Experiments in Sampling, Reconstruction, and Filtering

KST, 4/2002

Introduction

This note describes some simple experiments in MATLAB to illustrate the sampling and reconstruction processes, and the implementation of filtering concepts.

The required theoretical background is primarily continuous-time systems, Fourier transform and a basic understanding of their discrete-time counterparts.

For the illustration of the theoretical principles, the various signals are compared in time and frequency domain. The corresponding sounds are also reproduced with MATLAB's "sound" function to provide a tangible understanding of the concepts.

(MATLAB commands appear in `Courier` font; the commands are MATLAB Version 6.1, or Release 12)

Experiment 1

Here we create a sinusoid with frequency 1 kHz and listen to the sound. We use a high sampling frequency of 44.1 kHz that represents a fairly good approximation of the continuous time signal.

Define the sampling frequency and the time vector. (8000 points / 44 kHz ~ 0.2 s worth of data)
Plot the output and look at the first 0.004s. Play the sound.
	`fs=44100; no_pts=8192;`
	`t=([1:no_pts]'-1)/fs;`
	`y1=sin(2*pi*1000*t);`
	`plot(t,y1);axis([0,.004,-1.2,1.2])`
	`disp('original');sound(y1,fs);`

Check the frequency domain signal. fr is the frequency vector and f1 is the magnitude of $F\{y1\}$.	`fr=([1:no_pts]'-1)/no_pts*fs; %in Hz`
	`fr=fr(1:no_pts/2);`
	`f1=abs(fft(y1));f1=f1(1:no_pts/2)/fs;`

F is the continuous time Fourier. (See derivation notes.) Compare the analytical continuous-time Fourier transform with its FFT computation.
 	
Notice the small amount of aliasing due to the fact that the truncated sinusoid is not bandlimited.
	`frp=fr*2*pi;tmax=max(t);`
	`F1=1/j*sin((frp-1000*2*pi)*tmax/2)`
	`.*exp(j*(frp-1000*2*pi)*tmax/2)`
	`./(frp-1000*2*pi);`
	`F2=1/j*sin((frp+1000*2*pi)*tmax/2)`
	`.*exp(j*(frp+1000*2*pi)*tmax/2)`
	`./(frp+1000*2*pi);`
	`F=abs(F1-F2);`
	`loglog(fr,F,fr,f1);pause`

Derivations*: Continuous Fourier transform interpretation of the Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT).*

The DFT is a discrete-time transform defined for a finite sequence of N numbers, as follows.

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j\frac{2\pi}{N}kn}$$

The term FFT is used to signify a specific method to compute the DFT in an efficient manner. While a transform in its own right, the DFT/FFT can be interpreted in terms of the CT-Fourier transform for a sampled signal *x(t)*. Let us consider a signal *x(t)*, sampled at the time instants *nT, n* = 0,1,…*N*-1, and suppose that the signal is 0 outside the sampled interval [0, (N-1)T]. The Fourier transform of this signal is

$$X(jw)=\int_{-\infty}^{\infty}x(t)e^{-jwt}dt=\int_{0}^{(N-1)T}x(t)e^{-jwt}dt$$

First, in a naïve approach, let us approximate the integral by an Euler discretization, whereby the integrand is taken as piecewise constant.

$$X(jw) = \int_0^{(N-1)T} x(t)e^{-jwt}\,dt \cong \sum_{n=0}^{N-1} x(nT)e^{-jwnT}\,T$$

Let us consider now a sampled version of the Fourier transform at the frequencies $w = kw_0 = k\dfrac{2\pi}{TN}$, where k=0,1,…,N-1. Then,

$$X(jkw_0) \cong \sum_{n=0}^{N-1} x(nT)e^{-j\frac{k2\pi}{NT}nT}\,T = T\sum_{n=0}^{N-1} x(nT)e^{-j\frac{2\pi}{N}kn}$$

Thus,

$$F\{x\}\,|_{w=kw_0} \cong FFT\{x(nT)\}T$$

This approximation is valid as long as the integrand approximation by a piecewise constant function is reasonable. This implies that k must be small and *x(t)* should not change significantly inside any sampling interval of length *T*.

