References:

1. Discrete-Time Signal Processing
   1. 4.8.2 A/D Conversion
   2. 6.7 Overview of Finite-Precision Numerical Effects

# A/D conversion

High performance ADCs usually consist of two circuits: a sample-and-hold circuit that samples the input waveform as quickly as possible and holds it as constant as possible and an A/D converter that converts the input voltage or current into a binary code.

The A/D converter can be further split into a quantizer followed by a coder. The quantizer converts from real numbers to a finite set of numbers, and the coder converts the finite set of numbers to binary numbers.

# Two’s complement

There are different types of coding. For example, for 3 bits,

Table

Description automatically generated

Two’s complement allows us to do arithmetic directly with the code words. For example, 000 represents 0 in two’s complement but -4 in offset binary code: In two’s complement, adding 000 to any binary number intuitively returns that binary number.

Two’s complement is a type of modular arithmetic, which is defined as any integer system where the numbers wrap around after reaching the modulus. In two’s complement, the modulus is , where is the total number of bits.

For the binary code , where and is the MSB/sign bit, the decimal number is

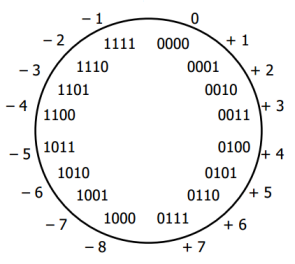
We can normalize this to . The binary code becomes , and the decimal number is

These two representations are given in this table:

Table

Description automatically generated

Example graphical representation of two’s complement that shows the modularity (clocks/time are another modulator system):



# Finite-precision numerical effects

## Round off and quantization error

Any real number may be represented with infinite precision in two’s complement as

In this notation, the MSB/sign bit is (otherwise it’s difficult to write this expression). When , , and when , .

Switching back to the original notation, where the MSB/sign bit is , the quantized number is

Where , , and is the bitwidth.

The smallest difference between quantization levels (1 LSB) is .

Quantization error is defined as the difference between the real number and the quantized number:

When rounding, . When truncating, .

Quantization error also occurs when quantizing from bits to bits, where . Let . Then the quantization error bounds are for rounding and for truncation.

Explanation: When quantizing from real numbers to , there can be a max error (for truncation) of , so the error of the subsequent truncation is reduced by up to .

## Overflow and saturation

Table

Description automatically generated

In two’s complement, addition and subtraction is modular (for modulus). This means that when overflow happens, the result “wraps around”. For example, . In decimal, . Another example, , or . Left untreated, overflow results in large error.

Saturation prevents overflow: When a sum would overflow, saturation clips the sum to the maximum (011) and minimum (100) values.

An interesting property of modular arithmetic: When adding multiple numbers together, if the overall sum does not overflow, then even if the intermediate sums overflow, the final sum does not overflow. However, saturation breaks this property.

# System analysis of quantization (TBD)

6.7.2

# Round-off noise in digital filters

Reference: DTSP 6.9

Quantization, whether rounding or overflow or saturation, is nonlinear, but in most practical applications, precise analysis of these effects is not required. Instead, linear additive noise sources may be used to model these effects, except for the phenomenon of zero-input limit cycles, which is strictly nonlinear.

Often, the most effective approach is to simulate the system and measure its performance.

# Quantization noise for different fixed-point operations

## Summary table

is the number of bits to round off.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Noise Mean ()** | **Noise Variance ()** | **Noise Power (rms)** |
| Truncation |  |  |  |
| Round Up |  |  |  |
| Round |  |  |  |

## Truncation

From the graphical representation of two’s complement, truncation – removing the LSBs – always incurs a nonpositive error. For example, removing 2 LSBs from 1111 (-1) gives 1100 (-4), an error of -3. Removing 2 LSBs from 0101 (+5) yields 0100 (+4), an error of -1. Given , an integer, and , the number of bits to round off, truncation is in two’s complement.

Error due to truncation takes possible values:

For uniformly distributed error, the mean is

The variance is

See “Square Pyramidal Number” on Wikipedia: The sum of squares is given by

In our case, . Then

Then

This matches the analysis under the assumption that the error can take any value (not just discrete values) in :

RMS noise power is equal to

## Round up (asymmetric rounding)

Round up means adding before truncation, which means

The mean of the error is greatly improved. Variance is the same as truncation. Round up is .

## Round (symmetric rounding)

For rounding, and variance is the same as truncation. Round is .

# SQNR

## Analog-to-digital quantization noise

Let be 1 LSB. Then quantization noise samples are uniformly distributed from to . The variance of the noise, which is equal to its average power, is given by

Where is the quantization noise random variable. Then

Let’s say you have bits – signed or unsigned doesn’t matter. In this case, assumed signed.

Let’s say you have a full-scale sinusoid, e.g. and

The power of I or Q is

Since is 1 LSB, in this example, . So SQNR is

If you include both I and Q, signal power rises by 3dB but so does noise power, so overall SQNR remains the same.

## Digital-to-digital quantization noise

I don’t think this is any different from analog-to-digital quantization noise, except maybe less random. The bits you cut off are the fractional bits

Let be the signed data bitwidth. Then signal power is

Then SQNR (linear) is

This expression is independent of and is identical to the analog-to-digital quantization noise derivation.

## Cascaded quantization noise (TBD)