Theorem 2.1.2. *If $f : \mathbb{R} \longrightarrow \mathbb{R}$ is an integrable function of bounded variation such that $f(x+h) + f(x-h) \to 2f(x)$ as $h \to 0$ for every $x \in \mathbb{R}$, then* (2.9) *holds true.*

Indeed, with some modifications in (2.9) it is possible to cover not integrable cases like $f(x) = x^{-\alpha}$ [Gui41]. Poisson summation formula has striking consequences in a broad range of topics [CR17], for instance, believe or not the best known results on sphere packing are based on it [Coh17].

If we apply (2.9) to $f(x) = g(qx+a)$, we get the following generalization

$$\sum_{n=-\infty}^{\infty} g(qn+a) = \frac{1}{q} \sum_{n=-\infty}^{\infty} e(an/q)\widehat{g}(n/q). \tag{2.11}$$

For $q = 1$, $a = 0$ we recover (2.9). In principle one could derive Shannon sampling theorem from here taking $g(x) = \operatorname{sinc}(\nu_s x) f(t+x)$ with $q = \nu_s^{-1}$ and $a = -t$. In this way the left hand side of (2.11) is the right hand side of (2.3).

Note also that for $q = 1$, $a = x$ we get the Fourier expansion of F in (2.10). In general, Poisson summation formula can be used instead of Fourier expansion when the coefficients can be easily interpolated to a smooth function.

One of the most famous and interesting applications of the Poisson summation formula is the θ modular relation

$$\sum_{n=-\infty}^{\infty} e^{-2\pi\alpha n^2} = \frac{1}{\sqrt{2\alpha}} \sum_{n=-\infty}^{\infty} e^{-\pi n^2/2\alpha} \qquad \text{for} \quad \alpha > 0. \tag{2.12}$$

If α is very small the first sum is expensive from the computational point of view while the second gives readily the approximation $1/\sqrt{2\alpha}$ with exponential gain.

Suggested Readings. The whole book [Mar91] is devoted to topics around Shannon sampling theorem. In [Byr15] there are several discussions about sampling and reconstruction scattered along several chapters. See [CR17] for more about the Poisson summation formula.

2.1.2 Basic quantization

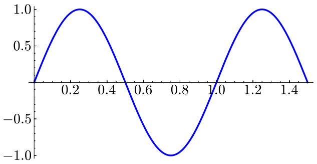
Recall the scheme (2.1). The sampled values $f(t_n)$ are real numbers and we want to approximate them by discrete quantities. This process is called *quantization*.

The function mapping each real number into the nearest integer, sometimes called "round", is given by the formula $x \mapsto \lfloor x + 1/2 \rfloor$ where $\lfloor x \rfloor$ is the integral part defined in (1.56). If instead of the integers $\mathbb{Z}$ we want the output to be Δ multiples of integers to get more (or less precision), we re-scale the argument and the value of this function to find the so called *uniform quantizer*

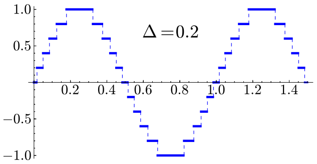
$$Q(x) = \Delta \lfloor \frac{x}{\Delta} + \frac{1}{2} \rfloor \qquad \text{with } \Delta > 0. \tag{2.13}$$

This is the simplest and more common quantization. The geometric idea is that $Q(f)$ is the approximation of f by a step function such that the spacing between steps is always a multiple of Δ. With Δ small the approximation will be good.

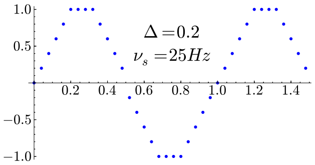
Do not forget that in our case we apply Q to a discrete set of values after sampling not to a function or signal. In computer science very often the application of the uniform quantizer is called *pulse code modulation* and abbreviated as PCM (sometimes preceded by the word “linear”). For instance the (uncompressed) `wav` audio file format uses PCM with a sampling frequency of $44100\,Hz$.



