Second Year Engineering Mathematics Laboratory

Michaelmas Term 1998

-M L G Oldfield 30 September, 1999

Exercise 1: Fourier Series & Transforms *Revision 4*

Answer all parts of Section A and B which are marked with a ✎, making notes in your logbook. Use your *A1 Fourier Series & Transforms* lecture notes and look at the relevant information in HLT pp 5-6 and10-14.

A. Preparatory work

A1. Fourier Series

A pulse function, such as may be measured in a digital circuit, with periodicity T_o is defined by:

$$f(t)=\begin{cases} 1 \quad \text{for } 0 \le t \le \dfrac{T_o}{12} \\ \\ 0 \quad \text{for } \dfrac{T_o}{12} \le t \le \dfrac{T_o}{4} \end{cases} \qquad \begin{cases} f(-t)=f(t) \quad \text{half-wave even symmetry} \\ f\left(\dfrac{T_o}{4}+t\right)=-f\left(\dfrac{T_o}{4}-t\right) \text{ quarter-wave odd symmetry} \end{cases} \tag{1}$$

✎ **Sketch** $f(t)$ over one period. **Show** that the Fourier Series for $f(t)$ is as given below. What is θ_1?

$$f(t)=\sum_{i=1}^{\infty} a_{2i-1}\cos\left[(2i-1)\frac{2\pi}{T_0}t\right] \quad \text{where} \quad a_{2i-1}=\frac{4}{(2i-1)\pi}\sin(2i-1)\theta_1 \tag{2}$$

A2. Convolution

A linear time-invariant system has transfer function and impulse response:

$$G_2(s)=\frac{A}{s+a}; \qquad g_2(t)=Ae^{-at}u(t), \qquad \text{where } u(t)=\text{Heavyside unit step-function} \tag{3}$$

The output response $y(t)$ to an input signal $x(t)$ is given by the convolution integral:

$$y(t)=g(t)*x(t)=\int_0^t g(\lambda)x(t-\lambda)\,\mathrm{d}\lambda$$

✎ Find the responses $y_1(t)$, $y_2(t)$ to the inputs

$$x_1(t)=u(t) \qquad \text{and} \qquad x_2(t)=u(t)e^{-bt} \tag{4}$$

✎ Explain why the integration limits are 0 and t not $-\infty$ to ∞.

A3 Fourier Transform.

The frequency response (i.e. steady state response to sinusoid) of a system is given by the Fourier transform $G(\mathrm{j}\omega)$ of the impulse response $g(t)$.

✎ For the system in Equation (3), obtain $G_2(\mathrm{j}\omega)$ by taking the Fourier Transform of $g_2(t)$.

A4. Sampling and Aliasing

Engineering signals is often processed by digital computers. The signal being processed is not the continuous signal $x(t)$, but a sampled version of the signal $x(nT)$, where T is the sampling interval, and $n = 1, 2, 3, 4, ...$ is the sample number.

The **Nyquist sampling theorem** implies that $x(nT)$ is a true representation of $x(t)$ provided the sampling frequency $f_s = 1/T$ is at least twice the highest frequency in $x(t)$.

Explain with sketches what you would expect to see in a plot of $x(nT)$ for $x(t) = \sin(16\pi t)$, with

(i) $T = 0.01$ sec (ii) $T = 0.05$ sec (iii) $T = 0.1$ sec (iv) $T = 0.2$ sec?

B. The experiment

B1. Fourier Series

Write a MATLAB function coeff(n) to compute the first n non-zero coefficients of the Fourier series for the function $f(t)$ given in Equation 1 in Part A1 of the preparatory work and store the results in a 1×n vector ***a***. In case you have forgotten, here is the structure of the coeff.m file you must create with a text editor All you have to do is replace the ?? signs:

```
function a = coeff(n)
%       file coeff.m            Your Name    5/11/97
%       Returns first n coefficients in the Fourier Series for the
%       function in Ex. 1
theta1=?;
for i=1:n,
        a(i)=4*sin((??)*theta1)/(2*i-1)/pi;
end
plot(a,'+')
```

Use a = coeff(100) to generate the first 100 non-zero coefficients. For the case where the periodicity of $f(t)$ is T_o=2, divide the interval from -T_o/2 to T_o/2 into 2000 equal intervals and form the time column vector t Note the vector t will have 2001 elements.

Write a MATLAB function pltfs(a,n) which plots the sum of the first n terms of the Fourier series of $f(t)$ Again here is some help. You will need the following code in your function: just replace the ?'s:

```
t=[-1:0.001:1]';
f=zeros(2001,1);
for i=1:n,
        f=f+a(i)*cos((?)*t);
end;
plot(t,f);
```

Try different values of n from 1 to 100 Comment on the quality of approximation and verify that at discontinuities the Fourier Series gives the average value of *f*(*t*). What is the effect of increasing the number of terms on the (i) frequency and (ii) amplitude of the Gibbs ripple phenomenon apparent in the plots?

