

Fast Fourier Transform

Background on Fourier series

- It can be shown that any continuous function $x(t)$ defined over $T_1 \leq t \leq T_2$ can be expressed as the following infinite sum

$$x(t) = \sum_{-\infty}^{\infty} z_k (\cos(k\omega_0 t) + i \sin(k\omega_0 t)) = x(t) = \sum_{-\infty}^{\infty} z_k \exp(ik\omega_0 t)$$

z_k are complex numbers $x + iy$, $i = \sqrt{-1}$

$$\omega_0 = 2\pi / (T_2 - T_1)$$

- We call the infinite sum a *Fourier series*

Deriving a Fourier Series

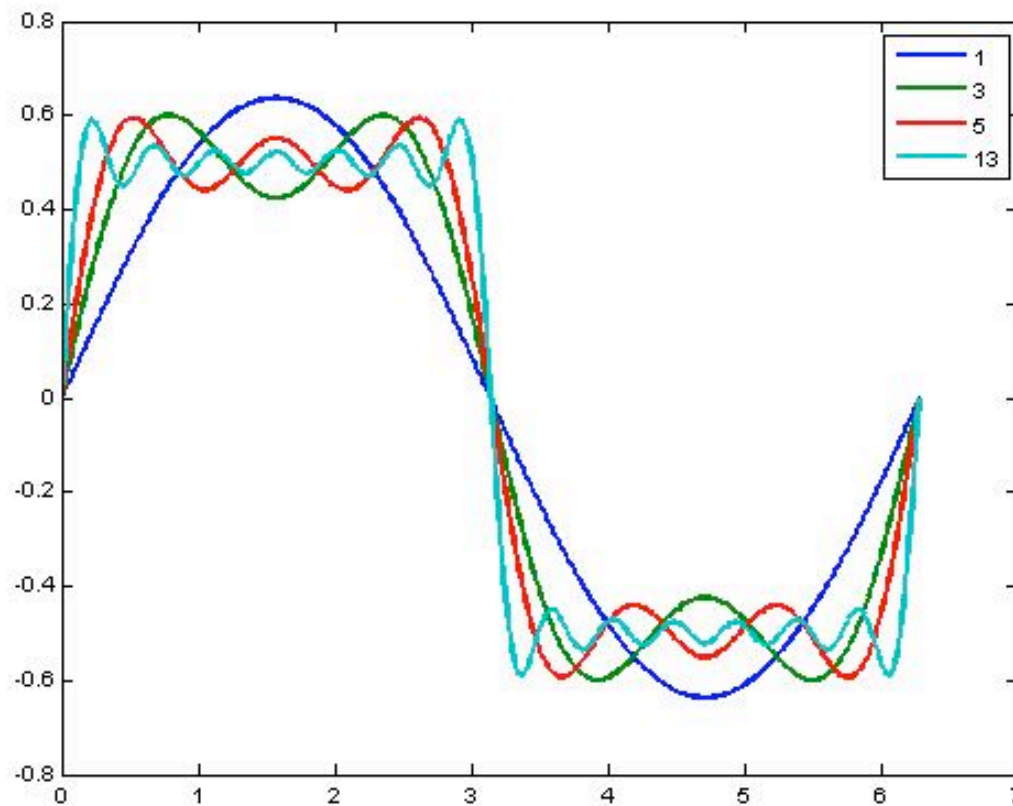
- The coefficients z_k are called the ***complex amplitude of the k^{th} component***
- They may be obtained by the following formula

$$z_k = \frac{1}{T} \int_{-\infty}^{\infty} e^{-ik\omega_0 t} x(t) dt$$

- For example, the square wave may be expressed as a sum of odd harmonics:

$$A_1 * \sin(x) + A_2 * \sin(3x) + A_3 * \sin(5x) + \dots$$

Fourier Series approximation of a square wave



$$(2/\pi) \sum_{k \text{ odd}} \sin(kx)/k$$

Fourier transform

- For a continuous signal defined from $+\infty$ to $-\infty$ there is a *continuous Fourier spectrum*
- This is the *Fourier transform*

