Fourier Transform

Prof. George Wolberg
Dept. of Computer Science
City College of New York

10-30
2:24 rec start

Objectives

- This lecture reviews Fourier transforms and processing in the frequency domain.
  - Definitions
  - Fourier series
  - Fourier transform
  - Fourier analysis and synthesis
  - Discrete Fourier transform (DFT)
  - Fast Fourier transform (FFT)

Background (1)

- Fourier proved that any periodic function can be expressed as the sum of sinusoids of different frequencies, each multiplied by a different coefficient. → *Fourier series*
- Even aperiodic functions (whose area under the curve is finite) can be expressed as the integral of sinusoids multiplied by a weighting function. → *Fourier transform*
- In a great leap of imagination, Fourier outlined these results in a memoir in 1807 and published them in *La Theorie Analitique de la Chaleur* (The Analytic theory of Heat) in 1822. The book was translated into English in 1878.

Background (2)

- The Fourier transform is more useful than the Fourier series in most practical problems since it handles signals of finite duration.
- The Fourier transform takes us between the spatial and frequency domains.
- It permits for a dual representation of a signal that is amenable for filtering and analysis.
- Revolutionized the field of signal processing.

Example

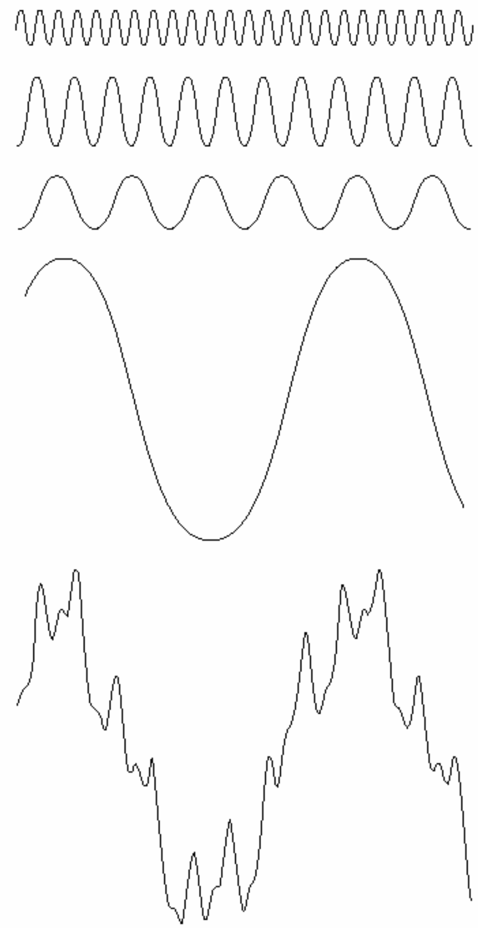
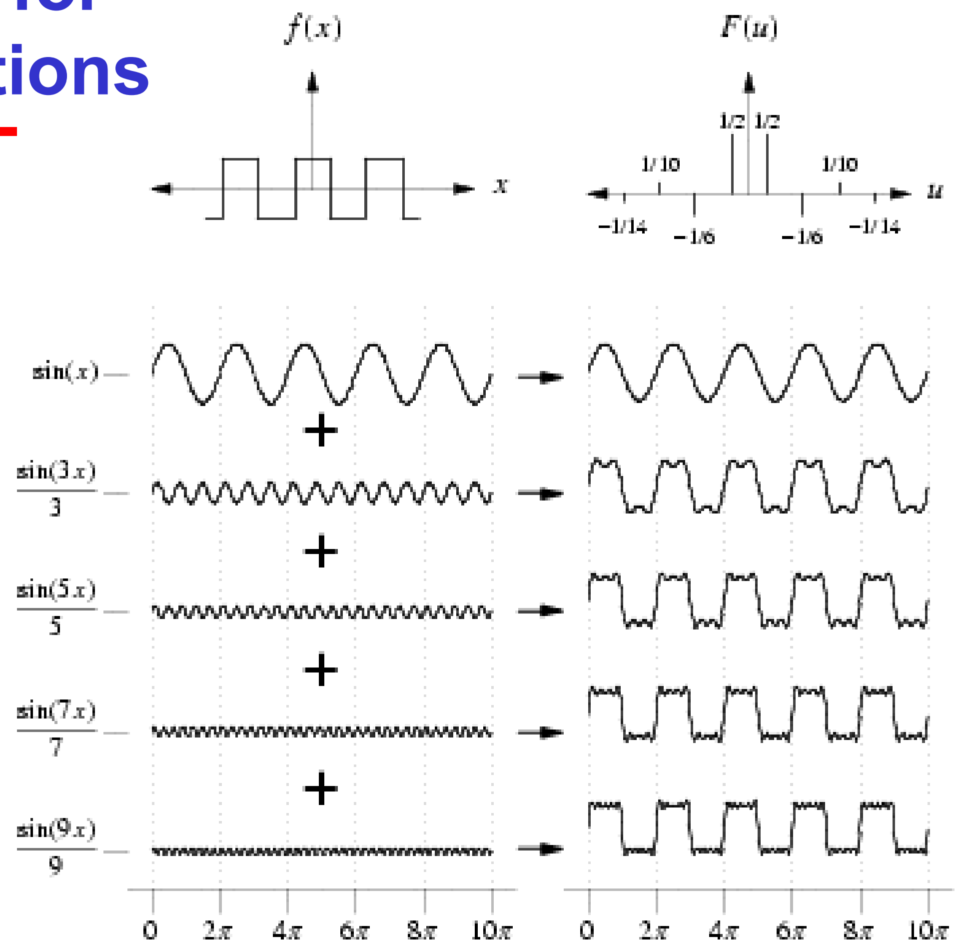


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

Useful Analogy

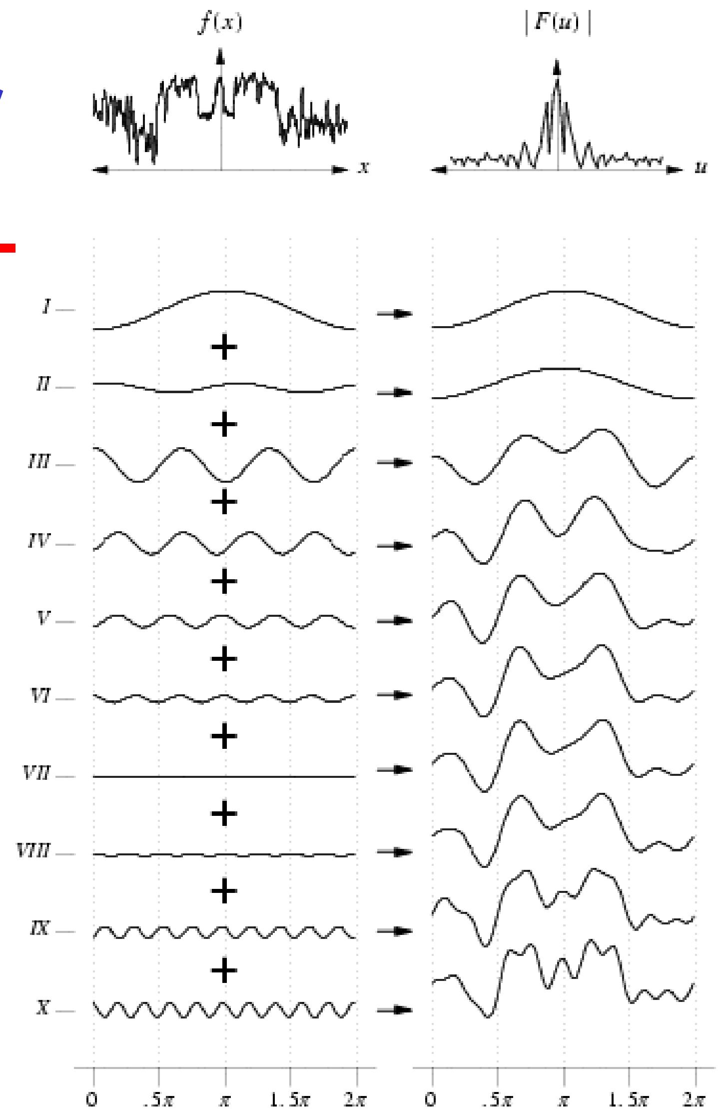
- A glass prism is a physical device that separates light into various color components, each depending on its wavelength (or frequency) content.
- The Fourier transform is a mathematical prism that separates a function into its frequency components.

Fourier Series for Periodic Functions



10-30 start (late by 25 min)

Fourier Transform for Aperiodic Functions



Fourier Analysis and Synthesis

- Fourier analysis: determine amplitude & phase shifts
- Fourier synthesis: add scaled and shifted sinusoids together
- Fourier transform pair:

has a mistake?

$$\text{Forward F.T.} \quad \underbrace{F(u)}_{\text{Forier domain}} = \int_{-\infty}^{\infty} \overbrace{f(x)\underbrace{e^{-i2\pi ux}}_{\text{some set of weights}}}^{\text{kinda like convolutio}} dx \quad \leftarrow \text{spacial to freq}$$

$$\text{Inverse F.T.} \quad f(x) = \int_{-\infty}^{\infty} F(u) e^{+i2\pi ux} dx \quad \leftarrow \text{freq to spacial}$$

where $i = \sqrt{-1}$, and

$$e^{\pm i2\pi ux} = \cos(2\pi ux) \pm i \sin(2\pi ux) \leftarrow \text{complex exponential at freq. } u$$

Euler's formula

Fourier Coefficients

- Fourier coefficients *F(u)* specify, for each frequency *u*, the amplitude and phase of each complex exponential.
- *F(u)* is the frequency spectrum.
- *f(x)* and *F(u)* are two equivalent representations of the same signal.