In a more precise formulation, the sampled signal has a Fourier transform

$$X_s(jw) = \frac{1}{2\pi}X(jw) * \frac{2\pi}{T}\sum_k \delta(w - kw_s) = \frac{1}{T}\sum_k X(j(w - kw_s))$$

where, as usual, $w_s = \dfrac{2\pi}{T}$. This is the familiar form of the sampled signal Fourier transform as a summation of shifted replicas of the signal Fourier transform. Of course, aliasing effects will occur if *X(jw)* extends beyond half the sampling frequency. On the other hand, using the Fourier transform definition on the sampled signal we find

$$\begin{aligned} X_s(jw) &= F\left\{x(t)\sum_k \delta(t-nT)\right\} = \int\left[x(t)\sum_n \delta(t-nT)\right]e^{-jwt}\,dt \\ &= \int\left[\sum_n x(nT)\delta(t-nT)e^{-jwnT}\right]dt = \sum_n x(nT)e^{-jwnT}\int\delta(t-nT)dt \\ &= \sum_n x(nT)e^{-jwnT} \end{aligned}$$

Again, evaluating this transform at a discrete set of frequencies $w = kw_0 = k\dfrac{2\pi}{TN}$, we get

$$X_s\left(jk\frac{2\pi}{TN}\right) = \sum_n x(nT)e^{-j\frac{2\pi}{N}kn} = FFT\{x(nT)\}$$

In other words, the FFT of the sampled signal is equal to the Fourier transform of the sampled signal evaluated at the frequencies kw_0, in the interval $[0, w_s]$. This means that the FFT will be symmetric about the point $w_s/2$. It will also be approximately equal to $X(jw)/T$, (at the corresponding discrete frequencies in the interval $[0, w_s/2]$) as long as any aliasing effects are small.

(Note: The symmetry of the FFT is the reason why only the first half of the FFT points are shown in the plots.)

More Derivations: *The Fourier transform of the signal under consideration.*
In this experiment we defined the signal $y(t) = \sin(w_0 t)$, for t in $[0, T_{max}]$. Consequently,

$$F\{\sin(w_0 t)\, pulse_{[0,T_{\max}]}(t)\} = \frac{1}{2\pi}\frac{\pi}{j}[\delta(w-w_0)-\delta(w+w_0)] * \left[\frac{2\sin(wT_{\max}/2)}{w} e^{-jwT_{\max}/2}\right]$$

$$= \frac{1}{j}\left[\frac{\sin((w-w_0)T_{\max}/2)}{w-w_0} e^{-j(w-w_0)T_{\max}/2} - \frac{\sin((w+w_0)T_{\max}/2)}{w+w_0} e^{-j(w+w_0)T_{\max}/2}\right]$$

This expression is used to compare the FFT-computed transform that may (does) include aliasing effects with its theoretical value as the sampling interval approaches zero.

Experiment 2

In this experiment we sample the same sinusoid (1 kHz) at a lower sampling rate to study the aliasing effects.

Sample the sin at 44/4 kHz.
Compare the two output and play the sound. This may still sound OK because of the internal filtering of the soundcard.
	`a=4; t_a=([1:no_pts/a]'-1)/fs*a;`
	`y_a=sin(2*pi*1000*t_a);`
	`plot(t,y1,t_a,y_a);`
	`axis([0,0.004,-1.2,1.2]);`
	`sound(y_a,fs/a);`

Check the frequency domain signal.
Notice that the replicas of $F\{y\_a\}$ are now in the audible range (not shown).
Also, the aliasing effects are in general more pronounced.
	`fr_a=([1:no_pts/a]'-1)/no_pts*a*fs/a;`
	`fr_a=fr_a(1:no_pts/a/2);`
	`f_a=abs(fft(y_a));`
	`f_a=f_a(1:no_pts/a/2)/fs*a;`
	`loglog(fr,F,fr,f1,fr_a,f_a);`