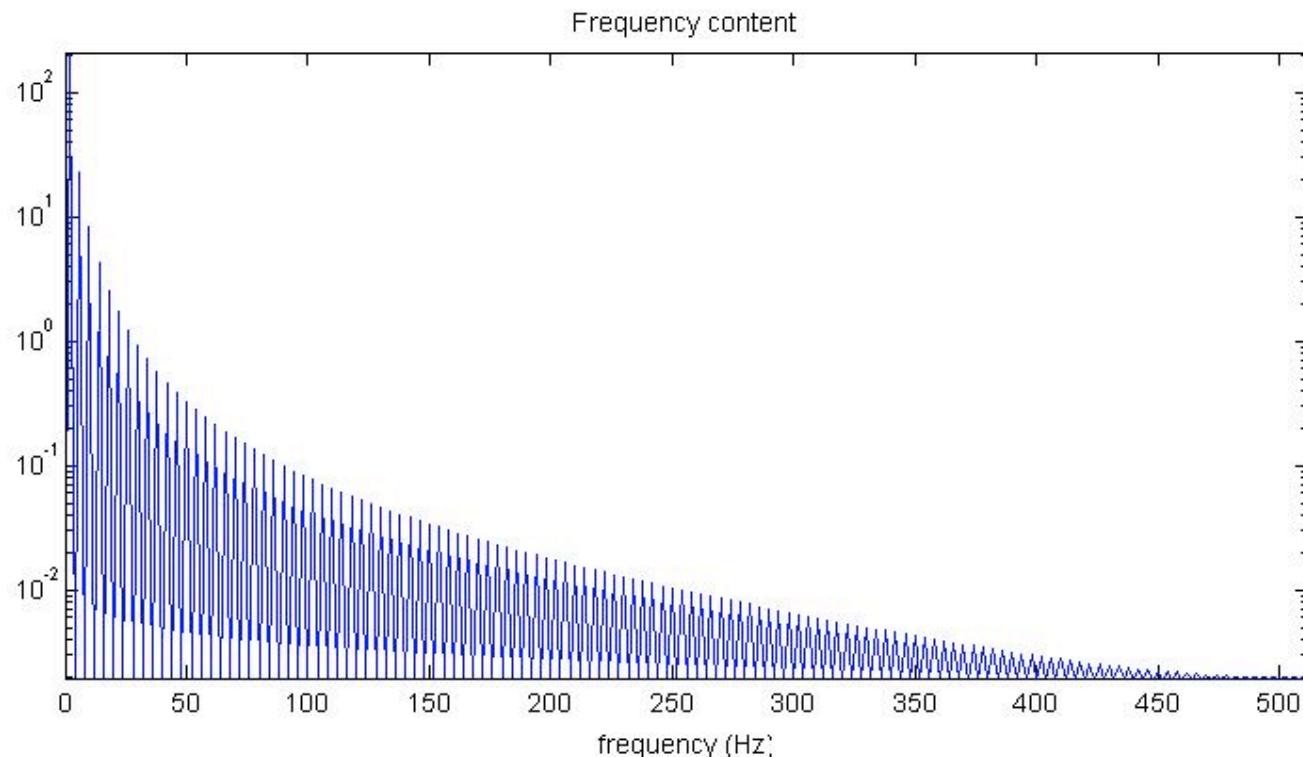
$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt, \text{ where } e^{-i\omega t} = \cos(\omega t) - \mathbf{i} \sin(\omega t)$$

- Whereas the *harmonics* corresponding to the z_k have frequencies that are discrete multiples of ω_0 the harmonics in the Fourier spectrum have all frequencies between $+\infty$ and $-\infty$
- We can recover the original function with the inverse transform

$$x(t) = (2\pi)^{-1} \int_{-\infty}^{\infty} X(\omega) e^{i\omega t} d\omega$$

Fourier transform of the square wave

Using the Fourier transform, we can determine the coefficients $z_k = \frac{1}{T} \int_{-\infty}^{\infty} (\cos(k\omega_0 t) + \mathbf{j} \sin(k\omega_0 t)) S(t) dt$



