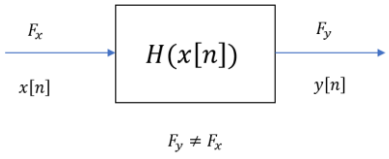
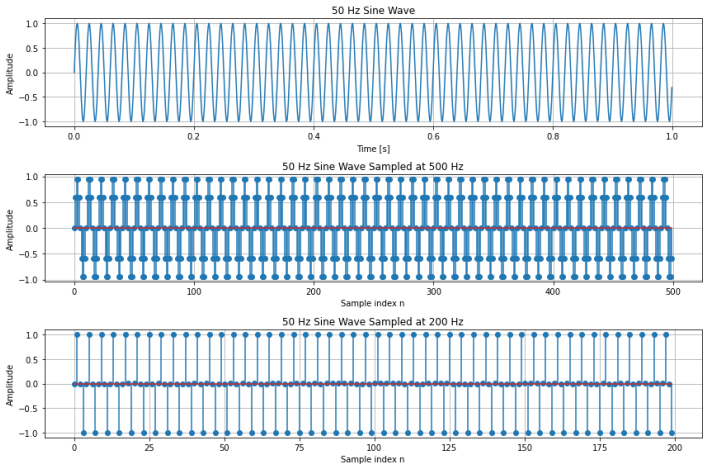
Signal Processing Algorithms & Architecture
Dr. Anirban Dasgupta
Department of Electronics & Electrical Engineering
Indian Institute of Technology Guwahati
Lec 14: Multi-rate Signal Processing – I

Hello everyone, welcome to a fresh new lecture on the topic of Multirate Signal Processing 1. This is Dr. Anirban Dasgupta and let us get started. So, this is a system, and we all know that this is basic stuff. We input a signal x[n] to this system, and we get an output y[n]. Now, if we see, this looks like a pretty simple system, and this system may be linear or may not be linear.



So, H is a transformation that transforms my input x[n] to y[n]. Now, in this case, both the discrete-time signals operate at the same frequency. By frequency, I mean the sampling frequency or the sampling rate. But imagine a system in which the input and output sampling rates are different.

Like the input sampling rate is F_x, the output sampling rate is F_y, and my $F_x$and F_y, are different from each other. So, this kind of system is called a multirate signal processing system. So here it involves manipulation of signals at different sampling rates. So visually, if you see, if I have a sine wave of 50 Hertz and I want to sample it, the Nyquist rate will be twice 50, which is 100 Hertz. But if you see this, 50 hertz can be sampled at different rates; for example, it can be sampled at 200 hertz or at 500 hertz.



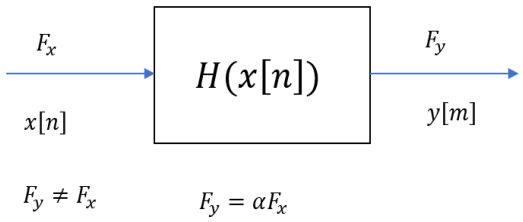
So, what is the big deal? The big deal is that this 500 hertz sample rate is giving a better structure of the original analog signal. However, it comes at a cost of 500 samples. Now, if each sample is, say, 32 bits, then this will be the total memory size. On the other hand, the signal sampled at 200 Hertz captured much less information, but this is good enough to reconstruct it based on Nyquist, though it will take up less space. So, if you clearly see, this version is better for transmission and storage, whereas this version is better for analysis.

Also, computational complexity will be less in this version because of a smaller number of samples. But information is greater at the 500 hertz sampling rate signal. So, where is this multirate signal processing used? Now this is typically used at analog to digital and digital to analog converters. So, when I am recording this video, I am not only recording my video, which is a sequence of images, but I am also recording my sound and my audio is probably recorded at 48 kilohertz, and my video is recorded at, say, 30 frames per second.

