SCHOOL *of* ENGINEERING and ELECTRONICS
MSc in Signal Processing and Communications

Digital Signal Processing Labs (PGEE11108)

Laboratory Session 2:

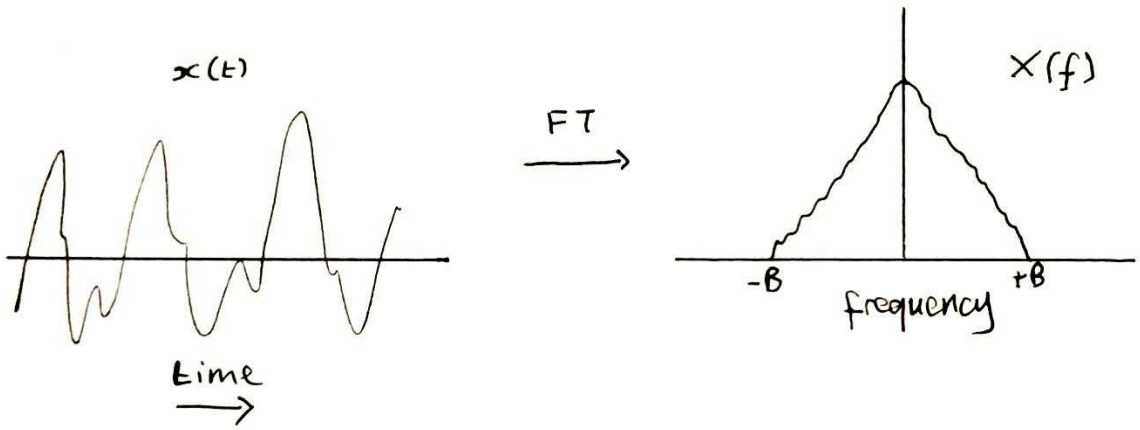
Signal Generation, Sampling and Aliasing

Introduction

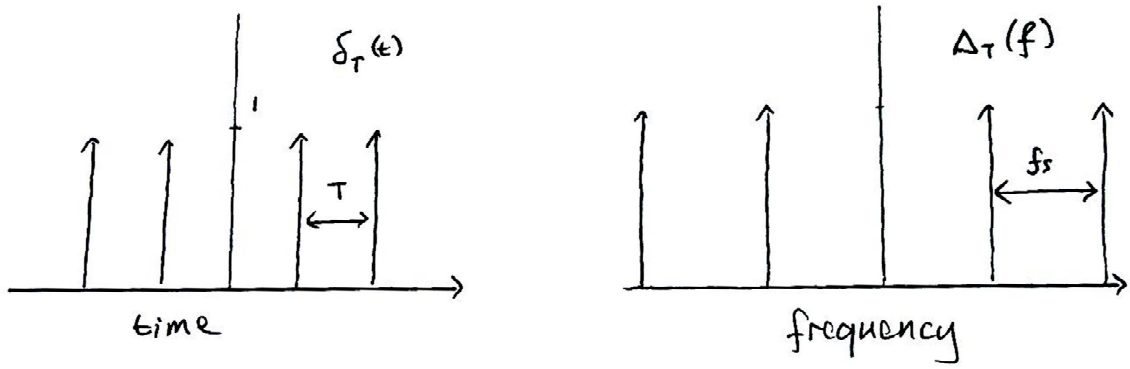
The objective of this laboratory is to illustrate the basic principles of sampling; aliasing; and reconstruction. The lab will get you to generate sine wave signals and sample them. Different reconstruction filters will be considered. The lab also examines the representation of these signals in the frequency domain.

Sampling Theory Refresher

Recall that a continuous time signal can be represented in either the time domain or the Fourier domain, the latter through the Fourier Transform (FT):

