Lecture 15

Fourier transforms

useful for signal analysis +
for solving differential equations

Fourier series -

a periodic function $f(x)$ defined on a finite interval $0 \leq x < L$ can be written as a Fourier series

If $f(x)$ is symmetric about midpoint @ $x = \frac{1}{2}L$, then we can write

$$f(x) = \sum_{k=0}^{\infty} \alpha_k \cos\left(\frac{2\pi k x}{L}\right)$$

where $\{\alpha_k\}$ is some set of coefficients

If $f(x)$ is antisymmetric about midpoint, then we have

$$f(x) = \sum_{k=1}^{\infty} \beta_k \sin\left(\frac{2\pi k x}{L}\right)$$

So we can write a function w/ no symmetry as

$$f(x) = \sum_{k=0}^{\infty} \alpha_k \cos\left(\frac{2\pi k x}{L}\right) + \sum_{k=1}^{\infty} \beta_k \sin\left(\frac{2\pi k x}{L}\right)$$

can make use of

$$\cos\theta = \frac{1}{2}\left(e^{i\theta} + e^{-i\theta}\right) \quad \& \quad \sin\theta = \frac{i}{2}\left(e^{-i\theta} - e^{i\theta}\right)$$

to write

$$f(x) = \frac{1}{2}\sum_{k=0}^{\infty} \alpha_k \left[\exp\left(i\frac{2\pi k x}{L}\right) + \exp\left(-i\frac{2\pi k x}{L}\right)\right]$$

$$+ \frac{i}{2}\sum_{k=1}^{\infty} \beta_k \left[\exp\left(-i\frac{2\pi k x}{L}\right) - \exp\left(i\frac{2\pi k x}{L}\right)\right]$$

Then we can collect & write as

$$f(x) = \sum_{k=-\infty}^{\infty} \gamma_k \exp\left(i\frac{2\pi k x}{L}\right)$$

where

$$\gamma_k = \begin{cases} \frac{1}{2}\left(\alpha_{-k} + i\beta_{-k}\right) & : k < 0 \\ \alpha_0 & : k = 0 \\ \frac{1}{2}\left(\alpha_k - i\beta_k\right) & : k > 0 \end{cases}$$

Fourier series can only be used for periodic functions. To extend to nonperiodic ones, just pick out an interval of a function + repeat it infinitely so that it becomes periodic.

How to calculate the coefficients γ_k?
just use the fact that $\left\{\exp\left(-i\frac{2\pi kx}{L}\right)\right\}_k$ constitutes an orthonormal basis for the space $[0,L]$. That is, consider that

$$\int_0^L f(x)\exp\left(-i\frac{2\pi kx}{L}\right)dx$$

$$= \sum_{k'=-\infty}^{\infty}\gamma_{k'}\int_0^L \exp\left(i\,2\pi\frac{(k'-k)x}{L}\right)dx$$

if $k' \neq k$ then

$$\int_0^L \exp\left(i\frac{2\pi(k'-k)}{L}x\right) = \frac{L}{i2\pi(k'-k)}\left[\exp\left(i2\pi\frac{(k'-k)x}{L}\right)\right]_0^L$$

$$= \frac{L}{i2\pi(k'-k)}\left[\exp\left(i2\pi(k'-k)\right) - 1\right]$$

$= 0$ b/c $e^{i2\pi n} = 1 \ \forall$ integers n

If $k' = k$ then the integral is equal to L.

$$\Rightarrow \int_0^L f(x) \exp\left(-i \frac{2\pi k x}{L}\right) dx = L \gamma_k$$

or $$\gamma_k = \frac{1}{L} \int_0^L f(x) \exp\left(-i \frac{2\pi k x}{L}\right) dx$$

Discrete Fourier transform

Might not be possible to calculate coefficients γ_k analytically. So we can use numerical methods. Turns out that approximation w/ the trapezoid rule is equivalent to the discrete Fourier transform.