Computing Fourier transform

- For a continuous function we can determine the coefficients using the ***Fourier transform***

$$z_k = \frac{1}{T} \int_{-\infty}^{\infty} e^{-ik\omega_0 t} x(t) dt$$

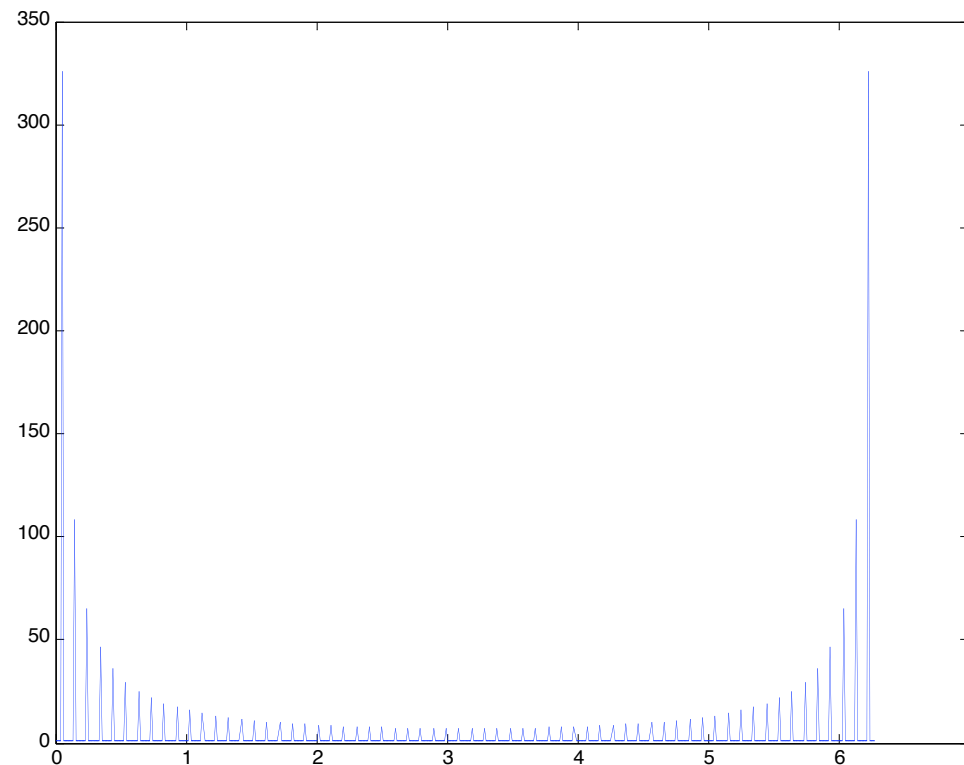
- The ***Discrete Fourier Transform*** determines the coefficient for a discrete function defined on uniformly spaced sampling points: $O(N^2)$ algorithm

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N} \quad x(n) = \sum_{k=0}^{N-1} X(k) e^{i2\pi kn/N}$$

- Fast Fourier Transform*** (FFT), Cooley and Tukey (1965)
- Divide and conquer, $O(N \lg N)$