So you can see that these two are quite different from each other. So my frames are coming at 30 frames per second, whereas these audio signals are coming at 48,000 samples per second. Now when you are listening, you are listening to it together. So, the systems or the hardware that are doing this kind of operation are operating at different sampling rates. So, this is a typical example of a multirate signal-processing system.

Now, sometimes in audio processing, like mixing, you all listen to a lot of songs. So, the vocals of these songs might be recorded at different sampling rates because the voice has a different bandwidth, whereas the instrumentals can be recorded at different rates because they have a higher frequency content. So, finally, they are combined. So, that is also a case of multirate signal processing. Then interpolation is used when you have fewer samples and want to find the value in the middle of some samples.

So then efficient filter implementations, so using multirate I can efficiently implement specific filters. Multirate signal processing also forms the basis for filter banks. And finally, the discrete wavelet transform, which is a very important tool in time-frequency analysis, uses inherent multirate signal processing. So now let us try to understand the methods of multirate signal processing. Recap, this is our multirate signal processing system, and here you can see that I have written $y[m]$, some texts write it as y[n], while others write y[m].



I have written y[m] just to signify that this is not the same index that is obtained by sampling the signal at f(x), where the index was n. But for simplicity, you can also use n; it does not really matter. So, we can represent my output frequency f of y as some scalar value alpha times my f of x. And now, what is my alpha? Alpha can be either rational, like twice of f(x), 2 by 3 of f(x), and here there are some specific cases of rational numbers; for example, it can be an integer, such as α being 2, which means f(y) is twice f(x). I am just doubling the sampling frequency, or it can be $\frac{1}{m}$ like one-third of my original sampling frequency, or it can be any rational fraction l by m, where l and m are positive integers. So here, integer means positive integer, of course, and then α can also be arbitrary; like, I can select f of y as some arbitrary number, 1.31 f(x), or I can choose f(y) as something random like 1024, and f(x) can be something random, like say 160. So, there is no clear relation with α here, although it looks like a rational factor; I am free to choose any, and I can even choose a decimal. So, if I want to choose α as arbitrary, then this is the method to convert the sampling rate: I will first pass the digital signal, which is sampled at f(x), through a DAC.

Now, when I pass through a DAC or a digital-to-analog converter, it becomes an analog signal again, and then I will resample it using an ADC at f(y). So, f(x) to analog and then analog to f(y). And here I need not have any specific relation, I can choose my F(x) and F(y) arbitrarily. But this conversion may introduce a lot of quantization errors because of the repeated ADC and DAC stages. So, another is directly performing the sampling rate conversion entirely in the digital domain.

And this is possible only for these three types and in this module, I will restrict these three cases. So first let us understand what the two forms of frequency rate conversion are. So, either since f(x) is not equal to f(y), I can have either f(x) or f(y) that I would say is greater, or f(y) is less than f(x). If it is equal, then it is no longer multirate.

So here, when I am increasing the sampling rate, this process is called upsampling, and when I am decreasing the sampling rate, it is called downsampling. What does an upsampler do? So, the upsampler will insert zeros between samples, and in this way, it will increase the sampling rate. For example,

$$x[n] = \{3,\ 5,\ 2\} \text{ and } y[n] = [\uparrow 2]x[n]$$

$$\text{then } y[n] = \{3,\ 0,\ 5,\ 0,\ 2\}.$$

$$x[n] \longrightarrow \boxed{\uparrow 2} \longrightarrow y[n]$$

$$y[n] = \begin{cases} x\left[\frac{n}{2}\right], \text{for } n \text{ even} \\ 0, \text{for } n \text{ odd} \end{cases}$$

if this is my signal, x equals 3,5,2, and I want to upsample by a factor of 2. So, I want to double the sampling rate. So I will insert one 0 in between each sample of my x[n], and this is how it is represented symbolically; this is the equation where y[n] is x[n/2] for n even because this is an integer index.