From Forier you get the the magnitude spectrum and the phase spectrum

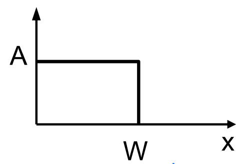
$$F(u) = R(u) + iI(u)$$

$$|F(u)| = \sqrt{R^2(u) + I^2(u)} \leftarrow \text{ magnitude spectrum; aka Fourier spectrum}$$

$$\Phi(u) = \tan^{-1}\frac{I(u)}{R(u)} \leftarrow \text{phase spectrum}$$

$$\begin{aligned} P(u) &= |F(u)|^2 \\ &= R^2(u) + I^2(u) \leftarrow \text{ spectral density} \end{aligned}$$

1D Example



$$f(x)=\begin{cases} A & 0\le x\le W \\ 0 & x>W \end{cases}$$

pixel val

$$F(u)=\int_{-\infty}^{\infty} f(x)e^{-i2\pi ux}dx$$

$$\boxed{\text{note}: \int e^{ax}dx=\frac{1}{a}e^{ax}}$$

e is eulers formula

I know this

$e^{i\theta} = \cos\theta + i\sin\theta$

$e^{-i\theta} = \cos\theta - i\sin\theta$

so: $e^{i\theta} - e^{-i\theta} = 2i\sin\theta$

So $\boxed{\frac{e^{i\theta}-e^{-i\theta}}{2i} = \sin\theta}$

$$F(u)=\int_{0}^{W} Ae^{-i2\pi ux}dx$$

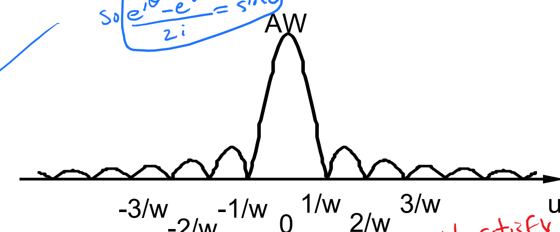
$$F(u)=\frac{-A}{i2\pi u}\left[e^{-i2\pi ux}\right]_0^W=\frac{-A}{i2\pi u}\left[e^{-i2\pi uw}-1\right]=\frac{A}{i2\pi u}\left[1-e^{-i2\pi uw}\right]$$

$$F(u)=\frac{A}{i2\pi u}\left[e^{i\pi uw}-e^{-i\pi uw}\right]e^{-i\pi uw}$$

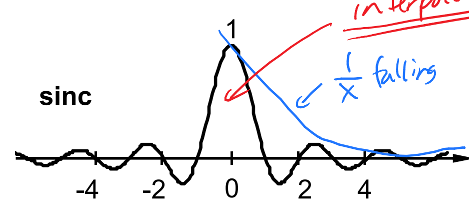
$$\boxed{\text{note}: \sin x=\frac{e^{ix}-e^{-ix}}{2i}}$$

$$F(u)=\frac{A}{\pi u}\sin(\pi uW)e^{-i\pi uW} \leftarrow \textit{complex function}$$

$$|F(u)|=\left|\frac{A}{\pi u}\right|\left|\sin(\pi uW)\right|\left|e^{-i\pi uW}\right|=AW\left|\frac{\sin(\pi uW)}{\pi uW}\right|=AW\left|\operatorname{sinc}(\pi uW)\right|$$



would satisfy the constraint that its a interpolation kernel



$$\boxed{\text{where } \operatorname{sinc}(x)=\frac{\sin(\pi x)}{\pi x} \quad (\textit{some books have } \operatorname{sinc}(x)=\frac{\sin(x)}{x})}$$

Wolberg: Image Processing Course Notes

a sin wave that is degraded by the x at the bottom

∴ Amplitude of the sine wave is droping by $\frac{1}{x}$

$\sin(x)$:

$\frac{1}{x}$:

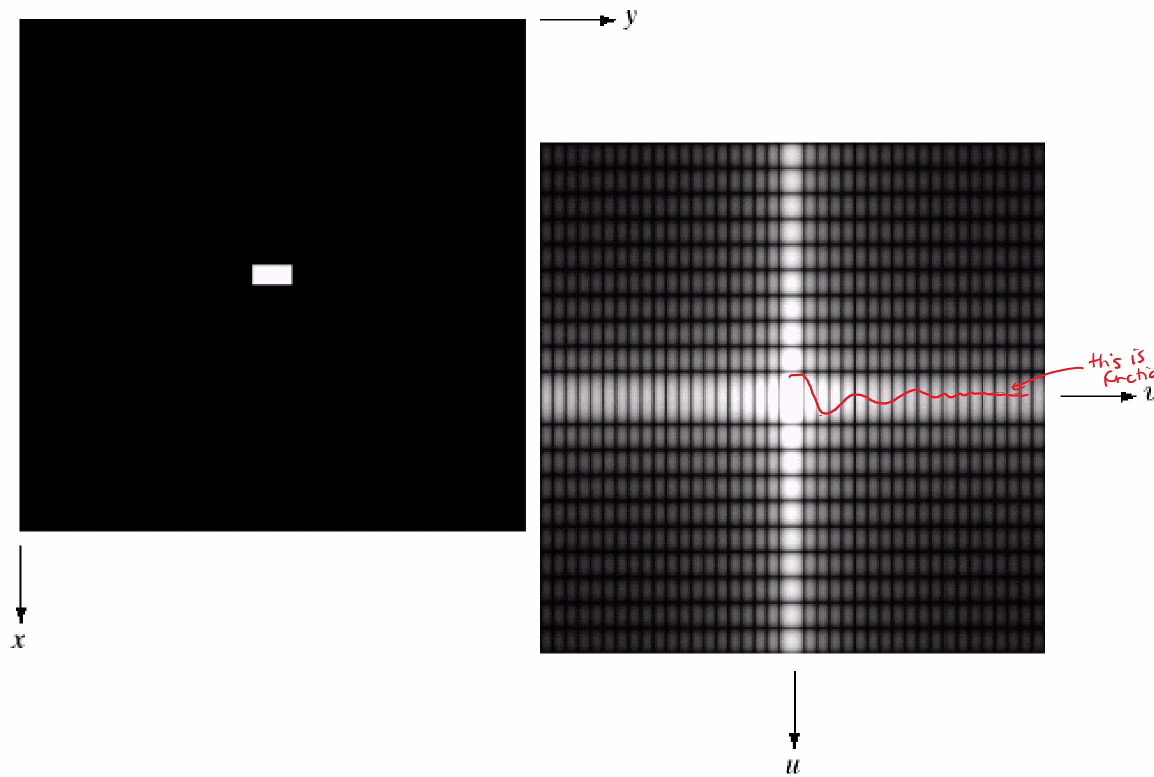
$\frac{\sin(x)}{x}$:

Something like that

2D Example

a b

FIGURE 4.3 (a) Image of a 20×40 white rectangle on a black background of size 512×512 pixels. (b) Centered Fourier spectrum shown after application of the log transformation given in Eq. (3.2-2). Compare with Fig. 4.2.



this is like a sinc function

Fourier Series (1)

For periodic signals, we have the Fourier series:

$$f(x) = \sum_{n=-\infty}^{n=\infty} c(nu_0) e^{i2\pi u_0 x} \text{ where } c(nu_0) \text{ is the } n^{th} \text{ Fourier coefficient}$$

$$c(nu_0) = \frac{1}{x_0} \int_{-x_0/2}^{x_0/2} f(x) e^{-i2\pi u_0 x} dx$$

That is, the periodic signal contains all the frequencies that are harmonics of the fundamental frequency.

Fourier Series (2)

$$c(nu_0) = \frac{1}{x_0}\int_{-x_0/2}^{x_0/2} f(x)e^{-i2\pi nu_0 x}dx = \frac{1}{x_0}\int_{-W/2}^{W/2} Ae^{-i2\pi nu_0 x}dx$$

$$c(nu_0) = \frac{A}{-i2\pi nu_0 x_0}(e^{-i\pi nu_0 W} - e^{+i\pi nu_0 W})$$

$$c(nu_0) = \frac{A}{\pi n}\sin(\pi nu_0 W) \quad \leftarrow \sin x = \frac{e^{ix} - e^{-ix}}{2i}; \; u_0 x_0 = 1$$

$$c(nu_0) = \frac{Au_0 W}{\pi nu_0 W}\sin(\pi nu_0 W) = Au_0 W\mathrm{sinc}(\pi nu_0 W)$$

Note that if $\frac{W}{2} = \frac{x_0}{2}$, then we have a square wave and

$$c(nu_0) = Au_0 x_0 \mathrm{sinc}(\pi nu_0 x_0)$$

$$c(nu_0) = \begin{cases} A\,\mathrm{sinc(n)} & n = \pm 1, \pm 3, \cdots \\ 0 & n = 0, \pm 2, \pm 4, \cdots \end{cases}$$

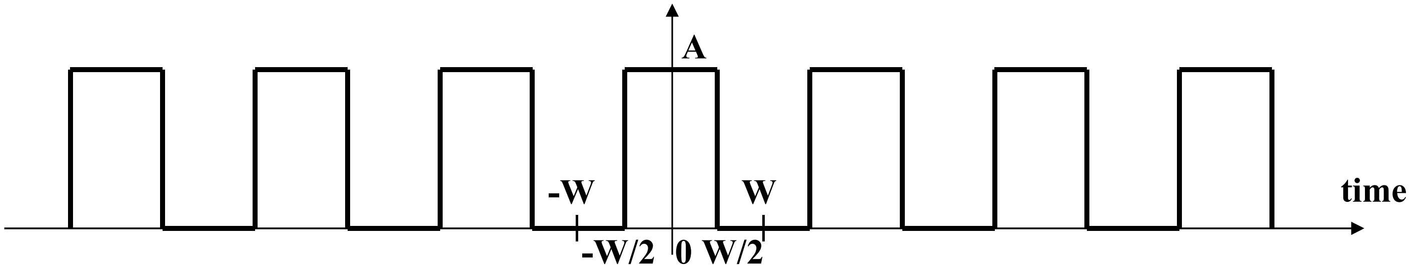
Fourier Series (3)

- The Fourier transform is applied for aperiodic signals.
- It is represented as an integral over a continuum of frequencies.
- The Fourier Series is applied for periodic signals.
- It is represented as a summation of frequency components that are integer multiples of some fundamental frequency.

