ENEE 425

Section 0101 Fall 2004

Homework #3

You must attach a printed version of Matlab code (a Matlab file is preferable to copying and pasting a session, if possible). You may put several problems into a single printout, if the start of each problem is made obvious.

Don't forget to answer the verbal questions too, e.g. (4d).

This problem demonstrates how to emulate sampling from within Matlab, which is a bit tricky, because sampling is continuous-to-discrete conversion, but Matlab itself is all discrete. We'll therefore need to use ***discrete emulations*** *of: the* ***continuous*** *signal, the* ***continuous****-to-discrete conversion, and the discrete-to-****continuous*** *reconstruction. Those three steps must be emulated in Matlab code that may not be self-evident, but I give you the exact code you need : the Matlab code immediately below emulates the continuous signal; the Matlab code immediately before (4c) samples that continuous signal; the Matlab code immediately before (4j) reconstructs a continuous signal from the discrete signal.*

Please do not get confused between the ***emulation****, which is not crucial to understanding sampling, and the* ***analysis of the sampled signal****, which is crucial to understanding sampling. The goal of this homework is to see what sampled signals look like in both the time and Fourier domains, and how they are related.*

In this homework we will sample the continuous (analog) function $x_c(t) = \cos(2\pi f_0 t)$ where $f_0 = 0.1875$ Hz.

Emulation

First emulate continuous *t* and *x*(*t*). On a computer we cannot let *t* be continuous and infinite in extent, but we can give it a fine (but not continuous) granularity and a longish (but not infinite) extent of 64 seconds.

```
% emulate continuous time
tmax = 64; % seconds
granularity = 2^(-11);
t = granularity*[0:tmax/granularity-1];% emulated continuous t in seconds
```

(Don't forget the last semicolon ";". or you will print all 2^{11} values of `t`.) This gives us 64 seconds of finely granulated time, starting at 0. (You can see the first and last few points of `t` by examining, e.g., `t(1:4)` and `t(end-3:end)`.)

Then we emulate the continuous (analog) function $x_c(t)$:

```
xc = cos(2*pi*0.1875*t); % emulate the continuous signal xc(t)
```

1) What is the highest frequency (in Hz) in this signal?
2) What is the maximum sample time *T* allowed without distortion from aliasing?
3) Plot $x_c(t)$ using `plot()` (since it is a "continuous" function). Make sure you get the correct values of *t* along the *t* axis. Fix the axis limits (for comparison with later results) using `axis([0 tmax min(xc) max(xc)])`. Give it a title using `title()`.

4) Repeat everything below (a – j) for these 3 values of T: `1 2 4` (in seconds).

a) What is the sampling frequency in Hz?

b) What is half the sampling frequency in Hz?

Emulation

Use the following Matlab code to emulate sampling. [It only works for granularities that are powers of 2 and sample times that are reciprocals of powers of 2 (in seconds).] It makes n from t, by first picking only the values of t which equal nT, and then making the array n that has the same length:

```
% the following 4 lines emulate sampling xc(t) to get x[n]
nFromt = find( abs(t/T-round(t/T)) < eps); % map n onto t
n = 0:length(nFromt)-1; % define n
N = length(n); % the number of discrete points we emulate
x = xc(nFromt); % discretize xc(t) and call it x[n]
```

c) What are the upper and lower values of `n`? what is the length of `n`?

Open up a new figure (for each value of T). You will draw 6 `subplots` in the figure.

d) In the first subplot, plot $x[n]$. Use `stem()` or `bar()` since this is a discrete sequence. Make sure you get the correct values of n along the n axis. Fix the axis limits (for comparison with other results) using `axis([0 n(end) min(xc) max(xc)])`. Give it a title using `title()`. Does it "look like" the plot of $x_c(t)$ in (c) above (this is a subjective judgment)? Do you think that the original signal can be re-constructed by this sampling?

Define two Fourier space variables, ω and f where $\omega = 2\pi Tf$: ω, as commonly used in class and in the textbook, has units of radians and repeats with period 2π . f, as commonly used in the real world, has units of Hz, but also repeats (as it must since it is defined in terms of ω).

```
omega = 2*pi*n/N; % radian
freq = omega/T/(2*pi); % Hz
```

e) What is the period of f? If values of ω between π and 2π correspond to negative frequencies between $-\pi$ and 0 (by periodicity), which values of f correspond to analogous values of negative frequencies?

We find the (discretized version of the) Fourier Transform of $x[n]$ i.e. $X\left(e^{j\omega}\right)$ by

```
X = fft(x); % note: X is not the same as x, and X is complex
```

f) In a subplot, plot $\left|X\left(e^{j\omega}\right)\right|$, which in Matlab is called `abs(X)` and should be plotted as a function of `omega`. Use `stem()` or `bar()` since this is a discrete sequence. Make sure you get the correct values of ω along the ω axis. Fix the axis limits (for comparison with other results) using

`axis([0 omega(end) 0 max(abs(X))])`. Give it a title using `title()`. Note that inside `title()`, you can use the expression `\omega` and it will be displayed as ω.

g) In a subplot, plot `abs(X)` as a function of `freq`. Use `stem()` or `bar()` since this is a discrete sequence.. Fix the axis limits using `axis([0 freq(end) 0 max(abs(X))])`. Give it a title using `title()`. Where are the peaks? What frequencies do they fall at? Where should they fall at? Interpreting frequencies above the half-way point as negative, where do the negative frequencies fall at? How is this consistent with what you know about the original function? This demonstrates that using ω is really useful from the theoretical perspective, but `freq` is **really** useful when interpreting and debugging plots.

Create an ideal low pass filter with cutoff at $\omega = \pi$ and gain T:

```
LP = T*[1,ones(1,N/2-1),0,ones(1,N/2-1)];
```

(recall any frequencies above $N/2$ can be interpreted as negative frequencies.)

h) This filter lets through almost all frequencies. In a subplot, plot `LP` as a function of `omega`. Use `stem()` or `bar()` since this is a discrete sequence.. Fix the axis limits using `axis([0 omega(end) 0 max(LP)])`. Give it a title using `title()`.Which frequencies are not passed in ω? In f? How does this compare to your answer to (ii)?

Apply the ideal low pass filter to X:

```
XLP = X.*LP;
```

i) In a subplot, plot the magnitude of `XLP` as a function of `omega`. Use `stem()` or `bar()` since this is a discrete sequence. Fix the axis limits using `axis([0 omega(end) 0 max(abs(XLP))])`. Give it a title using `title()`. Why is `XLP` made by multiplying `X` and `LP` instead of convolving them?

Emulation

The reconstruction of the analog signal, $x_r(t)$, is emulated in Matlab by:

```
xr = interpft(real(ifft(XLP))/T,length(t)); % reconstructed signal
```

j) In the last subplot, plot $x_r(t)$ using `plot()` (since it is a "continuous" function). Make sure you get the correct values of t along the t axis. Fix the axis limits (for comparison with figure 1) using `axis([0 tmax min(xc) max(xc)])`. Give it a title using `title()`. Does it reconstruct the original signal? Should it? Why or why not?