Use the interp1 function to get the ZOH version of the signal at 44 kHz.
The ZOH version is how the signal would sound with a 44/a kHz D/A converter.
Its difference from the original is significant in both the time and frequency domain and its reproduction is quite poor.
	`y_n =`
	`interp1(t_a,y_a,t,'nearest','extrap');`
	`f_n=abs(fft(y_n));f_n=f_n(1:no_pts/2)/fs;`
	`plot(t,y1,t,y_n);`
	`disp('original');sound(y1,fs);`
	`disp('nearest');sound(y_n,fs);`
	`loglog(fr,F,fr,f1,fr,f_n);`

Experiment 3

Here we use different digital filters to "interpolate" the values of the low rate signal and convert it to a high rate signal. This does not overcome any aliasing effects that occurred at sampling but improves the reproduction by reducing the undesirable properties of the low rate ZOH. (It does require a better D/A converter!) In the following table we concentrate on the implementation of a digital filter by approximating the impulse response of an ideal low-pass.

Generate an "Upsampled" version of the low-rate signal at 44 kHz.
The upsampled signal contains several replicas of the original Fourier transform within the sampling frequency and it is not an approximation of the original signal.
	`y_u=y1*0;k=1:a:no_pts;y_u(k)=y_a;`
	`f_u=abs(fft(y_u));`
	`f_u=f_u(1:no_pts/2)/fs*a;`
	`plot(t,y1,t,y_u);axis([0,0.004,-1.2,1.2])`

However, the original signal can be "recovered" after low-pass filtering. Here, the filter is defined in terms of its impulse response and the output is computed as a convolution sum. Notice that the impulse response HS is 2000 points long yielding	`hs=sin(3000*2*pi*t)/pi./t; i=1000:-1:2;`
	`HS=[hs(i);3000*2;hs(2:1000)]*a/fs;`
	`y_f=conv(HS,y_u);`
	`y_f=y_f(1000:999+no_pts);`
	`f_f=abs(fft(y_f));f_f=f_f(1:no_pts/2)/fs;`
	`plot(t,y1,t,y_f);axis([0,0.004,-1.2,1.2])`

an 1000 points delay in the sound reproduction. This is significant but not necessarily impractical.	`disp('original');sound(y1,fs);`
	`disp('filtered-upsampled');sound(y_f,fs);`
	`loglog(fr,F,fr,f1,fr,f_f);`

Notes: Given a discrete-time signal *y(n)*, the upsampled signal (by *a*) is defined as the signal *z(n)* such that

$$z(n) = \begin{cases} y(k) & if\ \ n = ak \\ 0 & otherwise \end{cases}$$

Equivalently, we may think of *z(n)* as the signal produced by sampling the continuous time signal *y(t)* with the impulse train $p_u(t) = \sum b_n \delta(t - n\frac{T}{a})$, where $b_n = 1$, if *n/a* is an integer, and 0 otherwise.

Clearly, the upsampling impulse train is identical with the original delta train $p(t) = \sum \delta(t - nT)$ and, therefore, the continuous-time sampled signals *y(t)p(t)* and $y(t)p_u(t)$ are the same, and have the same Fourier transforms. The latter, however, has a smaller sampling time (larger sampling frequency), implying that the FFT of the discrete-time sequence will contain several replicas of *F{y}*. The upsampling process does not create any new information about the time signal but allows for the implementation of better low-pass filters in discrete time.

Experiment 4

Equalizers are often used in audio equipment to amplify (or attenuate) frequency bands in an effort to improve the quality of reproduction of the original signal. Their objective is to cancel (or invert) the audio signal distortion caused by equipment limitations (amplifier, speaker, media) or the environment where the signal is reproduced.

In this experiment we use digital low-pass filters to implement a "frequency equalizer." The test signal is the standard Windows sound "Tada." The attached Matlab program loads the sound and filters it with three different low-pass filters. Then, by taking different linear combinations of the resulting signals we can amplify or attenuate the energy of the signal in the desired frequency band(s).