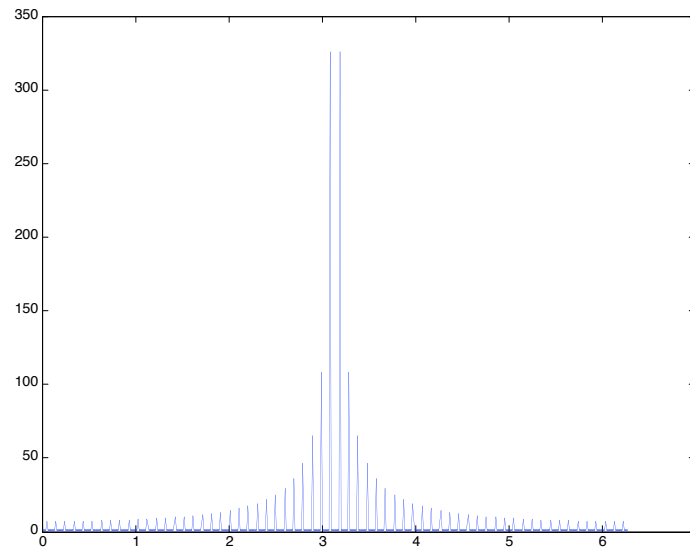
Using the Discrete Fourier transform

- Consider the spectrum of the square wave



Shifting

- The FT is centered about the origin
- But the DFT is centered about $N/2$
- We need to correct with a circular shift operation
- Or, multiply by $(-1)^k$ prior to taking the transform



FFT

- Cooley and Tukey (1965), $O(N \lg N)$
- Divide and conquer $\rightarrow$ iterative computation
- $K = 0:N-1$, $M = N / 2$

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}nk}$$

$$X_k = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k}$$

$$= \sum_{m=0}^{M-1} e_m e^{-\frac{2\pi i}{M}mk} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{M-1} o_m e^{-\frac{2\pi i}{M}mk}$$

$$= \begin{cases} E_k + e^{-\frac{2\pi i}{N}k} O_k & \text{if } k < M \\ E_{k-M} - e^{-\frac{2\pi i}{N}(k-M)} O_{k-M} & \text{if } k \geq M \end{cases}$$

Recursive formulation

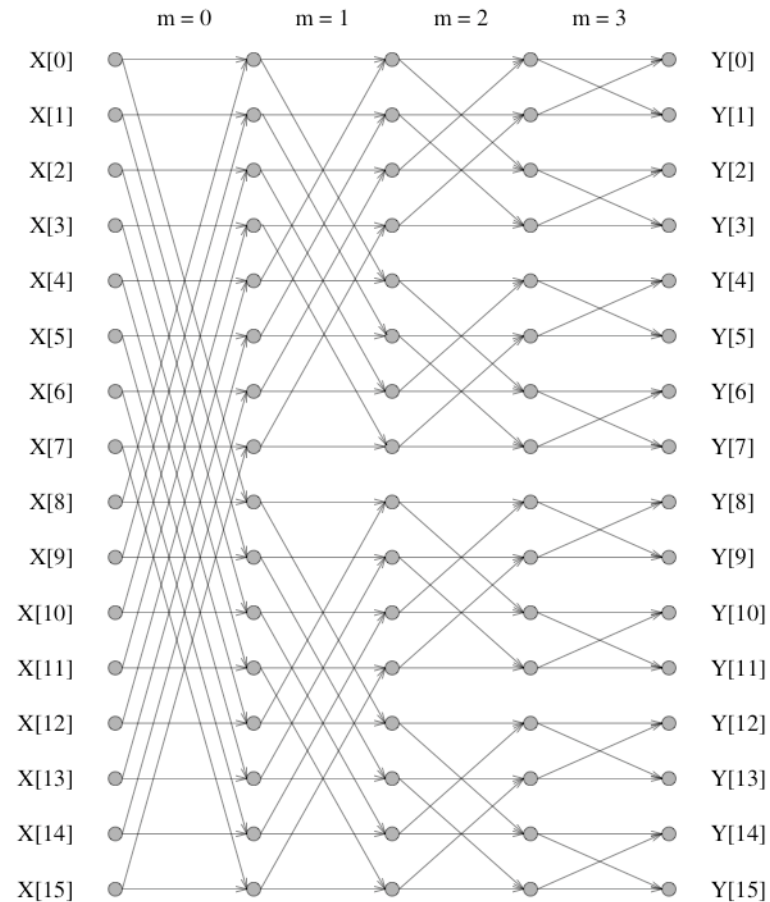
```
procedure R_FFT( $X, Y, n, \omega$ )  
if ( $n = 1$ ) then  $Y[0] := X[0]$  else  
begin  
    R_FFT( $\langle X[0], X[2], \dots, X[n-2] \rangle, \langle Q[0], Q[1], \dots, Q[n/2] \rangle, n/2, \omega^2$ );  
    R_FFT( $\langle X[1], X[3], \dots, X[n-1] \rangle, \langle T[0], T[1], \dots, T[n/2] \rangle, n/2, \omega^2$ );  
    for  $i := 0$  to  $n - 1$  do  
         $Y[i] := Q[i \bmod (n/2)] + \omega^i T[i \bmod (n/2)];$   
end R_FFT
```