For n odd, I would have the signal values be 0, and this is a graphical representation showing that I have this signal, which has three samples, and in the upsampled case, I insert two zeros here. So, what happens to the signal in the Z-domain? Let us try to figure it out. So, this is my signal, and this is my zeroth index. So, if I up-sample by 2, what I do is plug in zeros in between every sample. So, if I take the Z-transform, what happens to my x[n] ? This becomes my X(Z), which is this.

$$x[n] = \{\ldots,\ 3,\ \underline{5},\ 2,\ 9,\ 6,\ \ldots\}$$

$$y[n] = [\uparrow 2]\, x[n] = \{\ldots,\ 0,\ 3,\ 0,\ \underline{5},\ 0,\ 2,\ 0,\ 9,\ 0,\ 6,\ 0,\ \ldots\}$$

Taking to z-domain

$$X(z) = \cdots + 3\,z + 5 + 2\,z^{-1} + 9\,z^{-2} + 6\,z^{-3} + \cdots$$

$$Y(z) = \cdots + 3\,z^{2} + 5 + 2\,z^{-2} + 9\,z^{-4} + 6\,z^{-6} + \cdots$$

It is clear that

$$\boxed{Y(z) = \mathcal{Z}\{[\uparrow 2]\, x[n]\} = X(z^2)}$$

So, this is my 0, so no power of Z. Then my first component has Z^{-1} to the power of minus 1, Z^{-2}, and so on. What happens if I do to Y? Now, we have Y since there are 0s at the odd values of n, so here, the odd powers of z will be missing. Now, if you clearly compare these two, we see that the only change is that the power of z is doubled, see.

So I can conclude that if I upsample a signal by a factor of 2, my spectrum Y(Z) will be $X(z^2)$. We can obtain the same thing analytically, like this is just an example, but analytically you will get a very general idea. So this is my Z-transform of Y(Z), and what is Y(Z), or what is y(n)? y(n) will be x[n/2] for even n, right? And for odd values, the result is 0. Because I have plugged in zeros at my odd values of n, what I will do is

replace n with a dummy variable m. Here, n becomes 2m, and n/2 becomes m, which is nothing but the Z^2 square transform of x[m].

$$Y(z) = \sum_n y[n]z^{-n}$$

$$= \sum_{n\in\text{even}} x\left[\frac{n}{2}\right]z^{-n}$$

$$= \sum_m x[m]z^{-2m} = X(z^2)$$

So verified by both methods. So, what will happen to the Fourier transform when upsampling? So, we know that the Fourier transform, or DTFT in this case, is the Z transform evaluated at e raised to the power of j omega, or the unit circle. So, we know that the Z transform will be x of z squared. And plugging in z equals e to the power of j omega, I get this value, which means that this is nothing but twice this omega. So, my output spectra y of omega is nothing but x of twice omega, which means the spectra is squeezed to half if I upsample by a factor of 2. Now this is interesting, so you might be tempted to say that if this is my spectrum of X, then this should be my spectrum of X at 2 ω, and this is my spectrum of Y, correct? wrong.

$$Y(e^{j\omega}) = X(z^2)\big|_{z=e^{j\omega}}$$
$$= X((e^{j\omega})^2)$$

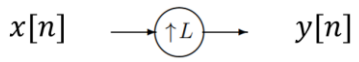
$$\boxed{Y^f(\omega) = \text{DTFT}\{[\uparrow 2]x[n]\} = X^f(2\omega).}$$

Why is it wrong? Now, see this is from π to $-\pi$, and this is just the period of the infinitely varying frequency. So, you can see that this spectrum is repeating after every 2 π. So, if you see this, only this fundamental spectrum, the fundamental period within $-\pi$ to π is not just squeezed. Everything, every infinite band, is squeezed. So, you have here, everything is squeezed.

