# Maths for AI

## Lecture 2: Bases, Transformation and the DFT

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

School of Mathematical Sciences and AIML,
University of Adelaide

March 15, 2022

# Discrete Fourier Transformation

Continuous Fourier transform $F(s) = \int_{-\infty}^{\infty} f(t) e^{-i2\pi st}\, dt$

But note that for a finite length, discrete-time signal, it can be written as

$$x(t) = \sum_{n=0}^{N-1} f(nt_s)\delta(t - nt_s)$$

The Fourier transform can then be written

$$X(s) = \sum_{n=0}^{N-1} f(nt_s) e^{-i2\pi snt_s}$$

The result is simpler to compute, but its still redundant.

# Discrete Fourier Transformation

If we have $N$ data points, we would like a (frequency domain) representation that only needs $N$ data points as well. Hence no redundancy.

Use $s = \frac{k}{Nt_s}$ for $k = 0, 1, \ldots, N-1$ and we get

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

where $x(n)$ are the $N$ discrete samples from the continuous time process.

This is the Discrete Fourier Transform **(DFT)**

# Inverse DFT

DFT

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

Inverse DFT (IDFT)

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

# Examples (i)

Take $x(n) = (1, 0, 0, 0)$

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

$$\begin{aligned}
X(0) &= e^{-i2\pi 0/4} &= 1 \\
X(1) &= e^{-i2\pi 0/4} &= 1 \\
X(2) &= e^{-i2\pi 0/4} &= 1 \\
X(3) &= e^{-i2\pi 0/4} &= 1
\end{aligned}$$

So $X(k) = (1, 1, 1, 1)$

# Examples (i) IDFT

Take $X(k) = (1, 1, 1, 1)$

$$
\begin{array}{rcll}
x(n) & = & \frac{1}{N}\sum_{n=0}^{N-1} X(k) e^{i2\pi kn/N} & \\
x(0) & = & \frac{1}{4}\left(e^{-i2\pi 0/4} + e^{-i2\pi 0/4} + e^{-i2\pi 0/4} + e^{-i2\pi 0/4}\right) & \\
 & = & \frac{1}{4}\left(1 + 1 + 1 + 1\right) & = \quad 1 \\
x(1) & = & \frac{1}{4}\left(e^{-i2\pi 0/4} + e^{-i2\pi 1/4} + e^{-i2\pi 2/4} + e^{-i2\pi 3/4}\right) & \\
 & = & \frac{1}{4}\left(1 + i - 1 - i\right) & = \quad 0 \\
x(2) & = & \frac{1}{4}\left(e^{-i2\pi 0/4} + e^{-i2\pi 2/4} + e^{-i2\pi 4/4} + e^{-i2\pi 6/4}\right) & \\
 & = & \frac{1}{4}\left(1 - 1 + 1 - 1\right) & = \quad 0 \\
x(3) & = & \frac{1}{4}\left(e^{-i2\pi 0/4} + e^{-i2\pi 3/4} + e^{-i2\pi 6/4} + e^{-i2\pi 9/4}\right) & \\
 & = & \frac{1}{4}\left(1 - i - 1 + i\right) & = \quad 0
\end{array}
$$

So $x(n) = (1, 0, 0, 0)$

# Examples (ii)

Take $x(n) = (0, 1, 0, 0)$

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

$$\begin{aligned}
X(0) &= e^{-i2\pi 0/4} &&= 1 \\
X(1) &= e^{-i2\pi 1/4} &&= e^{-i\pi/2} &&= -i \\
X(2) &= e^{-i2\pi 2/4} &&= e^{-i\pi} &&= -1 \\
X(3) &= e^{-i2\pi 3/4} &&= e^{-i\pi 3/2} &&= i
\end{aligned}$$

So $X(k) = (1, -i, -1, i)$

# Examples (iii)

Take $x(n) = (1, 1, 0, 0)$

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

$$\begin{aligned}
X(0) &= e^{-i2\pi 0/4} + e^{-i2\pi 0/4} &&= 1 + 1 &&= 2 \\
X(1) &= e^{-i2\pi 0/4} + e^{-i2\pi 1/4} &&= e^0 + e^{-i\pi/2} &&= 1 - i \\
X(2) &= e^{-i2\pi 0/4} + e^{-i2\pi 2/4} &&= e^0 + e^{-i\pi} &&= 0 \\
X(3) &= e^{-i2\pi 0/4} + e^{-i2\pi 3/4} &&= e^0 + e^{-i\pi 3/2} &&= 1 + i
\end{aligned}$$

So $X(k) = (2, 1 - i, 0, 1 + i)$

# DFT basis

We are simply changing basis

The basis vectors are a discrete set of sin and cosine functions.

