Digital Signal Processing
Introduction to Finite-Precision Numerical Effects

D. Richard Brown III

Floating-Point vs. Fixed-Point

DSP chips are generally divided into fixed-point and floating-point types. Examples:

- Fixed point: Texas Instruments TMS320C6455, Analog Devices Blackfin, ...
- Floating point: Texas Instruments TMS320C6727, Analog Devices SHARC, ...

Both types can process floating-point data, but the fixed-point processors use software libraries that are very slow. Hence, fixed-point processors are typically only used to process fixed-point data.

Floating-point DSP	Fixed-point DSP
Easier to program	Cheaper
Less likelihood of overflow	Lower power
Usually more accurate	Faster

Common Floating-Point Data Types (IEEE 754)

Single-precision (32 bit, usually the `float` datatype)

- 6-9 significant digits of precision
- smallest positive value: $\approx 1.4 \times 10^{-38}$
- largest positive value: $\approx 3.4 \times 10^{38}$
- Special values for $\pm\infty$ and NaN.

Double-Precision (64 bit, usually the `double` datatype)

- 15-17 significant digits of precision
- smallest positive value: $\approx 10^{-308}$
- largest positive value: $\approx 10^{308}$
- Special values for $\pm\infty$ and NaN.

Huge dynamic range means low likelihood of overflow. MATLAB uses double-precision for most of its calculations. Double-precision is often quite a bit slower than single-precision on most DSP chips.

Common Integer-Valued Data Types

Bits	Usual datatype	Minimum value	Maximum Value
8	`char` or `signed char`	-128	127
16	`short`	-32768	32767
32	`int`	-2^{31}	$2^{31}-1$

Smaller dynamic range than floating point data types means extra care must be taken to avoid overflow.

Given a $B+1$ bit datatype with bits $\{b_0, b_1, \ldots, b_B\}$, decimal values are typically represented with two's complement encoding:

$$\text{decimal value} = -b_0 2^B + \sum_{i=1}^{B} b_i 2^{B-i}$$

Fixed-Point Representation

A fixed-point number representation uses an integer-valued datatype and associates with it a certain number of **fractional bits**, denoted as q. This simply scales the integer value as

$$\text{decimal value} = 2^{-q}\left(-b_0 2^B + \sum_{i=1}^{B} b_i 2^{B-i}\right)$$

Note that q is usually positive, but can be negative.

As an example, suppose we want to quantize $\frac{1}{\sqrt{2}} \approx 0.7071$ to a fixed point number with $B+1=4$ total bits. The integer datatype has a range of -8 to $+7$.

Frac bits q	Quantized value (dec)	Quantized value (bin)	Integer value
0	1	$0001_\diamond$	1
1	0.5	$000_\diamond 1$	1
2	0.75	$00_\diamond 11$	3
3	0.75	$0_\diamond 110$	6
4	0.4375	$_\diamond 0111$	7

Fixed-Point Products

Consider:

- Fixed point variable x with $B_x + 1$ total bits and q_x fractional bits.
- Fixed point variable y with $B_y + 1$ total bits and q_y fractional bits.

If $z = x \times y$, then z will require

$$B_z + 1 \geq B_x + B_y + 2$$

total bits and

$$q_z \geq q_x + q_y$$

fractional bits to capture the product without any possibility of overflow or loss of precision.

Many fixed-point DSPs have an extended-precision (also called double-length) accumulator to accommodate fixed-point products without any possibility of overflow or loss of precision.

Fixed-Point Sums

Consider fixed point variables $\{x_1, \ldots, x_N\}$ all with $B_x + 1$ total bits and q_x fractional bits. If

$$z = \sum_{i=1}^{N} x_i$$

then z will require

$$B_z + 1 \geq B_x + 1 + \log_2(N)$$

total bits and

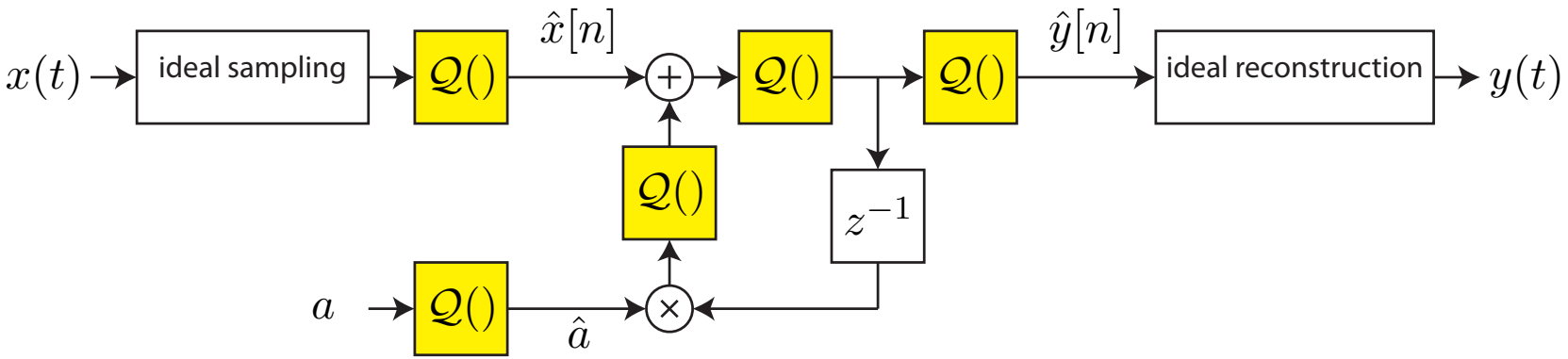
$$q_z \geq q_x$$

fractional bits to capture the sum without any possibility of overflow or loss of precision.

For fixed-point products and sums, if the z datatype does not have enough bits to hold the result, we usually reduce precision (decrease the number of fractional bits) to avoid overflow.

Overview of Finite-Precision Effects

- Input/output quantization, i.e., ADC and DAC.
- Filter coefficient quantization.
- Product roundoff.
- Potential overflow in sums.



In general, nonlinear systems like this are difficult to analyze. We will isolate the sources of quantization error and adopt some approximate approaches to make the analysis tractable.

Fixed-Point Filter Design in `fdatool`