So technically, you will get something like this, like this, like this, right? So hence, this is your actual spectrum because it will now be periodic from $-\pi/2$ to $\pi/2$. So, what I eventually get is that these are extra repeated copies, and this is called a spectral image, which is unwanted. Okay, so in general, if I do an n-fold upsampling, then what happens is my signal is represented as X of n by L, where n is a multiple of L because at other values of n, I am plugging in zeros. This is the symbolic representation of the upsampling

process, and this is the example. So, if I up-sample that signal, which I gave as an example, I am plugging in three zeros for upsampling by a factor of L.

$$y[n] = [\uparrow L]x[n] = x[\frac{n}{L}] \text{ when } n \text{ is a multiple of } L$$



$$y[n] = \begin{cases} x\left[\frac{n}{L}\right], \text{for } n \text{ multiple of } L \\ 0, \text{for } n \text{ otherwise} \end{cases}$$

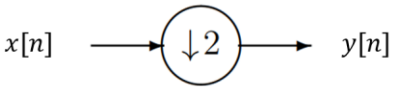
So, in general, if I upsample by L, I have to add L-1 zeros between two samples of my input signal. So, here are some remarks on upsampling. So, first of all, no information is lost in upsampling. Secondly, the upsampler introduces spectral images that are copies of the spectrum. So, in upsampling by 2, we have seen that one extra copy is replicated or one extra spectral image is formed.

And if I am sampling by L, then L-1 extra copies will be formed. Again, this is very important the system is linear, but it is not time-invariant because of this playing with the factor of n. Now, similarly, we will study downsampling. So, the down-sampler will keep every mth sample. For example, if this is my signal and this is my 0 index, I will keep every sample.

If I am downsampling by 2, I will skip every other sample and retain the rest. So, say there was a 4 here that was probably missed out. So, this is what I get. So, this is the symbol for downsampling, and this is also a representation of downsampling by a factor of 2. Now, if I see the signal in the Z domain just like we did for upsampling, we are discarding some alternate samples here if I am downsampling by 2, so this is my y[n].

$$x[n] = \{7, 3, 5, 2, 9, 4\},$$

$$y[n] = x[2n] = \{7, 5, 9, 4\}$$



$$y[n] = [\downarrow 2]x[n]$$

Now, clearly X(Z) is easily represented from this, so this is index 0, so 5, $2z^{-1}$, $9z^{-2}$,

and so on. Very clear and if we do the same for Y(Z), then this is my index 0, this is 9 z^{-1}, this is $4z^{-2}$. Great, that sounds so nice. Now how do I represent Y(Z) in terms of X(Z)? Because we have some extra terms like 3z and then this $2z^{-1}$.

$$x[n] = \{\ldots,\ 7,\ 3,\ \underline{5},\ 2,\ 9,\ 6,\ 4,\ \ldots\}$$

$$y[n] = [\downarrow 2]x[n] = \{\ldots,\ 7,\ \underline{5},\ 9,\ 4,\ \ldots\}$$

$$X(z) = \cdots + 7\,z^{2} + 3\,z + 5 + 2\,z^{-1} + 9\,z^{-2} + 6\,z^{-3} + 4\,z^{-4} + \cdots$$

$$Y(z) = \cdots + 7\,z + 5 + 9\,z^{-1} + 4\,z^{-2} + \cdots$$

So, all the odd powers of z do not appear in my Y(Z). So I need to delete those somehow. So, one way is to do an X(-Z), and that is what I have done here. So X-Z will negate all these values. So, this value, this value, this value, all these will be negated.

$$X(-z) = \cdots + 7\,z^{2} - 3\,z + 5 - 2\,z^{-1} + 9\,z^{-2} - 6\,z^{-3} + 4\,z^{-4} + \cdots$$

$$X(z) + X(-z) = 2 \cdot \left(\cdots + 7\,z^{2} + 5 + 9\,z^{-2} + 4\,z^{-4} + \cdots\right)$$

$$\boxed{X(z) + X(-z) = 2 \cdot Y(z^{2})}$$

And what happens when I negate this? Now, if I add them, I can easily cancel them, and that is what I have done. And by doing so, I am just retaining the even powers of z, and since I am adding them twice, it is two times this, which is $Y(Z^2)$. So now, how do I get Y(Z) back? So first of all, I bring these two into the denominator and then take the square root of z. So, Y(Z) will be-

$$\boxed{Y(z) = \mathcal{Z}\left\{[\downarrow 2]\ x[n]\right\} = \frac{X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})}{2}}$$

