filter design.

In practice, 16-bit wide memory locations and registers, and large accumulators (48 to 56 bits) are sufficient for most high fidelity digital audio applications; speech processing requires even less precision; video even less.

Limit Cycle Oscillation in Recursive Systems

Consider the simple one pole system:

Nonlinearities cause periodic oscillations even when the input is zero. As the input of a filter goes to zero, the filter will oscillate after a few iterations, and sustain a random output bounded in amplitude until a new input is introduced. Such behavior is called a limit cycles, and is very common in nonlinear systems. (They can be conveniently represented in state spaces.)

We can analyze the above systems:

$$y(n) = ay(n-1) + x(n)$$
$$v(n) = Q[av(n-1)] + x(n)$$

The quantization error is bounded by:

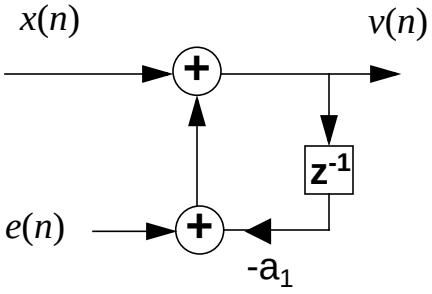
$$|Q[av(n-1)] - av(n-1)| \le \frac{1}{2} \bullet 2^{-b}$$

$$|v(n-1)| \le \frac{\frac{1}{2} \bullet 2^{-b}}{1-|a|}$$

Note that as $a \to 1$, noise becomes greater!

Statistical Characterization of Quantization Effects

We can analyze the noise introduced by a filter as follows:



We seek a relationship between the output noise and input noise:

$$v(n) = Q[av(n-1)] + x(n)$$

$$Q[av(n-1)] = av(n-1) + e(n)$$

We can express this in terms of additive noise:

$$v(n) = av(n-1) + x(n) + e(n)$$

or,

$$v(n) = y(n) + q(n)$$

What should we assume about the noise?

(1) $E[e(n)] = 0$

(2) white, uncorrelated with the signal (always?)

(3) $\sigma_e^2 = 2^{-2b}/12$

Under these assumptions:

$$y(n) = ay(n-1) + x(n)$$
$$q(n) = aq(n-1) + e(n)$$

Making use of our linear systems theory:

$$\sigma_q^2 = \frac{\sigma_e^2}{1-a^2}$$

Note that the noise increases as $a \to 1$.