Consider N slices of width $h = L/N$

Applying the trap. rule gives

5

$$\gamma_k = \frac{1}{L}\left(\frac{L}{N}\right)\left[\frac{1}{2}f(0) + \frac{1}{2}f(L) + \sum_{n=1}^{N-1} f(x_n)\exp\left(-i\,\frac{2\pi k\,x_n}{L}\right)\right]$$

where the sample point positions are

$$x_n = \frac{n}{N}\cdot L$$

Since $f(x)$ is periodic, we have $f(0) = f(L)$ & the above simplifies to

$$\gamma_k = \frac{1}{N}\sum_{n=0}^{N-1} f(x_n)\exp\left(-i\,\frac{2\pi k\,x_n}{L}\right)$$

$c_k = N\gamma_k$

can use this to evaluate coefficients.

useful for when sampled data is evenly spaced, as is ~~the~~ often the case when sampling.

these results were derived using trapezoid rule which is generally an approximation but there is a sense in which it is exact.

Recall that $\displaystyle\sum_{k=0}^{N-1} a^k = \frac{1-a^N}{1-a}$ when $a \neq 1$

then $$\sum_{k=0}^{N-1} \left(e^{i\frac{2\pi m}{N}}\right)^k = \frac{1 - e^{i2\pi m}}{1-e^{i\frac{2\pi m}{N}}}$$

But m integer $\Rightarrow$ $e^{i2\pi m} = 1$,

so the above is equal to zero.

if $m=0$ or is a multiple of N then the sum is N.

So $$\sum_{k=0}^{N-1} \exp\left(i\,2\pi \frac{km}{N}\right) = \begin{cases} N & \text{if } m = 0, N, 2N, \ldots \\ 0 & \text{else} \end{cases}$$

Then consider the sum

$$\sum_{k=0}^{N-1} c_k \exp\left(i\frac{2\pi kn}{N}\right)$$

$$= \sum_{k=0}^{N-1}\left[\sum_{n'=0}^{N-1} y_{n'} \exp\left(-i\,\frac{2\pi k n'}{N}\right)\right] \exp\left(i\frac{2\pi kn}{N}\right)$$

$$= \sum_{n'=0}^{N-1} y_{n'} \sum_{k=0}^{N-1} \exp\left(i\,2\pi k\left(\frac{n-n'}{N}\right)\right)$$

$$= \sum_{n'=0}^{N-1} y_{n'} \, \delta_{n,n'} \, N$$

$$= N y_n$$

$$\Rightarrow \quad y_n = \frac{1}{N} \sum_{k=0}^{N-1} c_k \exp\left(i \frac{2\pi k n}{N}\right)$$

this is the inverse discrete Fourier transform (inv. DFT)

this proves that the matrix w/ entries $U_{kn} = \frac{1}{\sqrt{N}} \exp\left(-i \frac{2\pi k n}{N}\right)$ is a unitary matrix

So we can recover the original values <u>exactly</u> by performing the inverse DFT.

can move freely back & forth between original values & Fourier coefficients

- we can compute this on a computer b/c the sum is finite.

- This discrete formula only gives sample values $y_n = f(x_n)$, so if function is oscillating rapidly between samples, the DFT won't capture this, so DFT just gives some idea of function.

If the function is real, then we can use this symmetry to simplify further.

Suppose all y_n are real & consider c_k for $\frac{N}{2} < k \le N-1$, so

$$k = N - r \quad \text{for} \quad 1 \le r < \tfrac{1}{2}N$$

then $$c_{N-r} = \sum_{n=0}^{N-1} y_n \exp\left(-i \frac{2\pi (N-r) n}{N}\right)$$

$$= \sum_{n=0}^{N-1} y_n \exp(-i 2\pi n) \exp\left(i \frac{2\pi r n}{N}\right)$$

$$= \sum_{n=0}^{N-1} y_n \exp\left(i \frac{2\pi r n}{N}\right) = c_r^*$$

$\Rightarrow c_{N-1} = c_1^*, \quad c_{N-2} = c_2^*, \text{ etc.}$

So when calculating the DFT of a real function, we only have to calculate c_k for $0 \le k \le \frac{1}{2}N$.

