

Problem Set #1

Problem Set Due: Friday, April 12

In the following problems, assume that

$$\delta_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

is an infinite array of impulses separated by a time T . This is a shorthand notation for the sampling function. It's Fourier transform is

$$\mathcal{F}\{\delta_T(t)\} = \omega_s \delta_{\omega_s}(\omega)$$

where $\omega_s = \frac{2\pi}{T}$ is the sampling frequency. This is an array of impulses in frequency, separated by the sampling frequency.

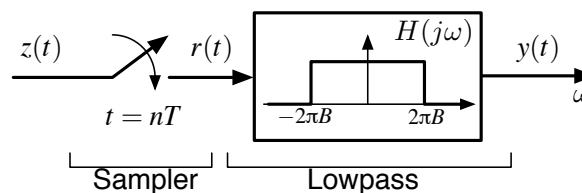
1. Demodulation with an Ideal Sampler

Multiplying a signal $f(t)$ with $\cos(\omega_c t)$ produces a modulated signal

$$z(t) = f(t) \cos(\omega_c t)$$

where ω_c is the carrier frequency. One way to demodulate this signal and recover $f(t)$ is to multiply $z(t)$ by $\cos(\omega_c t)$, and lowpass filter the result.

In this problem you will show that you can achieve the same effect with an ideal sampler. The block diagram of the receiver is



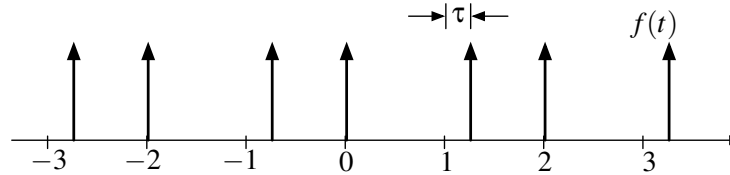
where the ideal sampler is drawn as a switch that closes instantaneously every T seconds to acquire a new sample. Assume that the signal $f(t)$ is bandlimited, with $F(j\omega) = 0$ for $|\omega| > 2\pi B$, and that the carrier frequency $\omega_c \gg 2\pi B$. Other than that $F(j\omega)$ is unspecified (plot it as a bandlimited triangle in the plots below).

- Show that we can recover $f(t)$ if the ideal sampler operates at a frequency ω_c (i.e. samples at a rate of $\omega_c/2\pi$ samples/s). Draw the spectrum of the input signal $Z(j\omega)$, and the spectrum of the signal right before the lowpass filter $R(j\omega)$.

- (b) What is the lowest frequency at which the sampler can operate, and still recover $f(t)$?
Draw the spectrum of $R(j\omega)$.

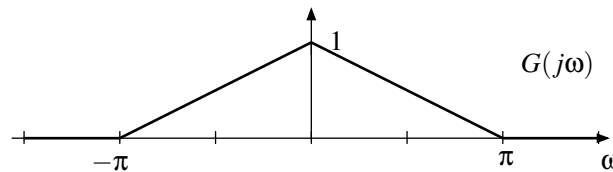
2. Sampling Timing Errors

Imperfections in a sampler cause characteristic artifacts in the sampled signal. In this problem we will look at the case where the sample timing is non-uniform, as shown below



The sampling function $f(t)$ has its odd samples delayed by a small time τ .

- Write an expression for $f(t)$ in terms of two uniformly spaced sampling functions.
- Find $F(j\omega)$, the Fourier transform of $f(t)$. Express the impulse trains as sums, and simplify.
- Find $F(j\omega)$ for the case where $\tau = 0$, and show that this is what you expect.
- Assume the signal we are sampling has a Fourier transform



Sketch the Fourier transform of the sampled signal. Include the baseband replica, and the replicas at $\omega = \pm\pi$. Assume that τ is small, so that $e^{j\omega\tau} \simeq 1 + j\omega\tau$.

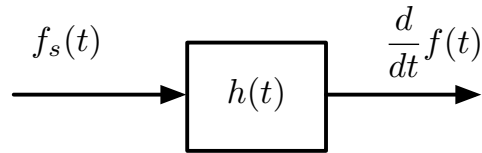
- If we know $g(t)$ is real and even, can we recover $g(t)$ from the non-uniform samples $g(t)f(t)$?
- Recall how we argued that the sinc is a perfect interpolator for a uniformly sampled signal at the Nyquist rate. The sinc is one at the sample, and zero at all of the other samples. There are also perfect interpolators for this case. Sketch what they would look like. You can actually solve for them explicitly, but you don't need to do that. We are just looking for qualitative answers here.

3. Reconstruction of the Derivative of a Signal

- Assume we have a sampled signal

$$f_s(t) = f(t)\delta_1(t)$$

where $f(t)$ is bandlimited, and is sampled exactly at the Nyquist rate which is 1 sample/second. Find the transfer function $H(j\omega)$ of a system that takes the sampled signal, and reconstructs the *derivative* of the original signal.

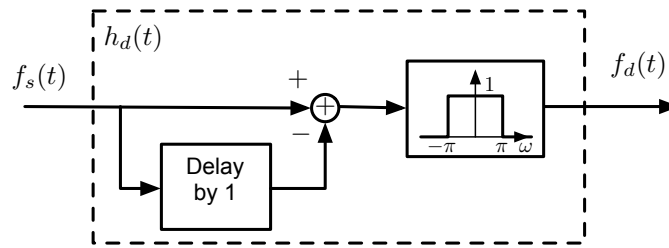


Hint: Think of this as two systems, one to reconstruct the signal, and a second to take the derivative. Then combine them.