B2. Convolution

In a Matlab m-file `Convolution` Create a sampled time column vector ***t*** running from 0 to 5 seconds, sampled at intervals Δt = 0.01 sec. Also compute column vectors $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ comprising the values of the input functions $x_1(t) = u(t)$ and $x_2(t) = u(t)e^{-bt}$ from equation (4) at these values of *t*, with the constant *b* = 0.5 sec^{-1}.

Also generate the sampled impulse response vector $\boldsymbol{g_2}$ of the system having transfer function and impulse response $G_2(s) = \dfrac{A}{s+a}$; $g_2(t) = Ae^{-at}u(t)$, with the constants *A* = 1 and *a* = 1.0 sec^{-1}:

```
dt=0.01;
t=[0:dt:5]';
x1=ones(length(t),1);
b=0.5;
x2=exp(-b.*t);
a=1;
g2=exp(-??);
```

If *x*(*t*) is the input to a system with transfer function *G*(*s*), the output *y*(*t*), given by the convolution integral can be approximated to

$$y(t) = \int_{-\infty}^{\infty} g(\lambda)x(t-\lambda)\mathrm{d}\lambda \approx \sum_{k=-\infty}^{\infty} g(k\Delta t)x(t-k\Delta t)\Delta t = \Delta t\, conv(g[k],x[k])$$

where "*conv*" denotes discrete time convolution (i.e. convolution of the elements of the vectors ***g*** and ***x*** you have already generated). It is available as the MATLAB function `conv` .

Use this to compute vectors $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$ of the output values of the system $\boldsymbol{g_2}$ for inputs $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ respectively:

```
y1=dt*conv(g2,??);
y2=??;
```

By plotting against *t*, compare these responses with the theoretical responses you derived in ***A2***. What happens if Δt = 1.0 sec?

B3. Fourier transforms

In the numerical domain, with sampled signals, an approximation to the Fourier transform is approximated to by the Discrete Fourier Transform (DFT), for frequencies sufficiently below the Nyquist limit $f = f_s/2 = 1/2T$, *where T* is the sampling interval. The MATLAB form is

$$X(f)=\int_{-\infty}^{\infty} x(t)e^{-\mathrm{j}2\pi ft}\,\mathrm{d}t \approx X\left(\frac{(k+1)}{NT}\right)=\sum_{n=0}^{N-1} x(n+1)e^{-\mathrm{j}\frac{2\pi nk}{N}}$$

The k+1 and n+1 arise because MATLAB vectors of length N run from 1 to N, not 0 to N-1.

If the time input vector $x(n)$ consists of N samples taken at a sampling period of T seconds, the frequency vector $X(k)$ gives the N DFT values spaced at $1/NT$ Hz, with $X(1)$ corresponding to 0 Hz, or DC. The Nyquist frequency limit is at $1/2T$ Hz, and so there is no point in plotting frequencies beyond $X(N/2)$!

Matlab uses a very fast and efficient method of calculating the DFT, called the Fast Fourier Transform. (FFT), using the function `fft`. This works best when N is a power of 2, such as 1024.

Create a time impulse response vector, 1024 points long, by sampling the impulse response $g_2(t)$ in Equation 4 for A = 1 and a = 1:

$\left\{ g_2(t)=e^{-t}u(t)\ ; \quad G_2(\mathrm{j}\omega)=\dfrac{1}{1+\mathrm{j}\omega} \right\}$ at a sampling interval of T = 0.05 sec, remembering that $x(1) = g_2(0)$.

```
dt=0.05;
N=1024;
t = [0:dt:dt*(N-1)]';
x = exp(-t);
```

Create a frequency column vector $f(k)$, 512 points long, starting at 0 Hz, and spaced at $1/NT$ Hz. Use the FFT routine to generate a frequency response and plot the magnitude:

```
f = [0:1/N/dt:511/N/dt]';
xf = fft(x);
xf = abs(xf(1:512));
plot(f,xf)
```

Obtain the Bode magnitude plot

```
loglog(f(2:512),xf(2:512))
```

By plotting (try logarithmic scales `loglog`), see how your results agree with what you would expect from $G_2(\mathrm{j}\omega)=\dfrac{1}{1+\mathrm{j}\omega}$?

Find and plot on linear and logarithmic scales the FFT of the square pulse function

```
x=[ones(1,1/dt),zeros(1,N-1/dt)]';
```

Is it as you would expect?

B4. Sampling and Aliasing

Enter the function alias(T) below and use it to examine the effect of aliasing on the sampled 8 Hz sine wave in section ***A4***. As well as using the values
T = 0.01, 0.05, 0.1 and 0.2 sec suggested, try other values to determine which give good representations of the sine wave. T = 0.125 sec is interesting!

```
function alias(T)
%	alias.m			MLGO	14/11/97
%	plots aliased sine wave, frequency 8 Hertz for 3 sec.
%	sampled every T sec.
NT = 3;
t = [0:T:NT]';
x = sin(16*pi*t);
plot(t,x)
```

What do you consider to be the longest usable sampling frequency 1/T which can be used before aliasing becomes an apparent problem? (An engineering judgement, not a mathematical formula, please!).