Now this may appear complicated, although the derivation is straightforward, but you will slowly get acquainted with this. Now that Z is done, what to do with the Fourier? Again, similarly, I will evaluate Fourier by giving the value Z equals

$$Z = e^{j\omega}$$

Which gives me this relation, and by doing the math, I get this and this, and by using the property, I finally come to the conclusion that my output spectrum is

$$\boxed{Y^{f}(\omega) = \mathrm{DTFT}\left\{[\downarrow 2]\,x(n)\right\} = \frac{1}{2} \cdot \left(X^{f}(\frac{\omega}{2}) + X^{f}(\frac{\omega - 2\pi}{2})\right)}$$

And what does it mean? The frequency spectra will be spread out; it will be enlarged like this. And that is what is happening. So, this is your original spectrum from $-\pi\ to + \pi$, and again, as I said, this is just a fundamental period; this is occurring infinitely many times.

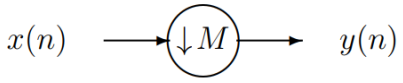
Now each will be spread out, but we know that this digital frequency has to be periodic within the span from $-\pi\ to + \pi$. So all this is outside this $-\pi\ to + \pi$ after the spread will be clipped off. And this is obvious because, suppose my signal bandwidth is say 100 and I have done something like sampling at 300, now this is safe because this is above Nyquist, which is 200. twice of this bandwidth and this is above Nyquist. But when I downsample by 2, I get 150, which is sub-Nyquist.

So, there will be aliasing, and this is a problem where we get, or we might get, aliasing. And this aliasing is a problem because you cannot recover the structure. So, this is a problem with downsampling. So, in up-sampling, we saw that we were getting extra copies; in downsampling, we are encountering the problem of aliasing. We might encounter; it is not always that we will encounter, but we might.

Like, say if my initial frequency was 1000 Hz and if I downsampled by 2, I would get a sampling frequency of 500 Hz, which is still above Nyquist. So, aliasing will not occur. Even after the spread, it will still be within the minus pi to pi range. So, in general, if I do an m-fold downsampling,

$$y[n] = [\downarrow M]x[n] \text{ keeps every } M\text{-th sample, discarding others.}$$

It will keep every mth sample and discard the m minus 1 samples, and this is the symbolic representation. So this is an example: if I do a 3-fold, this is 2; this will probably discard this and this, so this should be 4, and then it will discard this, this, and say if there is a value here, say 3, so here you will get a value of 3, and again this is 0, so it will discard 2.



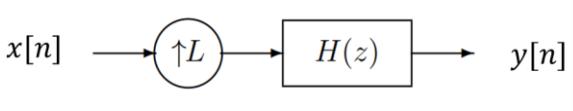
And here, say if we have a value, say 6, so you will get a value here, 6, like this. So, in general, whenever we are doing downsampling, information is lost because we are discarding samples. Now, how big that loss is a question. Now again, the down sampler is linear but not time invariant, just like the up sampler, and it may cause aliasing. Okay,

so how do we solve these two issues? So here we will learn two more concepts: interpolation and decimation.

So in general, say I have a signal like this, this, and this. Let me use a different color. So if I use green and say I have a 0 in between, So this is due to interpolation. So, initially, this looks like a signal that is not as good as the original stuff. So, if I do, I need some interpolation; I need some value that is not 0, which is some value somewhere in between.

$$y[n] = ([\uparrow L]x[n]) * h[n] = \Sigma_k\, x[k]h[n - Lk]$$

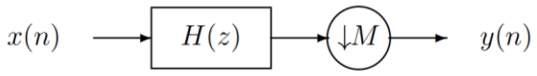
If I do a linear interpolation, let me pick another color like blue; if this value should be somewhere here and somewhere here, now this gives me a smoother estimate. That is interpolation. So how do you make a smoother estimate? I can always do a low-pass filtering because in smoothing in the time domain we studied what smoothing is, low-pass filtering removes sudden changes to zeros and produces a smoother value. So this can be done by using a low-pass filter. So, the low-pass filter should be before the upsampler or after the up-sampler.