- b) Find the impulse response $h(t)$ for this system. You don't have to explicitly differentiate in your answer.
- c) An approximate solution is to note that

$$f'(t) \simeq \frac{1}{T} (f(t) - f(t - T))$$

If we let $T = 1$, our sampling period, we can just subtract adjacent samples, and then reconstruct the result with an ideal lowpass filter.



Here $f_d(t)$ is the approximation to the derivative of $f(t)$, and $h_d(t)$ is the impulse response of this system. Find the transfer function $H_d(j\omega)$.

- d) Find the impulse response $h_d(t)$.
- e) The transfer functions of these two systems differ. Use the small angle approximation

$$e^{-j\omega} = \cos \omega - j \sin \omega \simeq 1 - j\omega$$

to show how these two are related.

Laboratory

In this lab we will look at some effects of signal aliasing, and one of the reasons for using an antialiasing filter.

Task 1: *Aliasing*

In this task we will examine the property of discrete time sinusoids with frequencies above the Nyquist rate to appear as lower frequencies. We will use a signal that has quadratic phase, sampled at 16384 Hz.

```
>> t = [0:16383]/16384;
>> y = cos(2*pi*4096*(t.*t))
```

A continuous time signal with a quadratic phase as a function of time is known as a “chirp”, and has a linear sweep in frequency. Listen to `y` using

```
>> sound(y, 16384)
```

Does is sound like you expect? Draw a plot of frequency verses time. Frequency is the derivative of the phase. If $f(t)$ is a continuous sinusoid $\cos(\theta(t))$, the frequency is

$$f(t) = \frac{1}{2\pi} \frac{d\theta(t)}{dt}$$

where $f(t)$ is in Hz, and $\theta(t)$ is in radians. In this case the phase is $\theta(t) = 2\pi(4096)t^2$ radians. Note that since this is a real cosine, there will be positive and negative frequencies.

Now, make a subsampled version (downsampled by a factor of 2)

```
>> y2 = y(1:2:end);
```

and listen to it.

```
>> sound(y2, 8192)
```

Does this sound like you expect? Why is this different than the previous case? Again, draw a plot frequency verses time.

Task 2: *Anti-Aliasing Filter*

As we described in class, it is common practice to put a lowpass filter right before a sampler, to ensure that no aliasing occurs. This is also highly recommended when downsampling a signal, as will be demonstrated here.

Conceptually, downsampling a signal by a factor of N simply means taking every N 'th sample. For example, assume we have a sinusoid of frequency 500 Hz, sampled at 256 kHz. We'd like to reduce the sampling rate to 8 kHz (typical land-line phone quality), or a factor of $N=32$,

```
>> t = [1:256000]/256000;
>> x = cos(2*pi*500*t);
```

where x will be one second of the sinusoid. In practice, the signal will also contain broadband noise that covers the spectrum

```
xn = 0.5 *x + 0.125* randn(1,256000) ;
```

where the scale factors are chosen so that the result will be between -1 and 1, so it doesn't clip when played with `sound()`.

If we are going to downsample by a factor of $N=32$ we should lowpass filter first. A possible filter is a truncated (time limited) sinc. Next week we'll talk about better options, but this is a reasonable choice. How do we design this sinc? The key is to think about the sinc interpolators from Wednesday's class. These were 1 at the sample, and zero at the other samples. That is what we want here. The sinc zeros should be spaced by 32 samples. There are tradeoffs with different filter lengths that we will also talk about next week. However, four zeros on either side is a reasonable choice. Then

```
>> th = [-32*4:32*4]/32;
>> h = sinc(th);
>> h = h/sum(h);
```

Plot this, and check that it does what you want it to do (zero spacing, number of samples). The last step sets the DC gain to 1, so that filtered and unfiltered signals will have the sample amplitude, and can be compared. The sinc value at time zero will not be 1.

If you don't have the signal processing toolbox, you don't have the `sinc()` function. In this case, define an m-file

```
function [y] = mysinc(t)
% sinc(t) = sin(pi*t)./(pi*t)

t(find(t==0)) = eps;
y = sin(pi*t)./(pi*t);
```

This avoids divide by zero by replacing zero with the machine precision `eps`, which is the closest number to zero that it can represent.

The downsampled signals are

```
>> xnd = xn(1:32:end);
```

if we don't use an anti-aliasing filter `h` and

```
>> xnh = conv(xn,h,'same');
>> xnhd = xnh(1:32:end);
```

The matlab function `conv()` does the convolution, and the argument 'same' makes the length of the result equal to the length of the first argument. This eliminates the time delay introduced by the convolution, making it easier to compare the two signals.

Compare `xnd` and `xnhd`. You can compare a segment of each on the same plot, plot the difference, look at the mean squared error between each of these and the ideal signal,

```
xi = 0.5*x[1:32:end]
mse1 = sum((xnd - xi).^2)/8000;
mse2 = sum((xnhd - xi).^2)/8000;
```

or listen to how they sound

```
>> sound(xnd,8000);
>> sound(xnhd,8000);
```

Describe what the anti-aliasing filter has done for the quality of the reconstruction.

For future reference, matlab has a function `decimate()` that designs and applies a filter, and then subsamples a signal. The 'FIR' option is recommended

```
>> xd = decimate(x,32,'FIR')
```

They actually recommend you don't decimate by more than 13, so you'd first decimate by 8, and then 4.