Iterative formulation

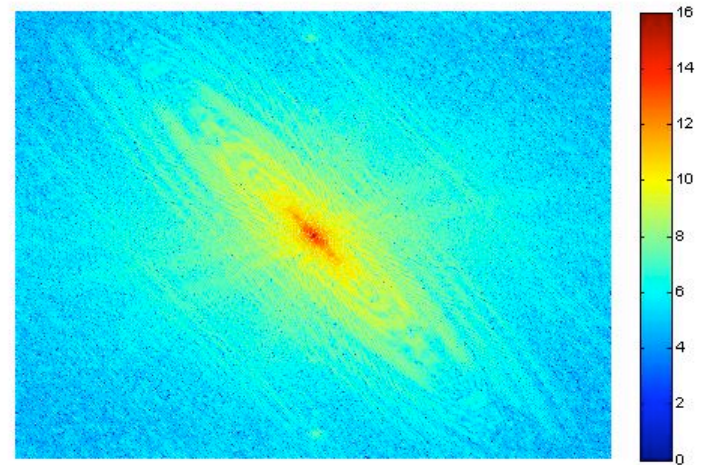
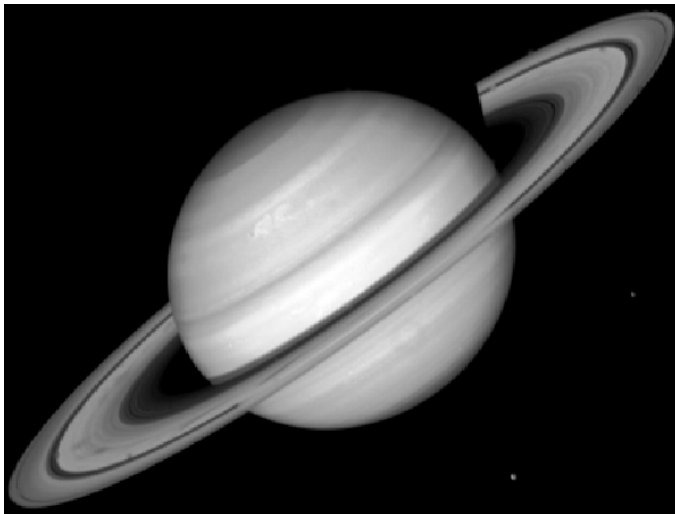
```
1.  procedure ITERATIVE_FFT( $X, Y, n$ )
2.  begin
3.       $r := \log n$ ;
4.      for  $i := 0$  to  $n - 1$  do  $R[i] := X[i]$ ;
5.      for  $m := 0$  to  $r - 1$  do          /* Outer loop */
6.          begin
7.              for  $i := 0$  to  $n - 1$  do  $S[i] := R[i]$ ;
8.              for  $i := 0$  to  $n - 1$  do /* Inner loop */
9.                  begin
10.                      $j := (b_0 \dots b_{m-1} 0 b_{m+1} \dots b_{r-1})$ ;
11.                      $k := (b_0 \dots b_{m-1} 1 b_{m+1} \dots b_{r-1})$ ;
12.                      $R[i] := S[j] + S[k] \times \omega^{(b_m b_{m-1} \dots b_0 0 \dots 0)}$ ;
13.                 endfor; /* Inner loop */
14.             endfor; /* Outer loop */
15.          for  $i := 0$  to  $n - 1$  do  $Y[i] := R[i]$ ;
16.  end ITERATIVE_FFT
```