Note, now we are operating in a finite dimensional space $\mathbb{R}^N$, so we can write the transform as

$$X = Ax \qquad \text{analysis}$$

The inverse transform is just

$$x = A^{-1}X \qquad \text{synthesis}$$

Where both $x$ and $X$ are just vectors in $\mathbb{R}^N$.

# DFT transform matrix

$$X = Ax$$

$$A = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-i2\pi 1/N} & e^{-i2\pi 2/N} & \cdots & e^{-i2\pi(N-1)/N} \\ 1 & e^{-i2\pi 2/N} & e^{-i2\pi 4/N} & \cdots & e^{-i2\pi 2(N-1)/N} \\ 1 & e^{-i2\pi 3/N} & e^{-i2\pi 6/N} & \cdots & e^{-i2\pi 3(N-1)/N} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{-i2\pi(N-1)/N} & e^{-i2\pi 2(N-1)/N} & \cdots & e^{-i2\pi(N-1)(N-1)/N} \end{pmatrix}$$

# Examples (i)

Take $x(n) = (1, 0, 0, 0)$

$$X = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-i2\pi 1/4} & e^{-i2\pi 2/4} & e^{-i2\pi 3/4} \\ 1 & e^{-i2\pi 2/4} & e^{-i2\pi 4/4} & e^{-i2\pi 6/4} \\ 1 & e^{-i2\pi 3/4} & e^{-i2\pi 6/4} & e^{-i2\pi 9/4} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

# Frequency resolution

Frequencies of basis functions are $k = 0, 1, 2, \ldots, (N-1)$ cycles over the data set. If the data set has $N$ samples at sampling frequency $f_s$, then its duration is $T = N/f_s$. To convert from data units to absolute units, we take $k/T = \frac{kf_s}{N}$
Frequency resolution is $\frac{f_s}{N}$

- higher sampling frequencies reduce frequency resolution
- longer data, improves frequency resolution

# Getting units right

Note that absolute frequency depends on sample frequency $f_s$, so we need to convert.
The component $X(m)$ will correspond to frequency

$$X(m) \equiv F\left(\frac{mf_s}{N}\right)$$

Output magnitude of DFT will be amplitude of sin wave signal $A$ times $N/2$. Alternative definitions of DFT exist

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

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

# FFT

- We don't actually perform the DFT this way
- We use the Fast Fourier Transform (FFT)
- One of the cleverest algorithms out there

# Matlab

Note, indexes in Matlab run from 1 to $N$ (not 0 to $N-1$).

$$\text{fft}(x(n)) = X(k) = \sum_{n=1}^{N} x(n) e^{-i2\pi(k-1)(n-1)/N}, \quad k = 1, \ldots, N.$$

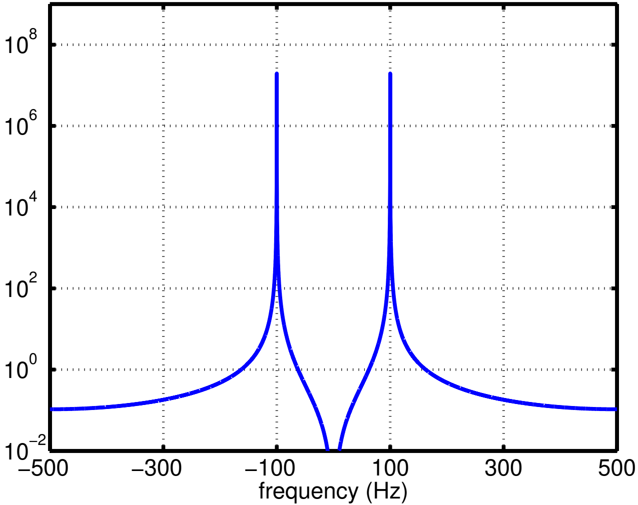
$$\text{ifft}(X(k)) = x(n) = \frac{1}{N} \sum_{k=1}^{N} X(k) e^{i2\pi(k-1)(n-1)/N}, \quad n = 1, \ldots, N.$$