Example

Ex : Rectangular Pulse Train

$$f(x) = \begin{cases} A & |x| < \frac{W}{2} \\ 0 & |x| > \frac{W}{2} \end{cases} \quad \text{in interval}[-W/2, W/2]$$



Discrete Fourier Transform

pixel val

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-i 2\pi \frac{ux}{N}} \qquad \text{forward DFT}$$

$$f(x) = \sum_{u=0}^{N-1} F(u) e^{+i 2\pi \frac{ux}{N}} \qquad \text{inverse DFT}$$

for $0 \le u \le N-1$ and $0 \le x \le N-1$ where N is the number of equi-spaced input samples.

The 1/N factor can be in front of *f(x)* instead.

$$F(u)=\frac{1}{N}\underbrace{\sum_{x=0}^{N-1} f(x)e^{-i2\pi\frac{ux}{N}}} \qquad \text{forward DFT}$$

$$\Sigma\, f(x)\left[\cos\left(\frac{2\pi ux}{N}\right)+i\sin\left(\frac{2\pi ux}{N}\right)\right]$$

$$\sum_{0}^{N} f(x)\cos\left(\frac{2\pi ux}{N}\right)+f(x)\,i\sin\left(\frac{2\pi ux}{N}\right)$$

for $(x=0;\ x<N;\ x{+}{+})\{$

real $+= f[x]\cdot\cos\left(\frac{2\pi ux}{N}\right)$

img $+= f[x]\cdot\sin\left(\frac{2\pi ux}{N}\right)$

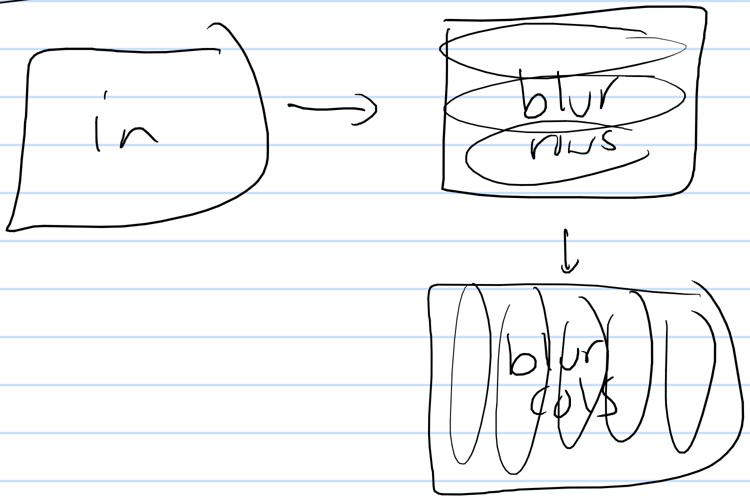
Save the coeff to real and img

freq u has a real and img coeff

$F_{real}[u] = real$

$F_{imaginary}[u] = img$

like blur will do for fourier



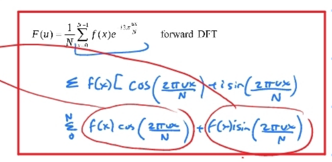
$$f[x]\cdot g[x]$$

$$[F_{real}+i\,f_{img}]\cdot[g_{real}+i\,g_{img}]$$

$$F_{real}\cos + i\,f_{real}\sin$$
$$+\ i\,f_{img}\cos - f_{img}\sin$$

did he make a mistake?

```
for(u=0; u<N; u++) { /*compute spectrum over all freq. u */
 real = imag = 0;    /*reset real, imag component of F(u)*/
 for(x=0; x<N; x++) { /* visit each input pixel */
      real += (f[x]*cos(-2*PI*u*x/N));
      imag += (f[x]*sin(-2*PI*u*x/N));
      /* Note: if f is complex, then
      real += (fr[x]*cos()-fi[x]*sin());
      imag += (fr[x]*sin()+fi[x]*cos());
      because (fr+ifi)(gr+igi)=(frgr-figi)+i(figr+frgi)
      */
 }
 Fr[u] = real / N;
 Fi[u] = imag / N;
}
```



$F(u)=\frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-i2\pi\frac{ux}{N}}$ forward DFT

$\Sigma\, f(x)[\cos(\frac{2\pi ux}{N})+i\sin(\frac{2\pi ux}{N})]$

$\sum_{0}^{N} f(x)\cos(\frac{2\pi ux}{N})+f(x)i\sin(\frac{2\pi ux}{N})$

$$F(u)=\frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-i2\pi\frac{ux}{N}} \qquad \text{forward DFT}$$

$$f(x)=\sum_{u=0}^{N-1} F(u)e^{+i2\pi\frac{ux}{N}} \qquad \text{inverse DFT}$$

forier.cpp

Fourier Analysis Code

signal to signal coeff

- DFT maps N input samples of f into the N frequency terms in F.

```
for(u=0; u<N; u++) { /*compute spectrum over all freq. u */
 real = imag = 0;    /*reset real, imag component of F(u)*/
 for(x=0; x<N; x++) { /* visit each input pixel */
       real += (f[x]*cos(-2*PI*u*x/N));
       imag += (f[x]*sin(-2*PI*u*x/N));
       /* Note: if f is complex, then
       real += (fr[x]*cos()-fi[x]*sin());
       imag += (fr[x]*sin()+fi[x]*cos());
       because (f_r+if_i)(g_r+ig_i)=(f_rg_r-f_ig_i)+i(f_ig_r+f_rg_i)
       */
 }
 Fr[u] = real / N;
 Fi[u] = imag / N;
}
```

$$F(u) = \frac{1}{N}\underbrace{\sum_{x=0}^{N-1} f(x)e^{-i2\pi\frac{ux}{N}}} \qquad \text{forward DFT}$$

$$\sum f(x)\left[\cos\left(\frac{2\pi ux}{N}\right) + i\sin\left(\frac{2\pi ux}{N}\right)\right]$$

$$\sum_0^N f(x)\cos\left(\frac{2\pi ux}{N}\right) + f(x)i\sin\left(\frac{2\pi ux}{N}\right)$$

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-i2\pi\frac{ux}{N}} \qquad \text{forward DFT}$$

$$f(x) = \sum_{u=0}^{N-1} F(u)e^{+i2\pi\frac{ux}{N}} \qquad \text{inverse DFT}$$

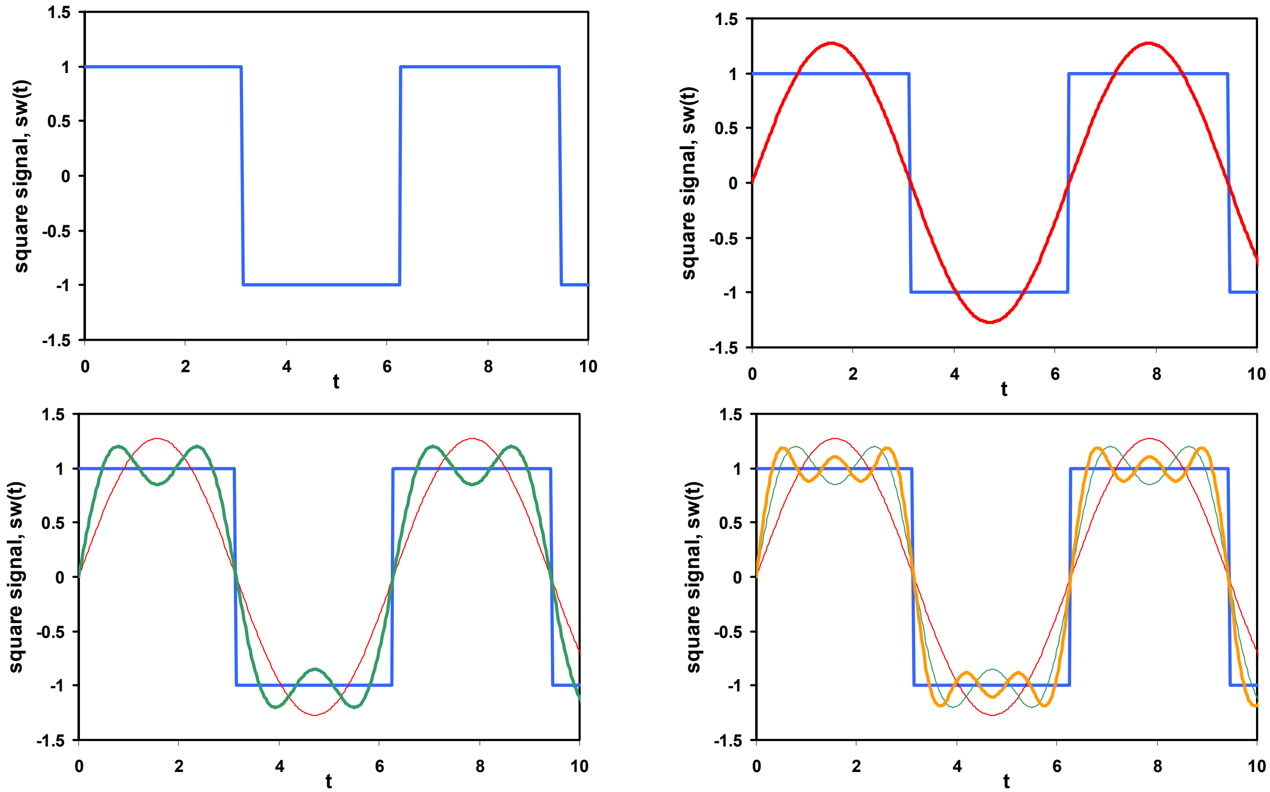
Fourier Synthesis Code

backwards.

```
for(x=0; x<N; x++) {/* compute each output pixel       */
 real = imag = 0;   /* reset real, imaginary component */
 for(u=0; u<N; u++) {
       c = cos(2*PI*u*x/N);
       s = sin(2*PI*u*x/N);
       real += (Fr[u]*c-Fi[u]*s);
       imag += (Fr[u]*s+Fi[u]*c);
 }
 fr[x] = real; /* OR f[x] = sqrt(real*real + imag*imag);
 fi[x] = imag;
}
```