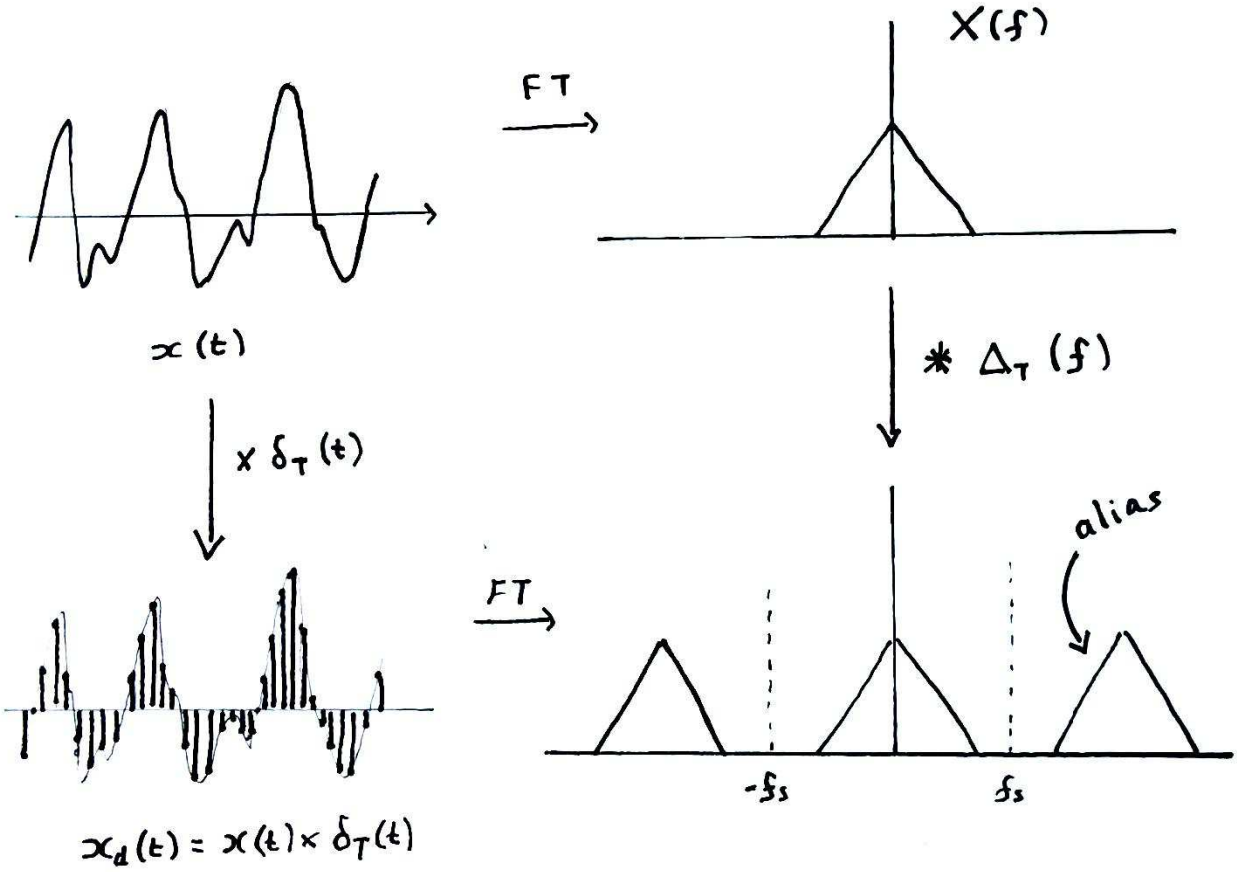


The process of sampling can be viewed as multiplication of the original signal $x(t)$ by a spike train $\delta_T(t)$ with a spike spacing of the sampling period, T. $\delta_T(t)$, looks something like:



As with $x(t)$ we can also represent $\delta_T(t)$ in the frequency domain as shown above. Now the spikes are spaced according to the sampling frequency, $f_s = 1/T$.

Using that fact that multiplication in the time domain is equivalent to convolution in the frequency domain we can determine what the FT of the sampled spike train, $x_d(t) = x(t) \times \delta_T(t)$, looks like:



The replica FT components are called 'aliases' and can be removed by an ideal low pass filter if: (1) $x(t)$ is bandlimited with bandwidth B; and (2) the sampling frequency, $f_s \geq 2B$. Otherwise the aliasing cannot be removed by filtering. In practice physically realizable filters are non-ideal (e.g. linear interpolation) and will not fully remove aliasing.

Experiments:
The following experiments will investigate signal generation, sampling and reconstruction. We will look at this in both the time domain and the frequency domain.

1. Signal Generation & Sampling

An extract of a sinusoid signal can be generated as follows:

```
% Generates a 40Hz cosine wave
%
N = 64;                 % 64 data points
fs = 1000;              % 1kHz sampling frequency
T = 1/fs;               % sample period
f = 40;                 % signal frequency in Hz
n = 0:N-1;              % discrete time vector

x = cos(2*pi*f*n*T);    %signal definition

%plot the signal against time
```

```
stem(n*T,x);
title('40Hz cosine')
xlabel('Time')
ylabel('x(nT)')
```

1.1. Enter the code in a script file named **sineGen.m** and verify that what you get on the plot really is a sampled 40Hz cosine wave. You should be able to identify the outline of the sinusoid.

1.2. Now plot the signal using the `plot` function. This connects the points with a straight line and can be considered to be a form of "signal reconstruction". It is not optimal but it is sufficient here.

1.3. Modify the code to generate sampled cosines at *signal* frequencies of 200Hz, 400Hz, 600Hz and 800Hz, with durations of **1 second** but still a sampling frequency, `fs`, of 1kHz.

In each case, generate plots of the *first 100 msec* of the signals using `stem` and `plot` on the same figure.

1.4. Add to the plots in 1.3 plots of the signals sampled at 10000Hz (this is highly over sampled and can be viewed as a good approximation to the analogue signal).