$X(1)$ is the DC term, $X(n)$ is the $f_s$ term. To plot symmetric power spectrum use, e.g.

```
f_s = 1000;
f_0 = 100;
x = 1:1/f_s:10;
y = sin(2*pi*f_0*x);
semilogy(-f_s/2+f_s/N:f_s/N:f_s/2, abs(fftshift(fft(y))).^2);
set(gca, 'ylim', 10.^[-2 9]);
xlabel('frequency (Hz)');
```

# Matlab example

matlab_ex_1.m

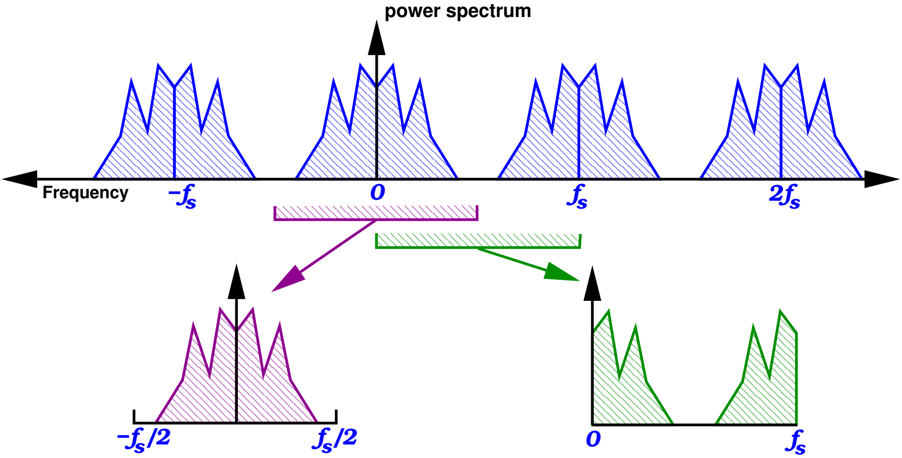


# Do It

- Use Colab and `torch.fft` to replicate the above Matlab example
  - `fftshift` is your friend
  - `fftfreq` can help you ge the x-axis units correct
- Use Colab to analyse an audio signal
  See `audio_example.ipynb` for the various bits and pieces needed.

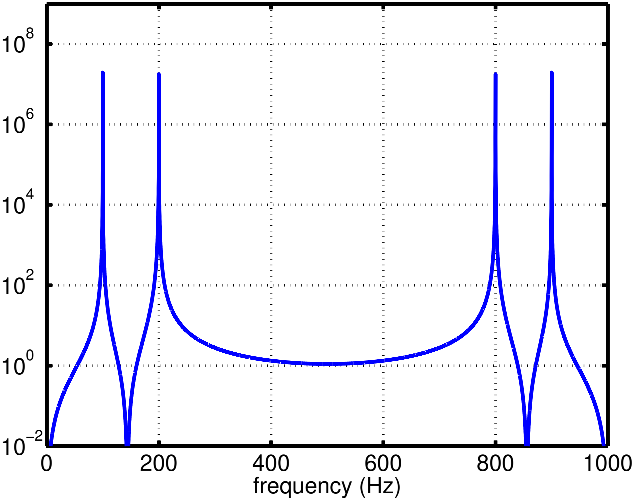
# Symmetry

Discrete power spectrum is **even** and **periodic** so we can display in a number of ways.



# Matlab example 2

matlab_ex_2.m



# Properties of the DFT

Mostly the same as Continuous FT

- invertible
- no redundancy so it is efficient
- Linearity: $ax_1(n) + bx_2(n) \rightarrow aX_1(k) + bX_2(k)$
- *Time shift:* $x(n - n_0) \rightarrow X(k)e^{-i2\pi k n_0}$
- Time scaling: a bit more complicated!
- Duality: a bit more complicated!
- Frequency shift: $x(n)e^{-i2\pi k_0 n} \rightarrow X(k - k_0)$
- Convolution: $x_1(n) * x_2(n) \rightarrow X_1(k)X_2(k)$

Now $n$ and $k$ are integers, with the result that we are missing properties related to derivatives.