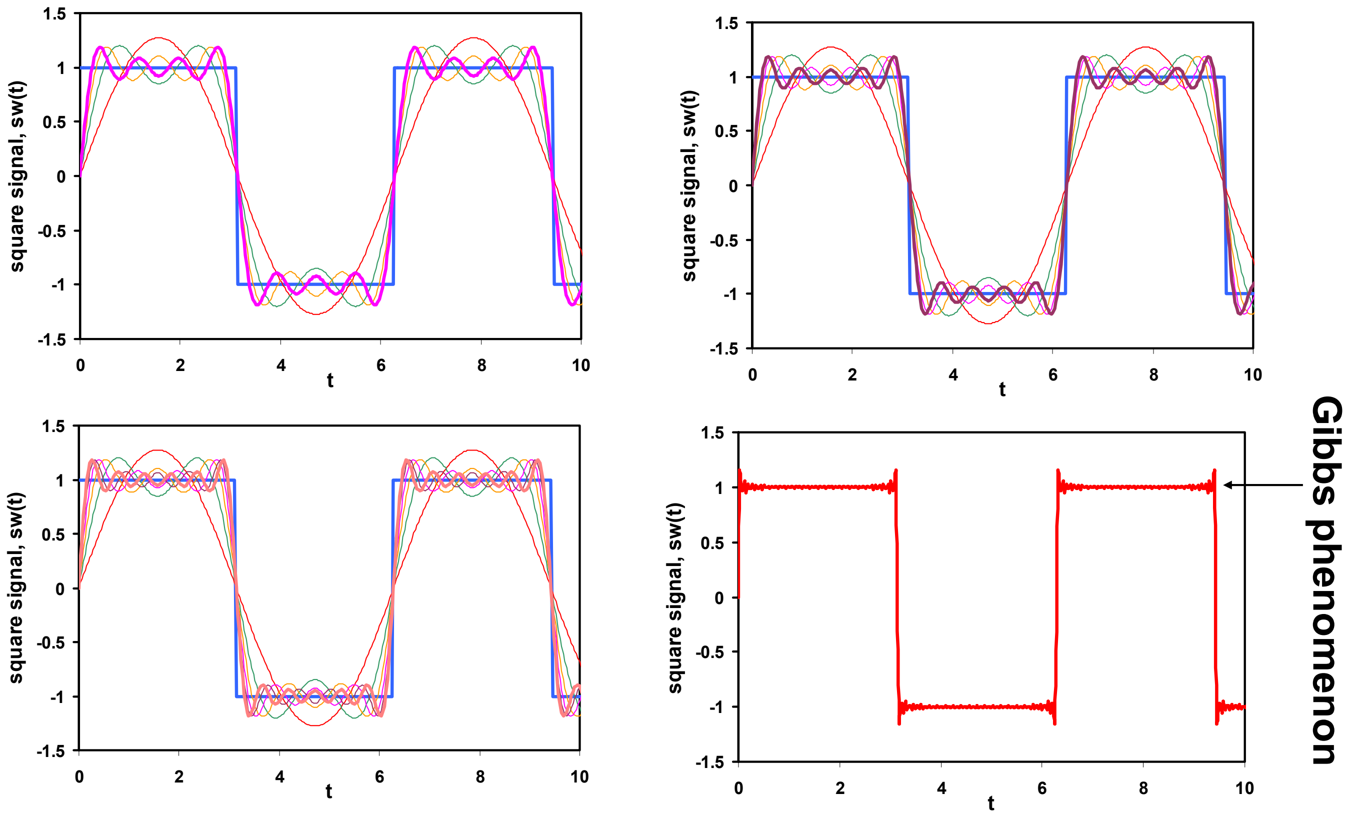
Late

Example: Fourier Analysis (1)



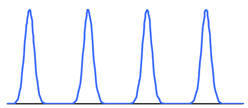
late

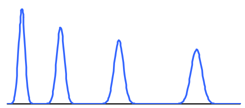
Example: Fourier Analysis (2)



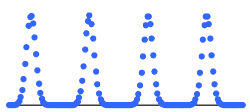
late

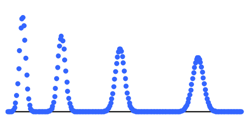
Summary





Continuous	Periodic *(period T)*	**FS**	Discrete	$c_k = \frac{1}{T}\cdot\int_0^T s(t)\cdot e^{-ik\omega t}\,dt$
	Aperiodic	**FT**	Continuous	$S(f) = \int_{-\infty}^{+\infty} s(t)\cdot e^{-i2\pi f t}\,dt$





Discrete	Periodic *(period T)*	**DFS**	Discrete	$\tilde{c}_k = \frac{1}{N}\sum_{n=0}^{N-1} s[n]\cdot e^{-i\frac{2\pi k n}{N}}$
	Aperiodic	**DTFT**	Continuous	$S(f) = \sum_{n=-\infty}^{+\infty} s[n]\cdot e^{-i2\pi f n}$
		DFT	Discrete	$\tilde{c}_k = \frac{1}{N}\sum_{n=0}^{N-1} s[n]\cdot e^{-i\frac{2\pi k n}{N}}$

Note: $i=\sqrt{-1}$, $\omega = 2\pi/T$, $s[n]=s(t_n)$, N = ***# of samples***

2:18
10 m

(4+C

2D Fourier Transform

= done in 2 passes
Just like bluring in HW2

Continuous:

$$F\{f(x,y)\} = F(u,v) = \iint f(x,y)e^{-i2\pi(ux+vy)}dx = \iint f(x,y)e^{-i2\pi ux}e^{-i2\pi vy}dx$$

$$F^{-1}\{F(u,v)\} = f(x,y) = \iint F(u,v)e^{+i2\pi(ux+vy)}dx = \iint F(u,v)e^{+i2\pi ux}e^{+i2\pi vy}dx$$

Separable: F(u, v) = F(u)F(v)
Discrete:

What I had by is the input to the 2nd pass

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1} f(x,y)e^{-i2\pi(\frac{ux}{N}+\frac{vy}{M})}dx = \frac{1}{M}\sum_{y=0}^{M-1}\left[\frac{1}{N}\sum_{x=0}^{N-1} f(x,y)e^{-i2\pi(\frac{ux}{N})}\right]e^{-i2\pi(\frac{vy}{M})}$$

$$f(x,y) = \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} F(u,v)e^{+i2\pi(\frac{ux}{N}+\frac{vy}{M})}dx = \sum_{y=0}^{M-1}\left[\sum_{x=0}^{N-1} F(u,v)e^{+i2\pi(\frac{ux}{N})}\right]e^{+i2\pi(\frac{vy}{M})}$$

7:21

Separable Implementation

$$F(u,v) = \frac{1}{N}\sum_{y=0}^{N-1} e^{-j2\pi vy/N} \boxed{\frac{1}{M}\sum_{x=0}^{M-1} f(x,y)e^{-j2\pi ux/M}}$$

transform each row

$$= \frac{1}{N}\sum_{y=0}^{N-1} F(u,y)e^{-j2\pi vy/N}$$

transform each column of intermediate result

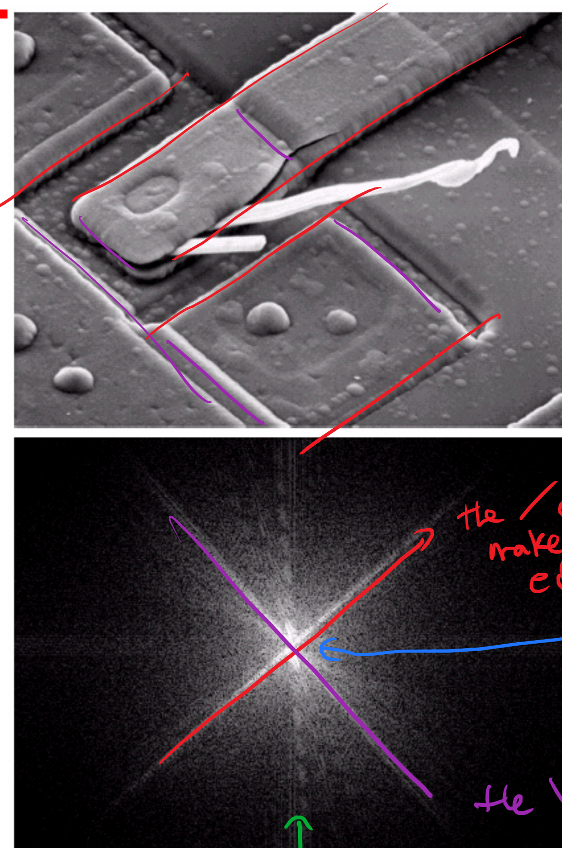
where $F(u,y) = \frac{1}{M}\sum_{x=0}^{M-1} f(x,y)e^{-j2\pi ux/M}$

The 2D Fourier transform is computed in two passes:

1) Compute the transform along each row independently.
2) Compute the transform along each column of this intermediate result.

Properties

- Edge orientations in image appear in spectrum, rotated by 90°.
- 3 orientations are prominent: 45°, -45°, and nearly horizontal long white element.



a
b

FIGURE 4.4 (a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

the / edges make this vertical edge

0 freq @ center

always ⊥

the \ edges make this

there are some horizontal — lines

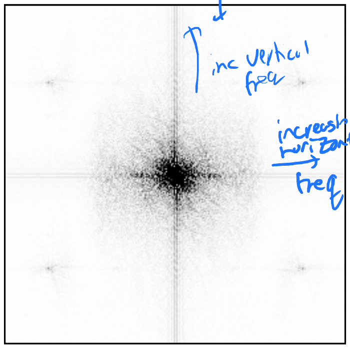
2:25

$\sqrt{\text{real}^2 + \text{imaginary}^2}$?