You can also 'listen' to the signal using the command:

```
soundsc(interp(x,8),8*fs);
```

This plays the signal 'x' assuming a sampling frequency of `fs` (the code actually interpolates the signal and uses a sampling frequency of `8*fs)`. Can you explain what you hear?

2. Reconstruction filters

DSP also involves reconstructing analogue signals from discrete samples. Above we have used linear interpolation to do this. It is important to realise that signal reconstruction is an instance of curve fitting. Given a finite set of points there are many possible curves (analogue signals) that can pass through the samples – there is no right answer!

General Interpolation: Signal reconstruction is typically achieved using a reconstruction filter. That is the interpolation can be written as an analog filtering operation (convolution) of the **sampled signal spike train,** $x_d(t)$:

$$\hat{x}(t) = \int_{-\infty}^{\infty} x_d(\tau)h(t-\tau)d\tau = \sum_{n=-\infty}^{\infty} x(t_n)h(t-nT)$$

where $h(t)$ is the interpolation filter, the sample values, $x(t_n)$, are taken at time $t_n = nT$, with T the sampling period, and the **sampled signal spike train** is the continuous time domain signal defined as:

$$x_d(t) = \begin{cases} x(nT) & \text{for } t = nT \\ 0 & \text{otherwise} \end{cases}$$

In this lab we will consider two important forms of interpolation - we will compare the 'ideal' low pass filter reconstruction with linear interpolation.

Linear interpolation: The plot command in 1.2 linearly interpolates between samples. This formula can be explicitly written as:

$$\hat{x}_l(t) = \left(1 - \frac{(t-nT)}{T}\right)x(t_n) + \left(\frac{t-nT}{T}\right)x(t_{n+1}), \text{ for } t_n \le t \le t_{n+1}$$

$$= \sum_{n=-\infty}^{\infty} x(t_n)h(t-nT)$$

Note in the last line that we have written this as a discrete convolution. For linear interpolation the interpolating filter is defined as:

$$h(t) = \begin{cases} \dfrac{T-|t|}{T} & \text{for } -T \le t \le T \\ 0 & \text{otherwise} \end{cases}$$

You should convince yourself that this is the convolution with this filter is equivalent to the linear interpolation formula given above.

'Ideal' Low pass filtering: ideal low pass filtering is often presented as the 'ideal' signal reconstruction. In the frequency domain this operation corresponds to multiplication of the spectrum of a signal by a band limited rectangular function. In the time domain this corresponds to filtering the sampled signal spike train with a sinc function. The reconstructed signal still has the formula:

$$x_r(t) = \sum_{n=-\infty}^{\infty} x(t_n)h(t-nT)$$

But now the filter is the sinc function:

$$h(t) = \frac{\sin(\pi t/T)}{\pi t/T}$$

Again, make sure that you fully understand where these equations come from.

Note that both the linear interpolation filter and the sinc function are acausal filters. Furthermore the sinc filter has an infinite impulse response. These facts will make the following exercise challenging.

2.1. Modify your code from section 1.4 to generate 3 new continuous time domain signals: the continuous time *sampled signal spike train*; the *linearly interpolated reconstruction signal*; and the 'ideal' *low pass filter reconstructed signal*.

[It is not possible to represent continuous time signals in MATLAB therefore use a discrete time signal sampled at a sufficiently high rate - 10000Hz - as before. It is also impossible to implement ideal low pass filters. Use a truncated sinc function (MATLAB `sinc` function – truncate to ±10T). Care will also be needed in dealing with the non-causal properties of the filters.]

Generate plots of the three new signals.

You can again listen to these signals to determine what is happening in terms of frequency – we will look at this next.

3. Frequency domain view of sampling

When a continuous time signal is sampled it can incur aliasing because regions of the frequency domain are shifted ('folded') by an amount equal to the sampling frequency. In MATLAB we can simulate this effect using our highly over sampled signals to represent continuous time. It will also be necessary to simulate the Fourier transform of analogue signals. This can be roughly done with the following code:

```
function fmagplot(xa,dt)
% FMAGPLOT
```

```
% fmagplot(xa,dt)
%
%   xa:     the "analog" signal
%   dt:     the sampling interval in seconds for the simulated
%           analog signal xa
%
L = length(xa);
Nfft = 2.^ceil(log2(L));        % Choose the nearest Power of 2
Xa = fft(xa,Nfft);
range = 0:(Nfft/4);             % show frequencies up to 1/4
                                % sampling rate
ff = range/Nfft/dt/1000;        % frequencies in kHz

% plot magnitude frequency

plot(ff,abs(Xa(1+range)))
title('Continuous-time Fourier transform (MAG)');
xlabel('FREQUENCY (kHz)');
grid
```

Modify your code to produce a new figure that plots the Fourier transform of (a) the analog signal, (b) the sample signal spike train, (c) the ideal LP filter reconstruction and (d) the linear interpolation reconstructed signal. Run this for the range of frequencies as before.

Can you explain the frequency content of each of the signals?