It should be after the upsampler. Why? Because if you see the frequency domain, what it is doing is creating extra spectral images if this is the spectra. So, I need to remove this. And I have to retain this. So, I can create a low-pass filter that will just retain the main component, and you can see that after adding zeros, I am doing the low-pass filtering. So, this is called an interpolation that combines the upsampler with the low-pass filter.

So, this is an example: I have a signal that I upsampled by a factor of 2 by inserting 0s in between, and then I did interpolation, like I did low-pass filtering. The whole process is interpolation, and now you can see I get a much smoother response. Again, it depends on the design of the filter what kind of response I will get. Similarly, in decimation, we combine a low-pass filter with the downsampler. And here you should note that the low-pass filter is kept before the downsampler.

$$y[n] = [\downarrow M](x[n] * h[n]) = \Sigma_k\, x[k]h[Mn - k]$$



Why? Because the down sampler is spreading the spectra, it is causing aliasing. So, this is like an anti-aliasing filter. So, this anti-aliasing will restrict the frequency range of the original signal before the downsampling process so that information is not lost. So, this is what we are getting in the decimated system.

So, we have to do this before actually downsampling. This downsampling is, so first we will perform the decimation filtering, and then we will do the downsampling. So we now understand that

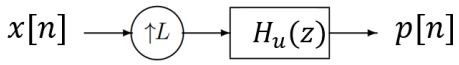
$$F_y = LF_x$$

$$F_y = \frac{1}{M}F_x$$

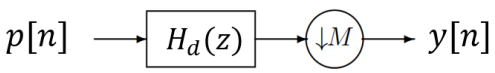
$$F_y = \frac{L}{M}F_x$$

if my F_y is some L times F_x, which is the case for my upsampling, and also the case where it is 1 by m times F_x, which is my downsampling, we have also covered interpolation and decimation. What about the fractional rate? Like if it is L divided by m, which is a rational number, how do you do it? So we can always first do an interpolation by L and then do a decimation by M, or we can do a decimation by M and then an interpolation by L. So I request you to pause for some time and think about which is a better strategy. So clearly you can think that if I do the interpolation first and then the decimation, this is a better strategy.

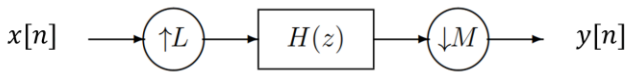
Why? Because in interpolation you are adding some samples and then discarding them. So the information loss is less. But if you do the decimation first, you are initially only discarding some samples and losing information before doing the interpolation. And this can also be efficiently defined by observing the block diagrams. So, this is an interpolator where we have an up sampler and then a low pass filter, which gives me an intermediate variable or signal P[n],



and this P[n], if I send it to another low pass filter and then the down sampler, which is basically the decimation system, I can get the output.



Now here, the good thing is that since this is a linear filter and they are in cascade, I can combine them into a single filter.



So I do not need to design two filters; I can get away with a single filter. So I will have an upsampler first, then this filtering; this filtering will do both things. That first, it will remove the spectral images of this upsampler, and it will also perform the anti-aliasing for the downsampler. And how do you design the cutoff frequency? So the cutoff is designed such that it will be the minimum of pi by L and pi by M.

$$\omega_o = \min\left\{\frac{\pi}{L}, \frac{\pi}{M}\right\}.$$

This design ensures that you both eliminate the spectral images and also avoid aliasing. So, thank you very much. We will meet again in the multi-rate signal processing lecture.