Magnitude and Phase Spectrum

each dot has a diff freq
0 @ center

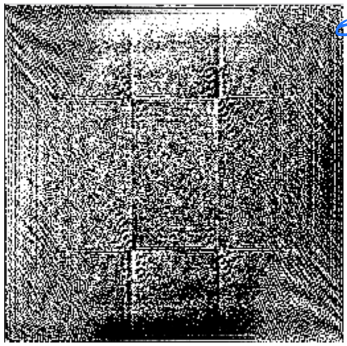
Mad.bw



inc vertical freq

increasing horizontal freq

Magnitude

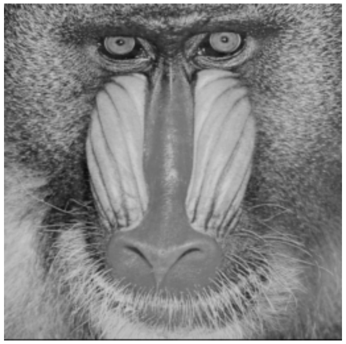


Phase

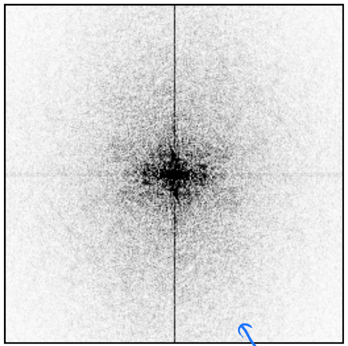
looks like garbage when we display phase (so not shown in textbks)

but its more imp bcc it has off → image wars

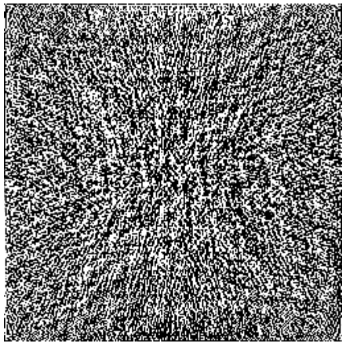
2D-Fourier transforms example



Mandrill.bw



Magnitude



Phase

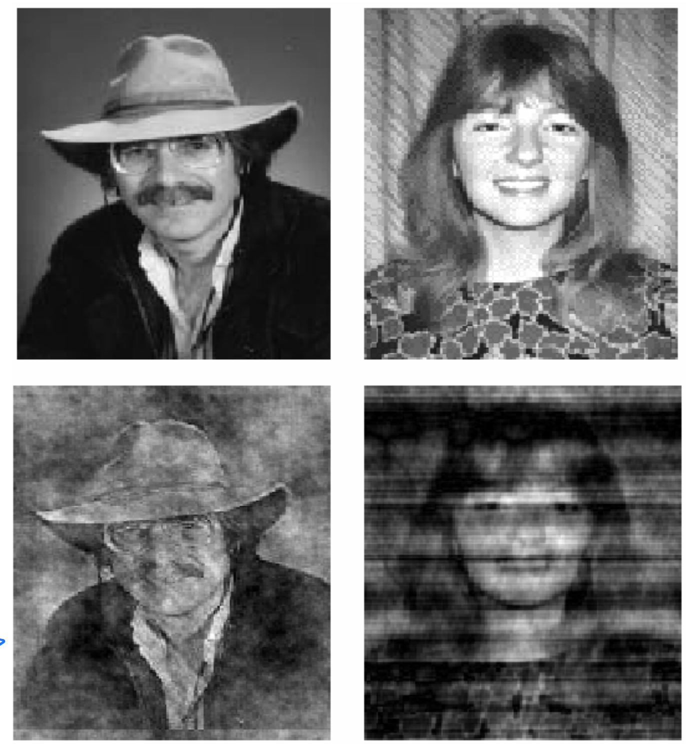
fur is high freq (on off on off ... fast)
so high freq in this

HW3: Inverse Foreir

Role of Magnitude vs Phase (1)

Rick *Linda*

Pictures reconstructed using the Fourier phase of another picture



Mag{Linda}
Phase{Rick}

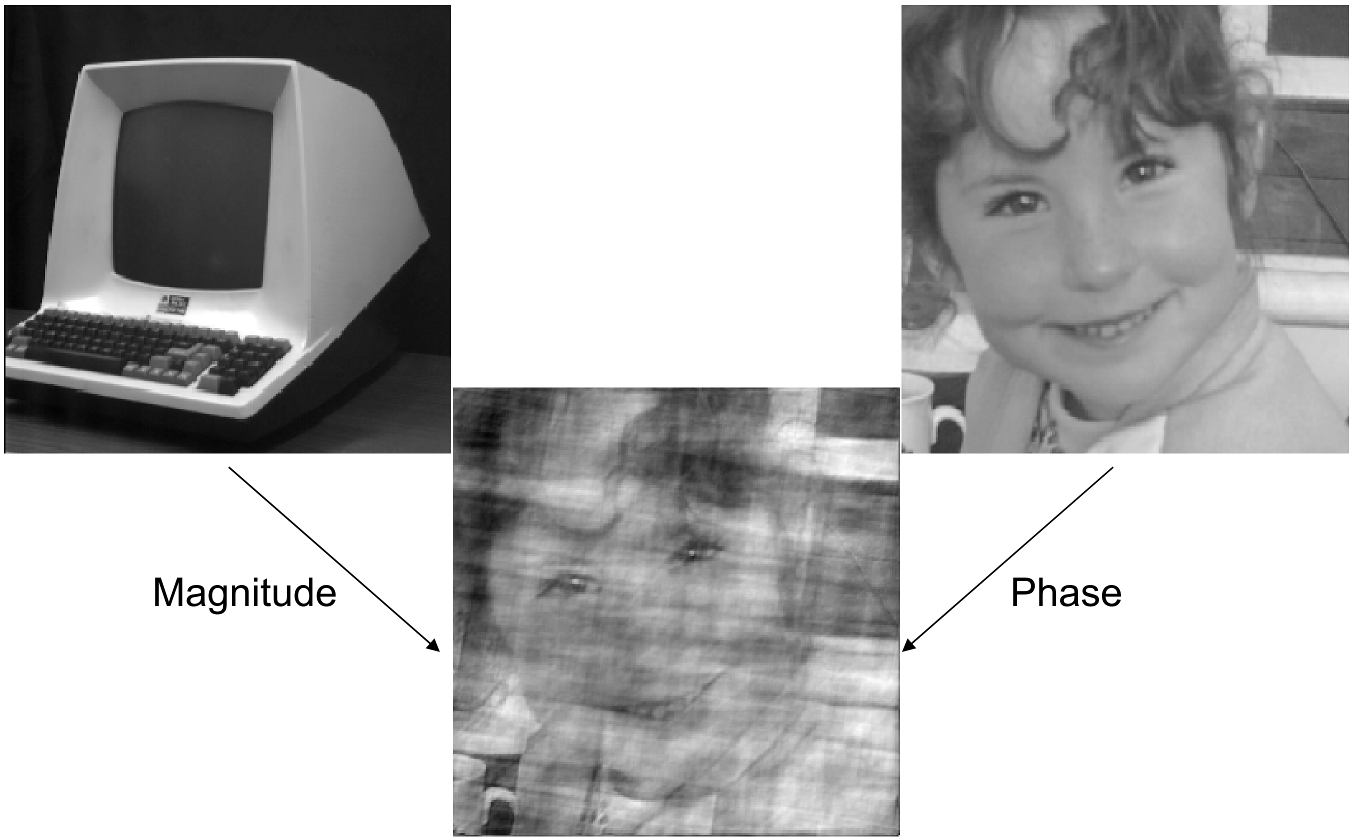
magnitude of linda
plase of Rick

Mag{Rick}
Phase{Linda}

Phase more imortant than magnitude!

why: phase keeps edges intact

Role of Magnitude vs Phase (2)

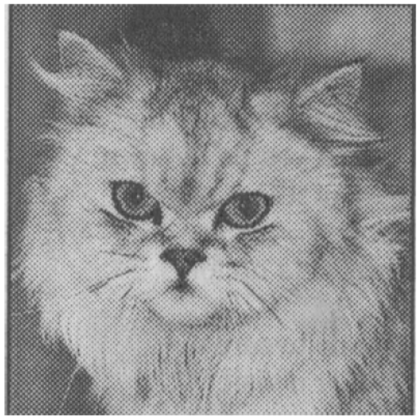


Fourier Transform can show noises that has a certain pattern
→ shows the freq

Noise Removal

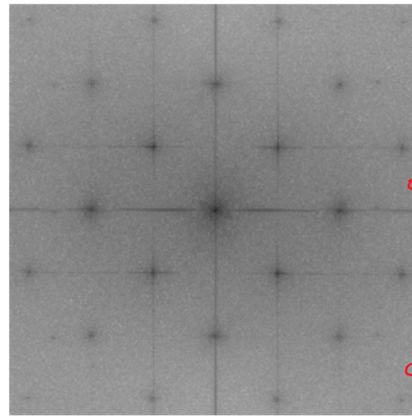
going from spacial to freq domain

1



Original with noise patterns

The noise is a pattern



Power spectrum showing noise spikes

← darker the val here the brighter the pixel

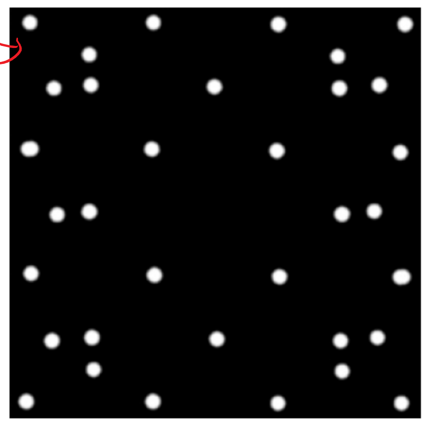
← higher freq → less dim spots

(periodic screening)

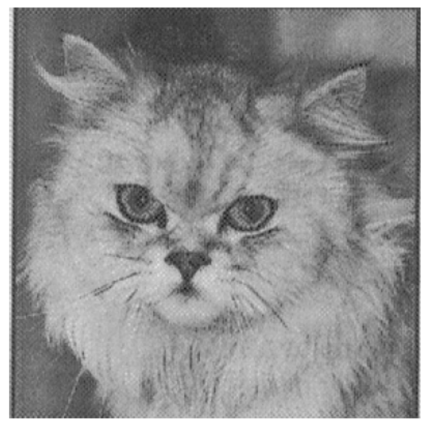
want to remove it!