However if the y_n are complex, then we need to calculate all N F. coefficients.

Bring up dft.py

using exp from cmath package.

not the quickest way to calculate DFT. can instead do FFT.

If we shift the positions of the sample points, then not much changes.

Suppose that instead of taking samples @ $x_n = \frac{n}{N} L$, we take them @ $x_n' = x_n + \Delta$. Then

$$c_k = \sum_{n=0}^{N-1} f(x_n + \Delta) \exp\left(-i \frac{2\pi k (x_n + \Delta)}{L}\right)$$

$$= \exp\left(-i \frac{2\pi k \Delta}{L}\right) \sum_{n=0}^{N-1} f(x_n') \exp\left(-i \frac{2\pi k x_n}{L}\right)$$

$$= \exp\left(-i \frac{2\pi k \Delta}{L}\right) \sum_{n=0}^{N-1} y_n' \exp\left(-i \frac{2\pi k n}{N}\right)$$

where $y_n' = f(x_n')$ are the new samples

can absorb the phase factors into the coefficients as $c_k' = \exp\left(i \frac{2\pi k \Delta}{L}\right) c_k$

so that

$$c_k' = \sum_{n=0}^{N-1} y_n' \exp\left(-i \frac{2\pi k n}{L}\right) \quad \text{\& so}$$

DFT is essentially independent of where samples are taken.

can distinguish between a Type-I DFT where we divide interval $[0,L]$ into N slices & take samples @ endpoints & a Type II where we take samples @ midpoints of slices.

2D – Fourier transform

useful for processing images, for example & astronomy

suppose we have an $M \times N$ grid of samples y_{mn}. 1st do an FT of the rows:

$$c'_{ml} = \sum_{n=0}^{N-1} y_{mn} \exp\left(-i\frac{2\pi l n}{N}\right)$$

& now FT the m variable

$$c_{kl} = \sum_{m=0}^{N-1} c'_{ml} \exp\left(-i\frac{2\pi k m}{M}\right)$$

can combine as

$$c_{kl} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} y_{mn} \exp\left(-i2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)\right)$$

What is the FT doing?

breaking down a signal into its frequency components, like a signal analyzer.

Bring up dft.py

1st spike is the frequency of the main wave & the others are harmonics

---

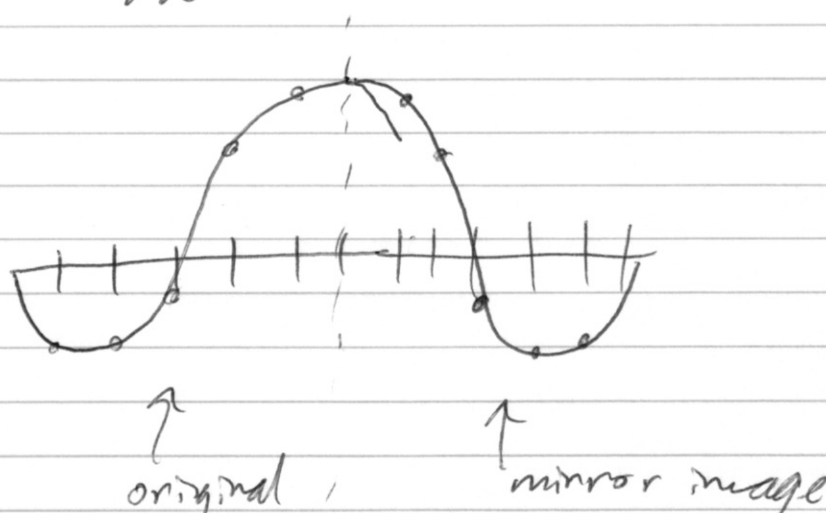
Discrete cosine transform

Recall that if a function is symmetric about $x = \frac{1}{2}L$ (midpoint) then we can write

$$f(x) = \sum_{k=0}^{\infty} \alpha_k \cos\left(\frac{2\pi k x}{L}\right)$$

so we cannot do this for all functions.

