Lecture 17 Review

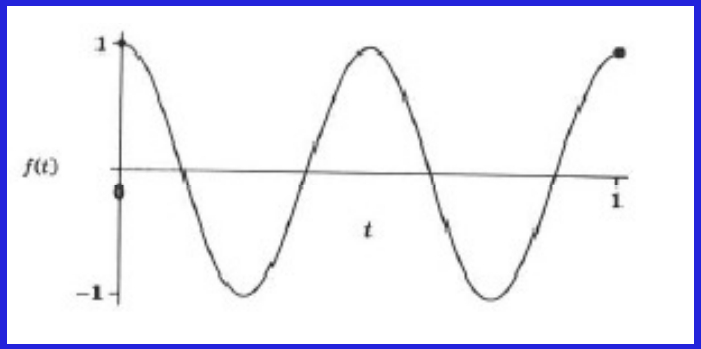
DFT (theory and lab example).

Discrete Fourier Transform (Reprise)

Issue: Often discretely sample a waveform and need to know:

its shape and/or its frequency characteristics.

Sample (i.e., measure w/ an instrument) every Δ seconds.
($1/\Delta$ is the "sampling rate")



cos(2πf t) f = 2
sampled every 1 second

"Nyquist critical frequency" $f_c \equiv \frac{1}{2\Delta}$

Discrete Fourier Transform

$$H_n \equiv \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N}$$

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k n/N}$$

`make_fourier_data.cc`

`dft.cc`

`inv_dft.cc`

Q: sin(2πf t): where is peak n in H_n?

Sampling Theorem and Aliasing

"Nyquist critical frequency" $f_c \equiv \frac{1}{2\Delta}$

Highest frequency
you can accurately reproduce

"Good news:"

- Continuous h(t) samples at intervals Δ
- "bandwidth" limited to frequencies < f_c , i.e., H(f) = 0 $\forall$ |f| > f_c

h(t) completely determined by h_n .

$$h(t) = \Delta \sum_{n=-\infty}^{n=+\infty} h_n \frac{\sin[2\pi f_c(t-n\Delta)]}{\pi(t-n\Delta)}$$

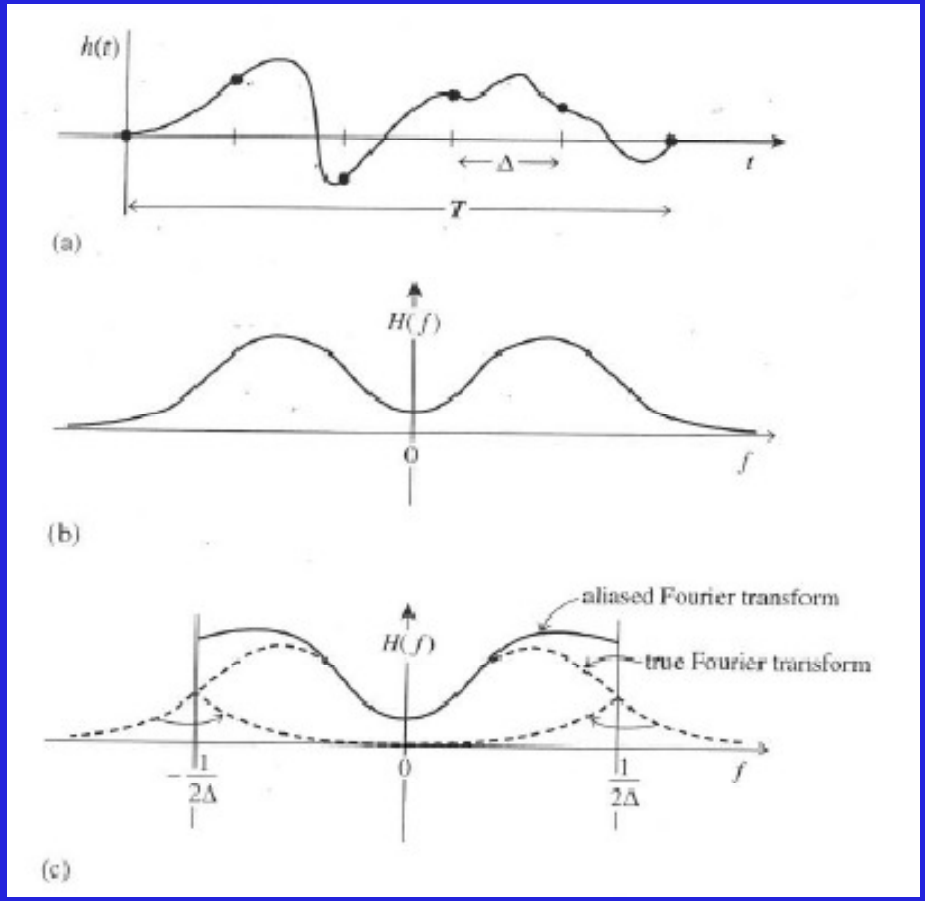
"Bad news:"