2

making a mask to remove the periodic noise



Mask to remove periodic noise



Inverse FT with periodic noise removed

3

← edit the noise out in freq dom by bring it back to spacial dom

Fast Fourier Transform (1)

- The DFT was defined as:

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x) e^{-i2\pi \frac{ux}{N}} \qquad 0 \le x \le N-1$$

Convolution: sum of prods: pixels · weight

Rewrite:

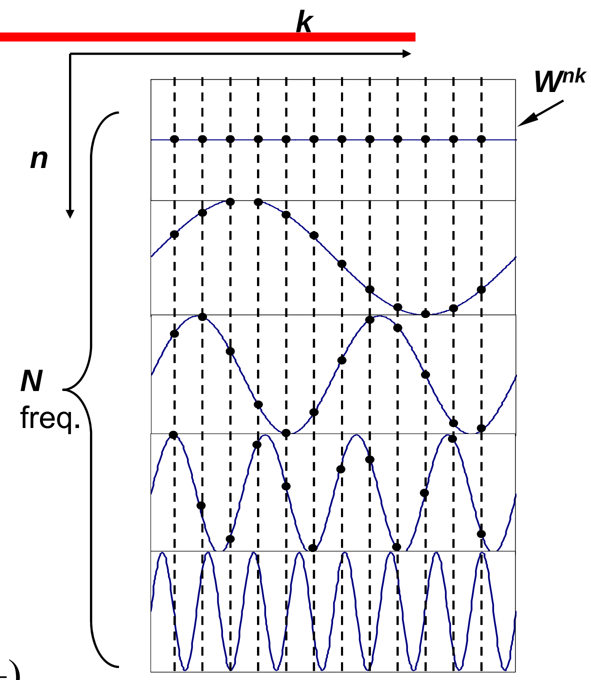
$$F_n = \frac{1}{N}\sum_{k=0}^{N-1} f_k e^{-i2\pi \frac{nk}{N}} \qquad 0 \le n \le N-1$$

$$\text{Let } F_n = \sum_{k=0}^{N-1} f_k W^{nk}$$

$$\text{where } W = e^{\frac{-i2\pi}{N}} = \cos\left(\frac{-2\pi}{N}\right) + i\sin\left(\frac{-2\pi}{N}\right)$$

Also, Let $N = 2^r$ (N is a power of 2)

Let N be a power of 2 will see why later

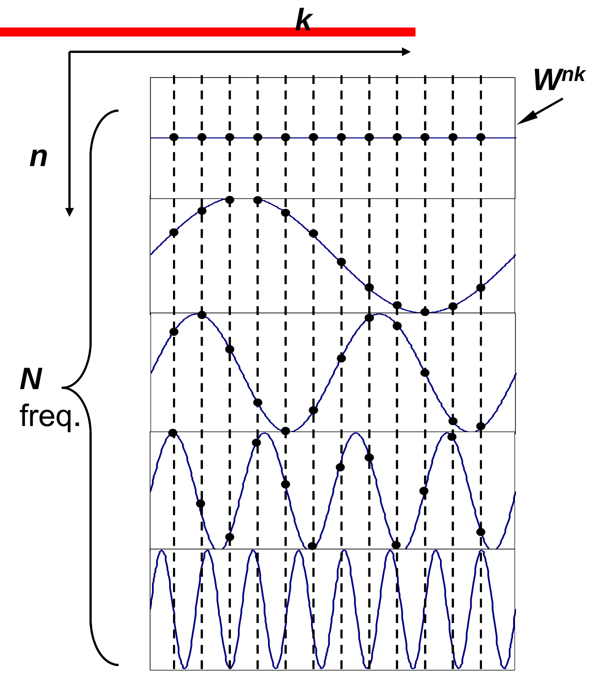


2:44

Fast Fourier Transform (2)

- W^{nk} can be thought of as a 2D array, indexed by n and k.
- It represents N equispaced values along a sinusoid at each of N frequencies.
- For each frequency n, there are N multiplications (N samples in sine wave of freq. n). Since there are N frequencies, DFT: $O(N^2)$
- With the FFT, we will derive an $O(N \log N)$ process.

faster version



1024·1024 vs 1024·10 ??

he was saying
something like th.

2:46

Computational Advantage

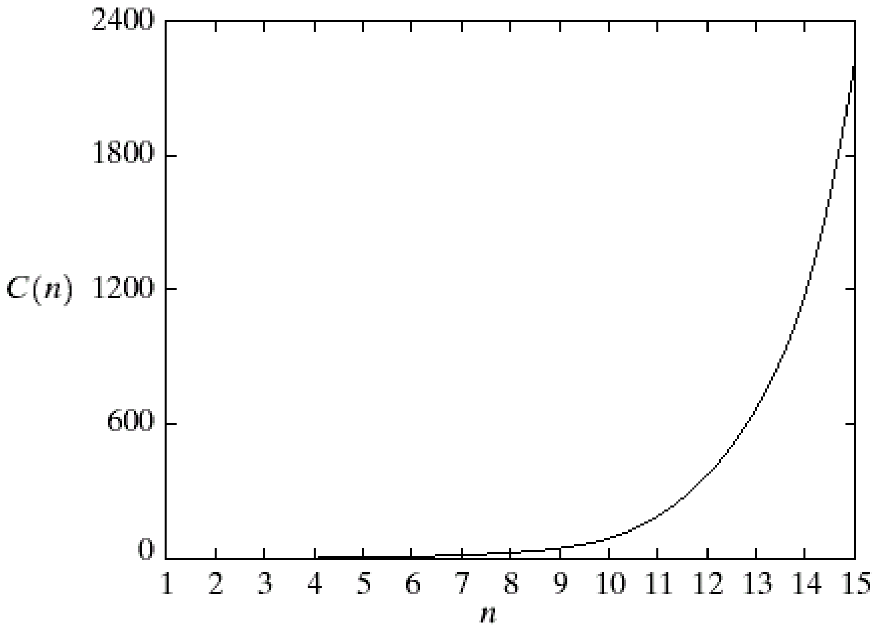


FIGURE 4.42 Computational advantage of the FFT over a direct implementation of the 1-D DFT. Note that the advantage increases rapidly as a function of n.

2:46

Danielson–Lanczos Lemma (1)

2 groups

1942:

$$F_n = \sum_{k=0}^{N-1} f_k e^{-i2\pi\frac{nk}{N}}$$

→ collecting odd vals

$$F_n = \sum_{k=0}^{\frac{N}{2}-1} f_{2k} e^{-i2\pi\frac{n(2k)}{N}} + \sum_{k=0}^{\frac{N}{2}-1} f_{2k+1} e^{-i2\pi\frac{n(2k+1)}{N}}$$

→ collecting every other value (even #s)

k=0 to $\frac{N}{2}$

Even Numbered Terms — Odd Numbered Terms

$f_0, f_2, f_4, \cdots$ — $f_1, f_3, f_5, \cdots$

$e^{-i2\pi\frac{n2k}{N}}$ $e^{-i2\pi\frac{n}{N}}$

Why do this?

$$F_n = \sum_{k=0}^{\frac{N}{2}-1} f_{2k} e^{\frac{-i2\pi nk}{N/2}} + W^n \sum_{k=0}^{\frac{N}{2}-1} f_{2k+1} e^{\frac{-i2\pi nk}{N/2}}$$

$$W = e^{\frac{-i2\pi}{N}}$$

$$F_n = F_n^e + W^n F_n^o$$

Divide+Conquer prob

Danielson–Lanczos Lemma (2)

$$F_n = \sum_{k=0}^{\frac{N}{2}-1} f_{2k} e^{\frac{-i2\pi nk}{N/2}} + W^n \sum_{k=0}^{\frac{N}{2}-1} f_{2k+1} e^{\frac{-i2\pi nk}{N/2}}$$

$$W = e^{\frac{-i2\pi}{N}}$$

need to multiply odd # by the W

even #

$$F_n = F_n^e + W^n F_n^o$$

odd # terms

n^th component of F.T. of length *N/2* formed from the **even** components of *f*

n^th component of F.T. of length *N/2* formed from the **odd** components of *f*

Divide-and-Conquer solution: Solving a problem (F_n) is reduced to 2 smaller ones.

Potential Problem: n in F_n^e and F_n^o is still made to vary from 0 to N-1. Since each sub-problem is no smaller than original, it appears wasteful.

Solution: Exploit symmetries to reduce computational complexity.