$\mathsf{f(x) = \sin(2\pi x)}$

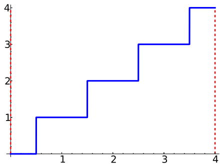


$\mathsf{f(x) = Q(\sin(2\pi x))}$

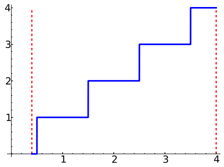


sampled & quantized

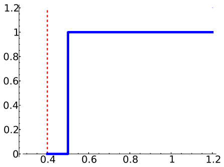
In practice, we often manage signals confined to an interval $[\alpha, \beta)$. If we use the uniform quantifier (2.13) as it is, the edges induce in general undesirable results. For instance, when applied with $\Delta = 1$ in the interval $[0, 4)$ and $[0.4, 4)$ we get



Interval $[0, 4)$



Interval $[0.4, 4)$



(detail)

In both cases we are using five levels and the results is not uniform. In the first example we use one level for $[0, 1/2)$ and another for $[1/2, 3/2)$ although the former interval has half length. The second example is even worse because we spend a whole level for the tiny interval $[0.4, 0.5)$.

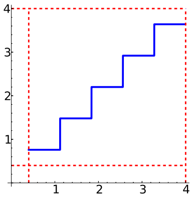
This problem does not appear if $\Delta^{-1}\alpha$ and $\Delta^{-1}\beta$ are half-integers (numbers of the form $k + 1/2$, $k \in \mathbb{Z}$) because in this situation Q jumps at α and β. We can force it with a convenient translation. Say that we want to employ $M \in \mathbb{Z}^+$ uniform levels and define for $I = [\alpha, \beta)$

$$\Delta = \frac{|I|}{M} \qquad \text{where} \quad |I| = \beta - \alpha. \tag{2.14}$$

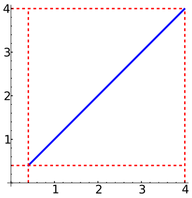
The translation $T(x) = x - \alpha - \Delta/2$ moves $[\alpha, \beta)$ to $[\alpha', \beta')$ with $\Delta^{-1}\alpha'$ and $\Delta^{-1}\beta'$ half-integers, then the *uniform quantizer* for $[\alpha, \beta)$ is the function

$$Q_{[\alpha,\beta)}(x) = (T^{-1} \circ Q \circ T)(x) = \Delta\lfloor \Delta^{-1}(x - \alpha)\rfloor + \frac{\Delta}{2} + \alpha. \tag{2.15}$$

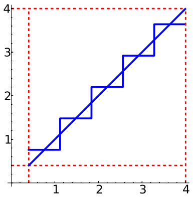
For instance, for $[0.4, 4)$ and $M = 5$ we have



uniform quantizer



identity



both

In this way the image of $Q_{[\alpha,\beta)}$ are the multiples of Δ displaced by $\Delta/2+\alpha$ because the multiples of Δ do not fit $[\alpha,\beta)$ in a uniform way. Note that $Q_{[\alpha,\beta)}$ fixes $\alpha+\delta/2$, $\beta-\Delta/2$ and in general $\alpha+\Delta(k+1/2)$ which are the centers of the intervals where $Q_{[\alpha,\beta)}$ is constant.

The *quantization error* for a certain x is $Q(x)-x$. If the sampled values are uniformly distributed in $[0,1]$ then the mean square error when we use the formula (2.15) is

$$\int_0^1 \big(Q_{[0,1)}(x)-x\big)^2\,dx=\frac{\Delta^2}{12}+O(\Delta^3). \tag{2.16}$$

In fact the term $O(\Delta^3)$ can be omitted, getting an exact formula, if $\Delta^{-1}\in\mathbb{Z}^+$ as required in (2.14). Hence typically the quantization error is like $\Delta/\sqrt{12}$.