If h(t) not bandwidth limited to f < f_c

All of PSD outside $-f_c < f < f_c$

folded into $-f_c < f < f_c$ ("aliasing")

Sampling Theorem and Aliasing (2)



Limit sampled frequencies to < f_c

Fast Fourier Transform

DFT is slooow. Execution time ∝ N*N.

Compare N = 1000 to N = 5000 (yes, try it now…)

Fast Fourier Transform (FFT) to the rescue.

Not a new type of transform, but a new way to calculate DFT.

Uses a "divide and conquer" strategy.

$W \equiv e^{2\pi i/N}$ "twiddle factor" (Nth root of 1)

$H_n \equiv \sum_{k=0}^{N-1} W^{nk} h_k$ index n: 0 → N -1

$F_k = \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j$ index k: 0 → N -1

$$= \sum_{j=0}^{N/2-1} e^{2\pi i(2j)k/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi i(2j+1)k/N} f_{2j+1}$$

$$= \sum_{j=0}^{N/2-1} e^{2\pi ijk/(N/2)} f_{2j} + W^k \sum_{j=0}^{N/2-1} e^{2\pi ijk/(N/2)} f_{2j+1}$$

repeat… $= F_k^e + W^k F_k^o$ $0 \le k \le N - 1$

Execution time ∝ N $\log_2$ N w/ N = power of 2

Fast Fourier Transform (2)

```
make_fourier_data.cc
gsl_fft.cc
gsl_inv_fft.cc
```

→ For optimum efficiency, make N = power of 2
e.g., N = 1024, N = 2048, …, 1 048 576 , …

GSL FFT routines store Imag(H_n) differently from DFT
H(k) = H(N-k)* (Not all 2N H(k) numbers independent.)

- Compare DFT output w/ FFT output.
- Compare execution times: DFT v. FFT w/ N = 4096

FFT (3)

square wave data

dft.dat

0	33	0
1	-27.3487	1.72063
2	13.846	-1.74915
3	-0.32786	0.0625427
4	-6.52502	1.67534
5	5.41695	-1.76007
6	-0.311563	0.123357

inefficient storage

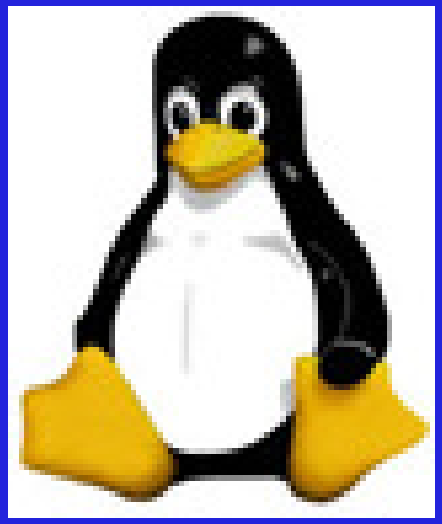
gsl_fft.dat

0	33
1	-27.3487
2	-1.72063
3	13.846
4	1.74915
5	-0.32786
6	-0.0625427
7	-6.52502
8	-1.67534
9	5.41695
10	1.76007
11	-0.311563
12	-0.123357

Why the relative minus (-) sign for Imag H(n) ?

Summary

Sampling theory and aliasing.

FFT (theory and exercises).

Don’t suffer in silence. Scream for help!!!