FFT “Butterfly graph”

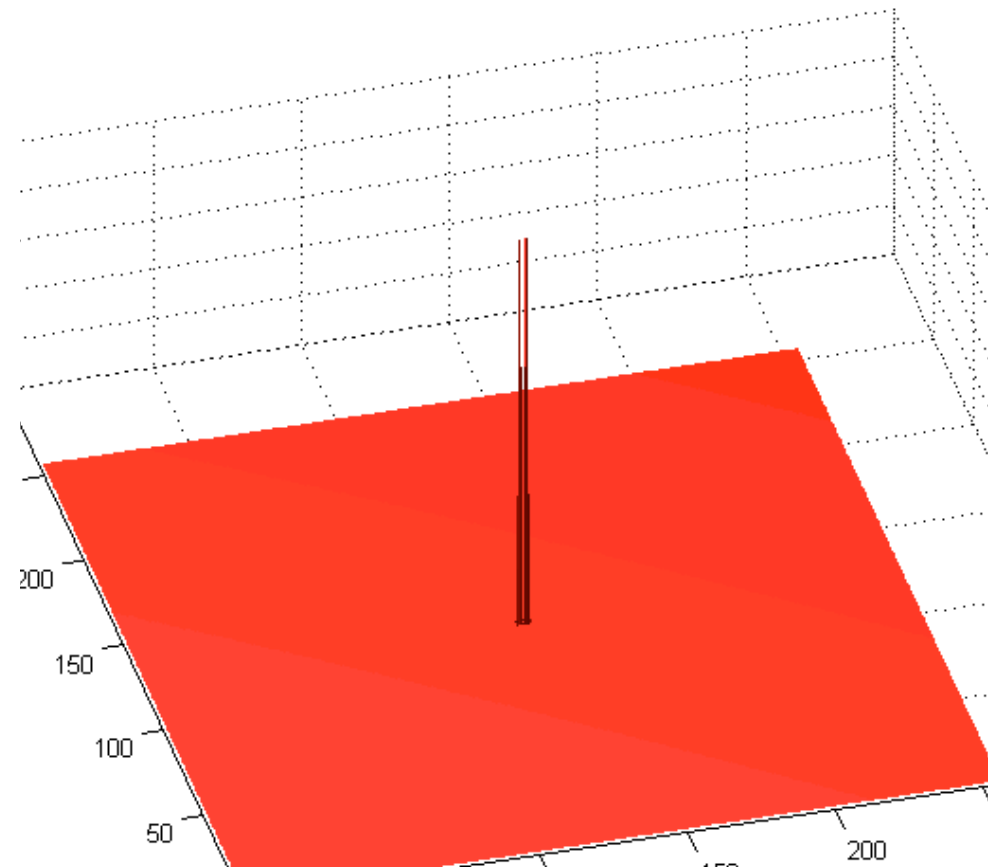
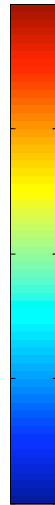
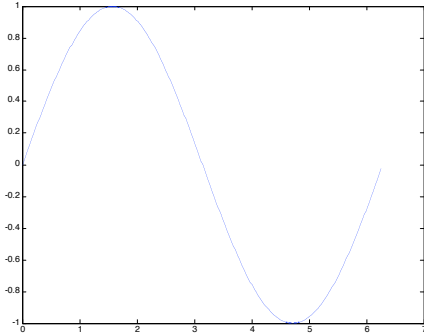
- $\lg N$ stages
- At each iteration i , apply a butterfly operation to combine pairs of elements spaced 2^{N-i} units apart
- “Twiddle” factors