Sometimes the processing of the digital data imposes some special ranges and scales and, in some sense, part of the quantizer (2.13) is applied in the A/D conversion and part in the the D/A conversion of (2.1). For example, imagine that we have a rectangular B/W image (this means shades of gray) that is sampled, let us say with a photometer, to get the gray tone of the pixel (i,j). This gives a collection of real numbers a_{ij}, a matrix, and we assume the normalization $a_{ij}\in[0,1)$ with $a_{ij}=0$ pure black and $a_{ij}=1$ pure white (as done in some applications). In the digital side we have a finite palette of c gray tones, commonly $c=256$ in our computer. Then $M=c$ in (2.14) gives $\Delta=c^{-1}$ and the effect of the uniform quantizer (2.15) is

$$\begin{aligned} Q_{[0,1)}\colon [0,1) &\longrightarrow \Big\{\frac{1}{2c},\frac{3}{2c},\frac{5}{2c},\dots,\frac{2c-1}{2c}\Big\}\\ a_{ij} &\longmapsto c^{-1}\big(\lfloor ca_{ij}\rfloor+1/2\big)\end{aligned} \tag{2.17}$$

But our computer internally works with integers then our digital processor prefers to receive only this part of the quantization:

$$\begin{aligned} Q^*\colon [0,1) &\longrightarrow \{0,1,2,\dots,c-1\}\\ a_{ij} &\longmapsto \lfloor ca_{ij}\rfloor\end{aligned} \tag{2.18}$$

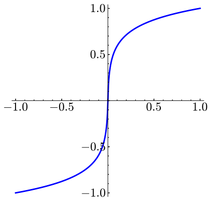
Once the information has been processed digitally, we sum $1/2$ and multiply by c^{-1} in the D/A converter of (2.1) and present it as a fake analog output or we smooth it to get a *bona fide* analog signal. There is a funny terminology around these simple concepts but probably you would find it verbosity. If you are eager to learn it, read [You14, Ch.2].

Uniform quantization is sometimes wasteful. If your signal is a sound wave corresponding to a conversation and you reserve an interval to cover a generous range of amplitudes (volume), typically the upper part of the range is seldom used because people is not shouting all the time (unfortunately there are counterexamples). Also a property of perception, sometimes called *Weber's law* enters in the game. It turns out that, especially beyond certain thresholds, one only feels proportional changes in a stimulus. It is like saying that our senses are logarithmic. If after quantization we get 2^{16} possible values (as in `wav` format) that we represent with 2 bytes but most of the time these numbers are small and when they are big we can allow large errors without noticing anything, it is clear that we are wasting bytes[2]. Solving this problem is closely related to coding, a topic with a pretty mathematical background that we shall treat later in these notes. Here we mention an idea at the analog level and an algorithm to devise optimal quantizers.

The idea is to apply a bijection f to the signal immediately after or before sampling in such a way that the sampled values become uniformly distributed or close to it. In this situation (2.13) is optimal. Correspondingly, the inverse function f^{-1} has to be applied in the last step of (2.1). In the previous example, if the upper part of the range is reached less frequently, then f must compress this part of the range.

A practical example is the *μ-law*, a standard in speech processing in telecommunications in U.S.A. and Japan. Once the signal is normalized to fit in $[-1,1]$, it is applied

$$f(x) = \operatorname{sgn}(x)\frac{\log(1+\mu|x|)}{\log(1+\mu)} \tag{2.19}$$



with μ a certain fixed constant. Note that $f : [-1,1] \longrightarrow [-1,1]$ is a homeomorphism that can be inverted with a simple explicit formula. The upper and lower parts of the interval $[-1,1]$ are compressed because the changes in these zones are less audible. On the other hand, for x around 0, the function f is well approximated by Cx with C a constant i.e., it is just a linear scaling.