N = length of list
n = 0 to N

Danielson–Lanczos Lemma (3)

Given : a DFT of length N, $F_{n+N} = F_n$

$e^{-i2\pi k} = \cos(2\pi k) - i\sin(2\pi k)$
$= 1$

$$\text{Proof}: F_{n+N} = \sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi(n+N)k}{N}} \Rightarrow \sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi nk}{N}} e^{\frac{-i2\pi \cancel{N}k}{\cancel{N}}} =$$

$$= \sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi nk}{N}} (\cos 2\pi k \cancel{- i\sin 2\pi k}) = \boxed{\sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi nk}{N}} = F_n}$$

1

$$W^{n+\frac{N}{2}} = \cos\left(\frac{-2\pi}{N}\left(n+\frac{N}{2}\right)\right) + i\sin\left(\frac{-2\pi}{N}\left(n+\frac{N}{2}\right)\right)$$

$$= \cos\left(\frac{-2\pi n}{N} - \pi\right) + i\sin\left(\frac{-2\pi n}{N} - \pi\right)$$

$$= -\cos\left(\frac{-2\pi n}{N}\right) - i\sin\left(\frac{-2\pi n}{N}\right)$$

$$\boxed{W^{n+\frac{N}{2}} = -W^n}$$

Why are they useful?
→ butterfly →

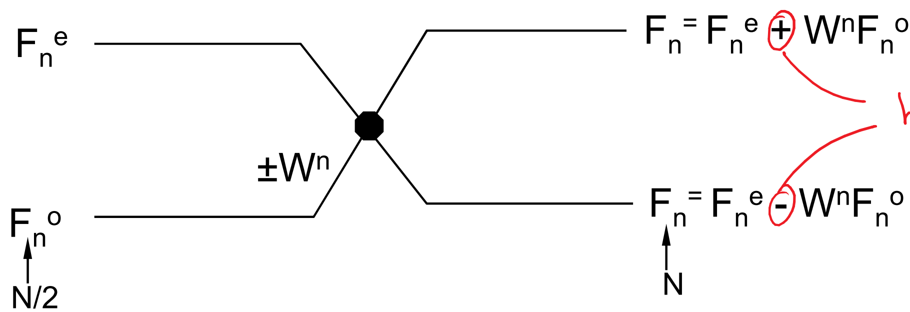
Main Points of FFT

$$F_n = \sum_{k=0}^{N-1} f_k e^{-i2\pi \frac{nk}{N}}$$

$$\begin{array}{ccccc} F_n & = & F_n^e & + & W^n F_n^o \\ \text{length } N & & \text{length } \tfrac{N}{2} & & \text{length } \tfrac{N}{2} \end{array}$$

but $0 \le n < N$

Butterfly
float path?



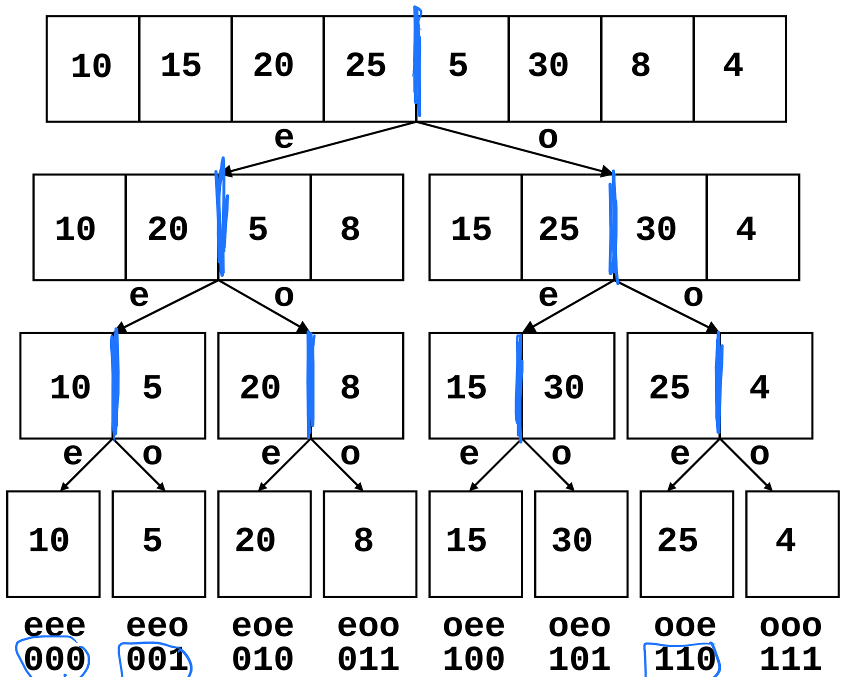
$W^{n+\frac{N}{2}} = -W^n$

Oh exam

FFT Example (1)

- Input: 10, 15, 20, 25, 5, 30, 8, 4

didn't I do this in algorithms?

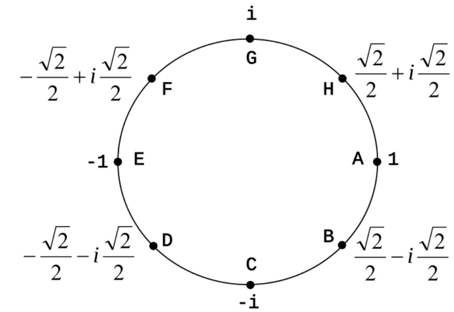


0 100 → 4 → index 4 is 5! 011 → index 3 is 25!

11-11 rec start 2:15 (came late)

why is it fast?

FFT Example (2)

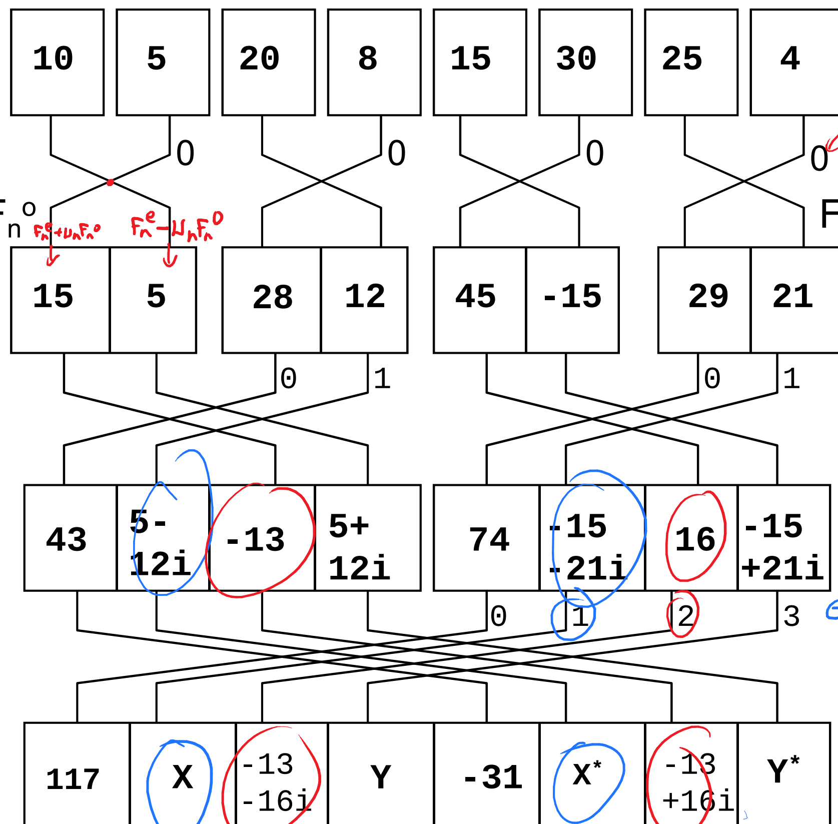


We compute this and that other # once so its trivial

$F_n = F_n^e + W^n F_n^o$

$F_n = F_n^e - W^n F_n^o$

represents the n in W^n



· n^2 to nlogn

· only do # crunching nlogn times instead of n^2 times?

Save result as DFT (DFT → just use eq) pg 405

if N=4 len is 4

$W^1 = e^{-i2\pi/4} = -i$

$\cos(\pi/2) - i\sin(\pi/2) = -i$

X=(5-12i)+(-15-21i)B

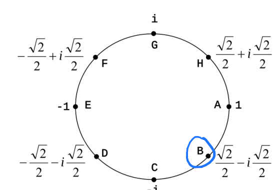
Y=(5+12i)+(-15+21i)D

$W^1 = e^{-i2\pi/8} = B$

$W^2 = e^{-i4\pi/8} = C$

$W^3 = e^{-i6\pi/8} = D$

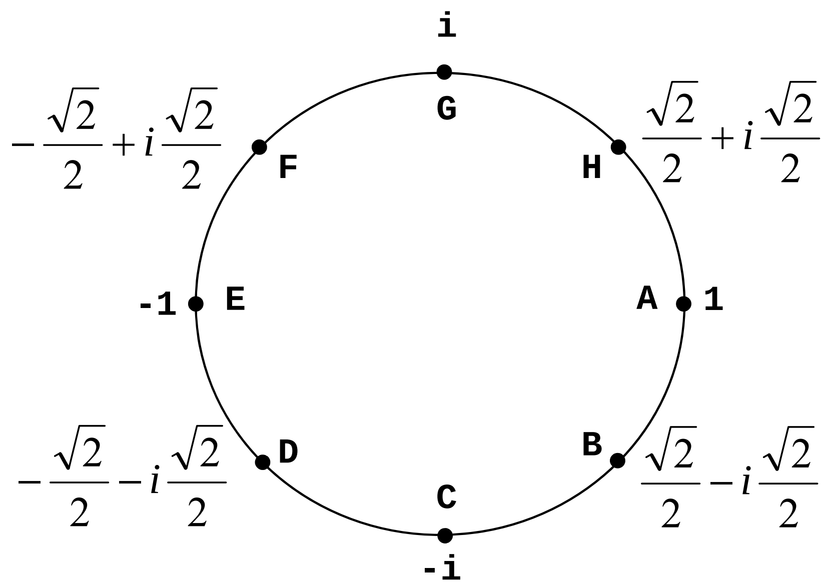
(see next page for weights B, C, D)



flip sign

Weights

- DFT is a convolution with kernel values $e^{-i2\pi ux/N}$
- These values are derived from a unit circle.



