Quantization – brief overview

Quantization is the process of mapping a large set of input values to a smaller set of discrete levels.
In digital signal processing it is used for:

1. **Audio/Video compression** – reducing the number of bits needed per sample.
2. **Image quantization** – converting 24-bit color images to fewer colors (palette reduction).
3. **Neural networks** – lowering weight precision from floating point to fixed-point or integer.

Typical steps:

- Choose a number of levels L (e.g., 256 for 8-bit output).
- Compute the step size $\Delta = \frac{max - min}{L-1}$.
- For each input value x, compute its quantized index $q = \text{round}\left(\frac{x - min}{\Delta}\right)$.
- Reconstruct (dequantize) with $x' = min + q \cdot \Delta$.

Below is a simple C program that demonstrates scalar uniform quantization on an array of floating-point numbers, then reconstructs the signal and computes mean squared error.

Below is a simple C program that demonstrates scalar uniform quantization on an array of floating-point numbers, then reconstructs the signal and computes mean squared error.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Quantize a single sample */
int quantize(double x, double min, double max, int levels) {
   double step = (max - min) / (levels - 1);
   int q = (int)round((x - min) / step);
   if (q < 0)      q = 0;
   if (q > levels-1) q = levels-1;
   return q;
}

/* Dequantize to a floating-point value */
double dequantize(int q, double min, double max, int levels) {
   double step = (max - min) / (levels - 1);
   return min + q * step;
}
```

```
int main(void) {
    /* Example data: sine wave samples */
    const int N = 1000;
    double *data = malloc(N * sizeof(double));
    for (int i = 0; i < N; ++i)
        data[i] = sin(2.0 * M_PI * i / N);

    /* Determine min/max of the dataset */
    double min = data[0], max = data[0];
    for (int i = 1; i < N; ++i) {
        if (data[i] < min) min = data[i];
        if (data[i] > max) max = data[i];
    }

    /* Quantization parameters */
    int levels = 256;          /* 8-bit quantizer */

    /* Allocate arrays for quantized indices and reconstructed samples */
    int   *q      = malloc(N * sizeof(int));
    double *rec   = malloc(N * sizeof(double));

    /* Quantize and dequantize */
    double mse = 0.0;
    for (int i = 0; i < N; ++i) {
        q[i] = quantize(data[i], min, max, levels);
        rec[i] = dequantize(q[i], min, max, levels);
        double diff = data[i] - rec[i];
        mse += diff * diff;
    }
    mse /= N;

    printf("Mean squared error after %d-level quantization: %.6f\n", levels, mse);

    /* Clean up */


    free(samples); free(indices); free(reconstructed);
    return 0;
}
```