The μ-law (2.19) produces a quasi-uniform distribution when applied to typical voice signals but in other applications similar functions can be expensive to get and invert. There is an algorithm than allows to get a similar effect acting on quantization knowing the density function g of the sampled real signal. To design a quantizer Q_g adapted to it, we consider that it is characterized by a partition of $\mathbb{R}$ into a finite number of intervals and a way to assigns to each interval a fixed element in it. In the jargon, the intervals are called *quantization regions* and the elements the *representation points*. Let us write

$$\mathbb{R} = \bigcup_{j=1}^{M} I_j, \quad I_1 = (-\infty, b_1), \quad I_M = [b_{M-1}, \infty), \quad I_j = [b_{j-1}, b_j) \text{ for } 1 < j < M, \tag{2.20}$$

with $a_j = Q_g(I_j) \in I_j$ the representation points. Clearly if g has a lot of mass around a point the a_j should cluster there. One could consider optimal a quantizer minimizing the mean square quantization error

$$\int_{-\infty}^{\infty} \big(Q_g(x) - x\big)^2 g(x)\, dx = \sum_{j=1}^{H} \int_{I_j} (a_j - x)^2 g(x)\, dx. \tag{2.21}$$

[2] I bet very rarely you store songs in your portable music player in `wav` format. Ripping CDs to another lighter formats (and even buying CDs) was a must not so long time ago.

This is a function F in $2M-1$ variables $b_1, \dots, b_{M-1}, a_1, \dots, a_M$ and their extrema are reached at values with $\nabla F = \vec{0}$. By the fundamental theorem of calculus, we have

$$\frac{\partial F}{\partial b_j} = (a_j - b_j)^2 g(b_j) - (a_{j+1} - b_j)^2 g(b_j) \qquad \text{for } j = 1, 2, \dots, M-1, \tag{2.22}$$

and differentiating under the integral

$$\frac{\partial F}{\partial a_j} = 2 \int_{I_j} (a_j - x)^2 g(x)\, dx \qquad \text{for } j = 1, 2, \dots, M. \tag{2.23}$$

If $g(b_j) \neq 0$, the vanishing of (2.22) is equivalent to $2b_j = a_j + a_{j+1}$. The *Lloyd-Max algorithm* [Llo82] [Max60] takes this information to solve $\nabla F = \vec{0}$ by successive approximations. Namely, the algorithm starts with any educated guess for the representation points and applies successively the following formulas (in the indicated order)

$$b_j = \frac{a_j + a_{j+1}}{2} \quad \text{for } j = 1, 2, \dots, M-1 \quad \text{and} \quad a_j = \frac{\int_{I_j} xg(x)\, dx}{\int_{I_j} xg(x)\, dx} \quad \text{for } j = 1, 2, \dots, M. \tag{2.24}$$

Note that the second set of equations solves (2.23) equated to 0. The algorithm ends when the values of the b_j's and a_j's stabilize with certain precision. A drawback of the algorithm is that it could converge to a critical point different from one in which the absolute minimum of F is attained. One can cook some counterexamples of this kind [Gal06] choosing a distribution function g with some valleys but anyway one trusts the educated initial guess should avoid this problem.

Suggested Readings. As you see, from the mathematical point of view, quantization is not a big deal. Probably you do not need extra bibliography. I only dare to repeat that in [You14] you can find the odd terminology used by engineers.

2.1.3 Data approximation

In practice, Theorem 2.1.1 allows to reconstruct a band limited function from their sampled values but as we mentioned in connection with Papoulis-Gerchberg algorithm and (2.8), its practical efficiency is objectionable in several situations. On the other hand, after quantization and digital processing very rarely it is judicious to assume that the resulting data are values of a band limited function.

Here we approach D/A conversion in a simple general mathematical form: We have discrete (digital) data and we want to approximate them with a somewhat smooth function. We restrict ourselves to 1D samples, as before, then have in mind sound signals rather than images.

You know how to do it, finite samples, simple functions like polynomials... join the dots, literally. It is interpolation from basic courses of numerical analysis. You have some *interpolation nodes* $x_0 < x_1 < \cdots < x_n$ and you want to find a polynomial P such that $P(x_j) = y_j$ for some given y_j. There is a nice theorem, with a short beautiful proof, providing the error with respect to the purported smooth function connecting the dots.