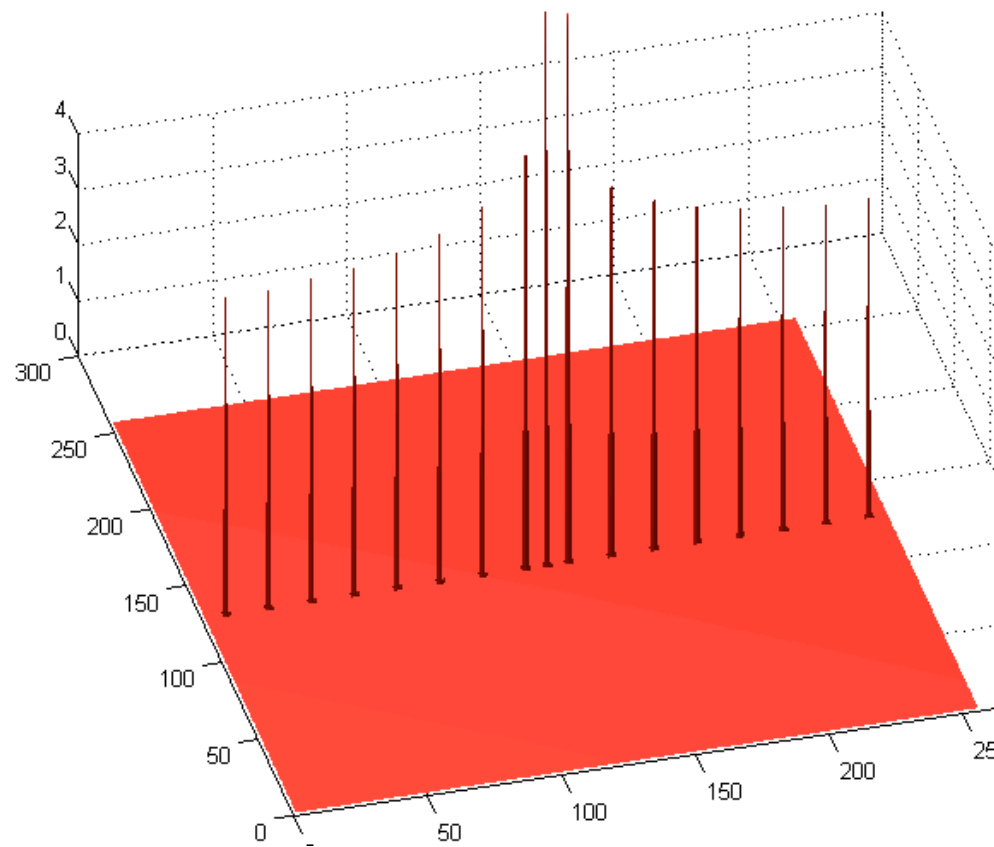
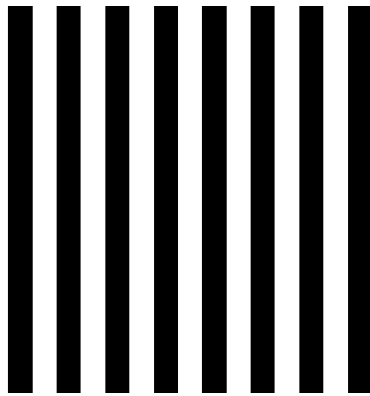
A 2D Discrete Fourier Transform