Code in new slide + fourier.c in website

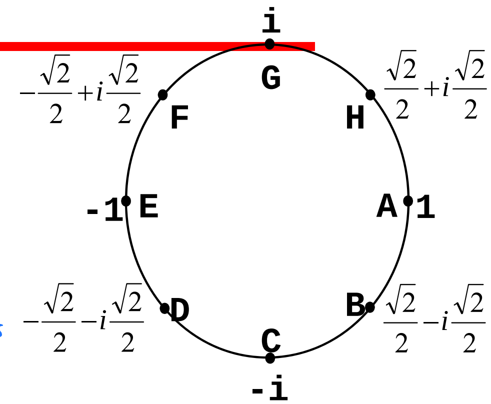
DFT method now

easier but FFT is faster (divide + conquer)

in test DFT vs FFT way do example

DFT Example (1)

- Input: 10, 15, 20, 25, 5, 30, 8, 4



$$F_n = \sum_{k=0}^{N-1} f_k e^{-i2\pi\frac{nk}{N}}$$

n=0 → e^0 → 1 so just sum of pixels

$$F_0 = 10+15+20+25+5+30+8+4 = \boxed{117}$$

$$F_1 = 10e^{-i2\pi(1)(0)/8} + 15e^{-i2\pi(1)(1)/8} + 20e^{-i2\pi(1)(2)/8} + \ldots + 8e^{-i2\pi(1)(6)/8} + 4e^{-i2\pi(1)(7)/8}$$

$$= 10A + 15B + 20C + 25D + 5E + 30F + 8G + 4H$$

$$F_2 = 10e^{-i2\pi(2)(0)/8} + 15e^{-i2\pi(2)(1)/8} + 20e^{-i2\pi(2)(2)/8} + \ldots + 8e^{-i2\pi(2)(6)/8} + 4e^{-i2\pi(2)(7)/8}$$

Weights =2 you skip the next weight

$$= 10\overset{(1)}{A} + 15\overset{(-i)}{C} + 20\overset{(-1)}{E} + 25\overset{(i)}{G} + 5\overset{(1)}{A} + 30\overset{(-i)}{C} + 8\overset{(-1)}{E} + 4\overset{(i)}{G} = \boxed{-13-16i}$$

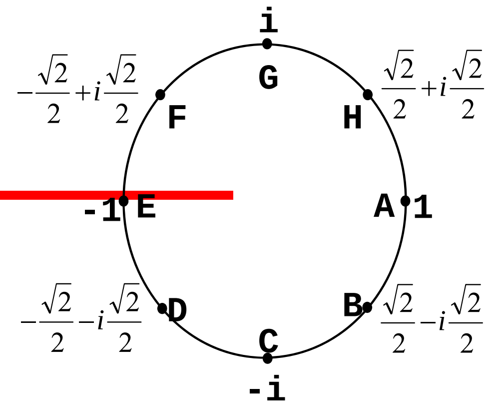
$$F_3 = 10e^{-i2\pi(3)(0)/8} + 15e^{-i2\pi(3)(1)/8} + 20e^{-i2\pi(3)(2)/8} + \ldots + 8e^{-i2\pi(3)(6)/8} + 4e^{-i2\pi(3)(7)/8}$$

Weight=3 you skip 2 steps (2 weights)

$$= 10A + 15D + 20G + 25B + 5E + 30H + 8C + 4F$$

As you can see → get same answer as in FFT!

DFT Example (2)



$$F_4 = 10e^{-i2\pi(4)(0)/8} + 15e^{-i2\pi(4)(1)/8} + 20e^{-i2\pi(4)(2)/8} + \ldots + 8e^{-i2\pi(4)(6)/8} + 4e^{-i2\pi(4)(7)/8}$$

$$= 10A + 15E + 20A + 25E + 5A + 30E + 8A + 4E = \boxed{-31}$$

$$F_5 = 10e^{-i2\pi(5)(0)/8} + 15e^{-i2\pi(5)(1)/8} + 20e^{-i2\pi(5)(2)/8} + \ldots + 8e^{-i2\pi(5)(6)/8} + 4e^{-i2\pi(5)(7)/8}$$

$$= 10A + 15F + 20C + 25H + 5E + 30B + 8G + 4D$$

$$F_6 = 10e^{-i2\pi(6)(0)/8} + 15e^{-i2\pi(6)(1)/8} + 20e^{-i2\pi(6)(2)/8} + \ldots + 8e^{-i2\pi(6)(6)/8} + 4e^{-i2\pi(6)(7)/8}$$

$$= 10A + 15G + 20E + 25C + 5A + 30G + 8E + 4C = \boxed{-13 + 16i}$$

$$F_7 = 10e^{-i2\pi(7)(0)/8} + 15e^{-i2\pi(7)(1)/8} + 20e^{-i2\pi(7)(2)/8} + \ldots + 8e^{-i2\pi(7)(6)/8} + 4e^{-i2\pi(7)(7)/8}$$

$$= 10A + 15H + 20G + 25F + 5E + 30D + 8C + 4B$$

TABLE 4.1
Summary of some important properties of the 2-D Fourier transform.

Property	Expression(s)
Fourier transform	$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$ ← average all vals then divide by total # pixel?
Inverse Fourier transform	$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$
Polar representation	$F(u,v) = \lvert F(u,v)\rvert e^{-j\phi(u,v)}$
Spectrum	$\lvert F(u,v)\rvert = \left[R^2(u,v) + I^2(u,v)\right]^{1/2}$, $R = \mathrm{Real}(F)$ and $I = \mathrm{Imag}(F)$
Phase angle	$\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right]$
Power spectrum	$P(u,v) = \lvert F(u,v)\rvert^2$
Average value	$\bar{f}(x,y) = F(0,0) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$ ← avg of all pixels but with frequency = 0
Translation	$f(x,y)e^{j2\pi(u_0x/M+v_0y/N)} \Leftrightarrow F(u-u_0, v-v_0)$ $f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0/M+vy_0/N)}$ When $x_0 = u_0 = M/2$ and $y_0 = v_0 = N/2$, then $f(x,y)(-1)^{x+y} \Leftrightarrow F(u-M/2, v-N/2)$ $f(x-M/2, y-N/2) \Leftrightarrow F(u,v)(-1)^{u+v}$

If you scale

Conjugate symmetry	$F(u, v) = F^*(-u, -v)$ $\lvert F(u, v)\rvert = \lvert F(-u, -v)\rvert$
Differentiation	$\dfrac{\partial^n f(x, y)}{\partial x^n} \Leftrightarrow (ju)^n F(u, v)$ $(-jx)^n f(x, y) \Leftrightarrow \dfrac{\partial^n F(u, v)}{\partial u^n}$
Laplacian	$\nabla^2 f(x, y) \Leftrightarrow -(u^2 + v^2)F(u, v)$
Distributivity	$\Im[f_1(x, y) + f_2(x, y)] = \Im[f_1(x, y)] + \Im[f_2(x, y)]$ $\Im[f_1(x, y) \cdot f_2(x, y)] \neq \Im[f_1(x, y)] \cdot \Im[f_2(x, y)]$
Scaling	$af(x, y) \Leftrightarrow aF(u, v), f(ax, by) \Leftrightarrow \dfrac{1}{\lvert ab\rvert} F(u/a, v/b)$
Rotation rotate img = rotate forier trans	$x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$ $f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$
Periodicity	$F(u, v) = F(u + M, v) = F(u, v + N) = F(u + M, v + N)$ $f(x, y) = f(x + M, y) = f(x, y + N) = f(x + M, y + N)$
Separability	See Eqs. (4.6-14) and (4.6-15). Separability implies that we can compute the 2-D transform of an image by first computing 1-D transforms along each row of the image, and then computing a 1-D transform along each column of this intermediate result. The reverse, columns and then rows, yields the same result.

if you scale the input vals of input → output is also scales

Linear

space ↓ = freq ↑ . space ↑ = freq ↓

spacial component

- If columns repeat itself after 30 pixels and another row repeats itself every 10 pxls → 10 pxl row is higher freq

- If you stretch out that row you lower the freq

↑ space ⇒ ↓ freq

TABLE 4.1 *(continued)*

Property	Expression(s)
Computation of the inverse Fourier transform using a forward transform algorithm	$\frac{1}{MN} f^*(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u, v) e^{-j2\pi(ux/M + vy/N)}$ This equation indicates that inputting the function $F^*(u, v)$ into an algorithm designed to compute the forward transform (right side of the preceding equation) yields $f^*(x, y)/MN$. Taking the complex conjugate and multiplying this result by MN gives the desired inverse.
Convolution†	$f(x, y) * h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) h(x - m, y - n)$
Correlation†	$f(x, y) \circ h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m, n) h(x + m, y + n)$
Convolution theorem†	Spacial: kernal slide needed Freq: just multiply pixels!!! $f(x, y) * h(x, y) \Leftrightarrow F(u, v) H(u, v)$; (convolution in one domain → multiplication in another) $f(x, y) h(x, y) \Leftrightarrow F(u, v) * H(u, v)$ • Dont need to slide Kernal!! • in freq dom: can just do pixel·pixel!!!
Correlation theorem†	$f(x, y) \circ h(x, y) \Leftrightarrow F^*(u, v) H(u, v)$; $f^*(x, y) h(x, y) \Leftrightarrow F(u, v) \circ H(u, v)$

TABLE 4.1
(continued)

Some useful FT pairs:

Impulse	$\delta(x, y) \Leftrightarrow 1$
Gaussian	$A\sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2(x^2+y^2)} \Leftrightarrow Ae^{-(u^2+v^2)/2\sigma^2}$
Rectangle	$\mathrm{rect}[a, b] \Leftrightarrow ab\dfrac{\sin(\pi ua)}{(\pi ua)}\dfrac{\sin(\pi vb)}{(\pi vb)}e^{-j\pi(ua+vb)}$
Cosine	$\cos(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow \frac{1}{2}\left[\delta(u + u_0, v + v_0) + \delta(u - u_0, v - v_0)\right]$
Sine	$\sin(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow j\frac{1}{2}\left[\delta(u + u_0, v + v_0) - \delta(u - u_0, v - v_0)\right]$

† Assumes that functions have been extended by zero padding.

TABLE 4.1
(continued)