However, if we'd like to do so, we can by simply sampling a function over an interval & then add to it its mirror image, i.e.,



So we make the function symmetric & when the samples are, we have

$y_0 = y_N$, $y_1 = y_{N-1}$, $y_2 = y_{N-2}$, etc...

we then get for the DFT:

$$c_k = \sum_{n=0}^{N-1} y_n \exp\left(-i\frac{2\pi kn}{N}\right)$$

$$= \sum_{n=0}^{\frac{1}{2}N} y_n \exp\left(-i\frac{2\pi kn}{N}\right) + \sum_{n=\frac{1}{2}N+1}^{N-1} y_n \exp\left(-i\frac{2\pi kn}{N}\right)$$

$$= \sum_{n=0}^{\frac{1}{2}N} y_n \exp\left(-i\frac{2\pi k n}{N}\right) + \sum_{n=\frac{1}{2}N+1}^{N} y_{N-n} \exp\left(i\,2\pi k \left(\frac{N-n}{N}\right)\right)$$

using that
$\exp(i 2\pi k) = 1$

Make a change of variables

$N-n \to n$ to get

$$c_k = \sum_{n=0}^{\frac{1}{2}N} y_n \exp\left(-i\frac{2\pi k n}{N}\right) + \sum_{n=1}^{\frac{1}{2}N-1} y_n \exp\left(i\,\frac{2\pi k n}{N}\right)$$

$$= y_0 + y_{N/2} \cos\left(\frac{2\pi k (N/2)}{N}\right) + 2\sum_{n=1}^{\frac{1}{2}N-1} y_n \cos\left(\frac{2\pi k n}{N}\right)$$

usually the discrete cosine transform is applied to real values,

$\Rightarrow$ c_k coefficients are real.

In this case, we have that

$c_{N-r} = c_r^* = c_r$ + the inverse transform is

$$y_n = \frac{1}{N}\sum_{k=0}^{N-1} c_k \exp\left(i\frac{2\pi kn}{N}\right)$$

$$= \frac{1}{N}\left[\sum_{k=0}^{\frac{1}{2}N} c_k \exp\left(i\frac{2\pi kn}{N}\right) + \sum_{k=\frac{1}{2}N+1}^{N-1} c_k \exp\left(i\frac{2\pi kn}{N}\right)\right]$$

$$= \frac{1}{N}\left[\sum_{k=0}^{\frac{1}{2}N} c_k \exp\left(i\frac{2\pi kn}{N}\right) + \sum_{k=\frac{1}{2}N+1}^{N-1} c_{N-k} \exp\left(-i\frac{2\pi (N-k)n}{N}\right)\right]$$

$$= \frac{1}{N}\left[\quad '' \quad + \sum_{k=1}^{\frac{1}{2}N-1} c_k \exp\left(-i\frac{2\pi kn}{N}\right)\right]$$

$$= \frac{1}{N}\left\{c_0 + c_{N/2}\cos\left(\frac{2\pi n (N/2)}{N}\right) + 2\sum_{k=1}^{\frac{1}{2}N-1} c_k \cos\left(\frac{2\pi kn}{N}\right)\right]$$

inverse discrete cosine transform

So much symmetry that the DCT is the same as its inverse!

If we take the samples at midpoints, then can show that coefficients are

$$a_k = 2 \sum_{n=0}^{\frac{1}{2}N-1} y_n \cos\left(\frac{2\pi k(n+\frac{1}{2})}{N}\right)$$

& inverse transform is

$$y_n = \frac{1}{N}\left[a_0 + 2\sum_{k=1}^{\frac{1}{2}N-1} a_k \cos\left(\frac{2\pi k(n+\frac{1}{2})}{N}\right)\right]$$

Nice feature of DCT is that it does not assume that function is periodic. Neither does DFT, but it does force the 1st & last values to be the same, which can create a large discontinuity. The DCT does not do this.

can also calculate discrete sine transform but often not used because it forces endpoints to be zero.