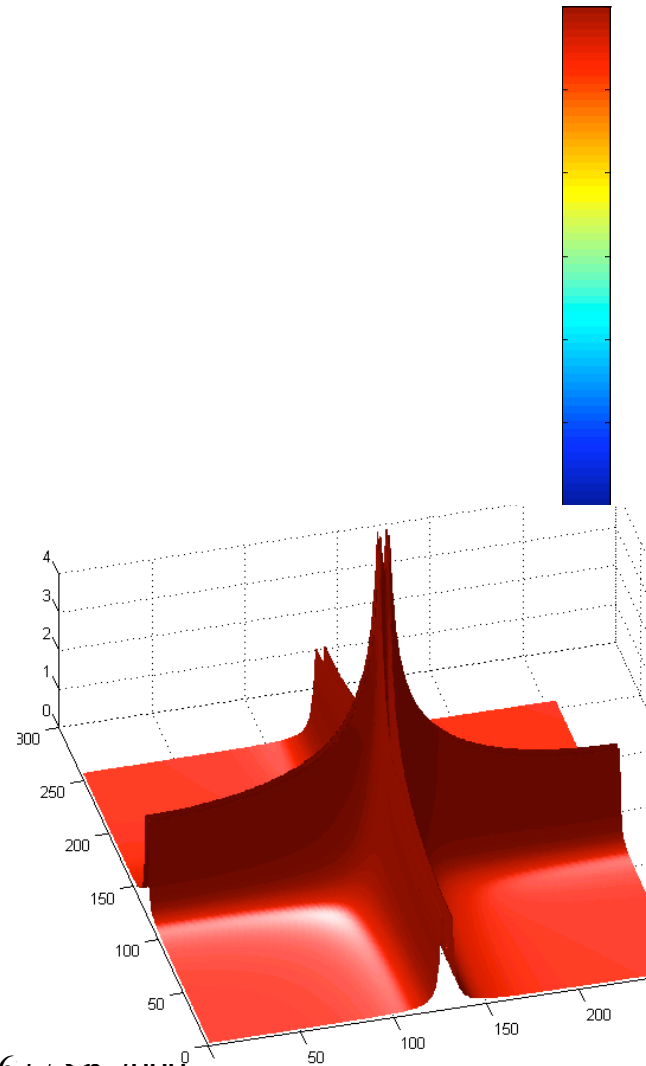
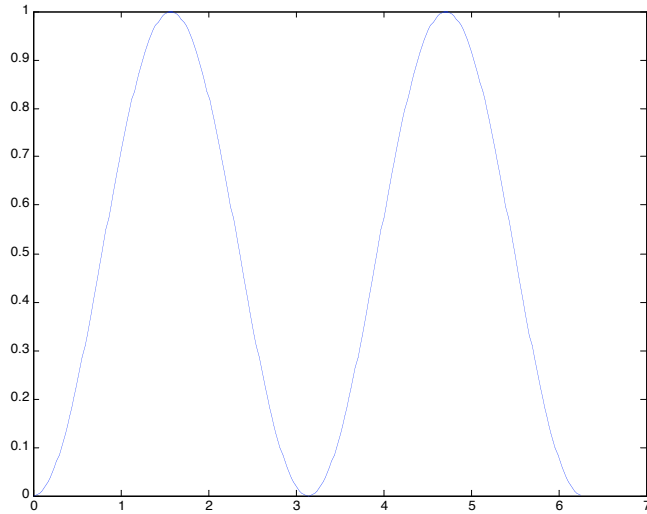
1D Spatial Frequencies



1D Spatial Frequencies



2D Spatial Frequencies



Implementing the 2D FFT

- The 2D FFT is *separable*
- We may express it in terms of 1D FFTs
- We take *row transforms*
 - Apply the 1D FFT algorithm to each row
- Next take *column transforms*
 - Apply the 1D FFT algorithm to each column

A few details

- The data end up in bit-reversed order
 - Thus, element 17=010001 $\rightarrow$ 100010 = 34
- We use libraries to compute the FFTs
 - MKLIB, ESSL
- Parallel libraries, e.g. PESSL
- Other algorithms
 - FFTW “Fastest FFT in the West:” www.fftw.org
 - “Faster FFTs via Architecture Cognizance,” K. Gatlin and L. Carter, PACT 2000
<http://www.cs.ucsd.edu/users/carter/Papers/pact00.ps>