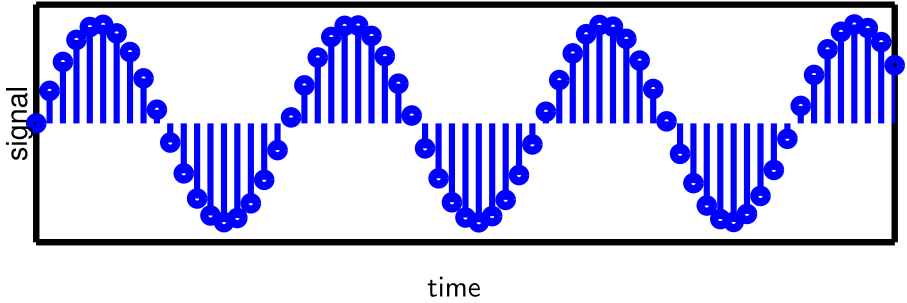
# Properties of the DFT

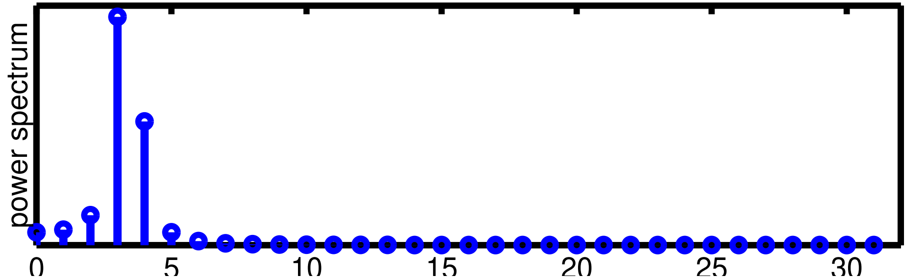
There are some new properties unique to DFTs

- Leakage that fits exactly our discrete frequencies
- Padding (packing)
- Similarity (discrete version of time scaling)

See below for details.

# Leakage example



# Properties of the DFT: Leakage

DFT is different from the continuous time FT is that the DFT suffers from **Leakage**.

- Unlike Continuous transform, DFT uses a finite number of frequencies.
- Not all signals fit this mold exactly: what happens to sinusoids with non-integral frequencies?
- Their power is spread over a few frequencies.
- Note we are representing the signal by a series of numbers $X(k)$ which represent the correlation of the signal to a particular sinusoid with freq. $k/N$,
- Note that, as the data gets longer, the frequency resolution improves

# DFT properties: padding

We can pad (or pack) a sequence with zeros to extend its length

$$y(n) = \begin{cases} x(n), & \text{if } 0 \leq n \leq N-1 \\ 0, & \text{if } N \leq n < KN \end{cases}$$

The resulting DFT is

$$\mathcal{F}\{y\} = Y(k) = X\left(\frac{k}{K}\right)$$

# Padding (packing) example (ii)

Data $x(n) = (0, 1, 0, 0)$ with transform $X(k) = (1, -i, -1, i)$
Pad to get $y(n) = (0, 1, 0, 0, 0, 0, 0, 0)$ then the DFT

$$Y(k) = \sum_{n=0}^{N-1} y(n) e^{-i2\pi kn/N}$$

$$\begin{aligned}
Y(0) &= e^{-i2\pi 0/8} &&= 1 \\
Y(1) &= e^{-i2\pi 1/8} &&= e^{-i\pi/4} &&= (1-i)/\sqrt{2} \\
Y(2) &= e^{-i2\pi 2/8} &&= e^{-i\pi/2} &&= -i \\
Y(3) &= e^{-i2\pi 3/8} &&= e^{-i\pi 3/4} &&= (-1-i)/\sqrt{2} \\
Y(4) &= e^{-i2\pi 4/8} &&= e^{-i\pi} &&= -1 \\
Y(5) &= e^{-i2\pi 5/8} &&= e^{-i\pi 5/4} &&= (-1+i)/\sqrt{2} \\
Y(6) &= e^{-i2\pi 6/8} &&= e^{-i\pi 3/2} &&= i \\
Y(7) &= e^{-i2\pi 7/8} &&= e^{-i\pi 7/4} &&= (1+i)/\sqrt{2}
\end{aligned}$$

# Padding (packing) example (ii)

Data $x(n) = (0, 1, 0, 0)$ with transform $X(k) = (1, -i, -1, i)$
Pad to get $y(n) = (0, 1, 0, 0, 0, 0, 0, 0)$ then the DFT

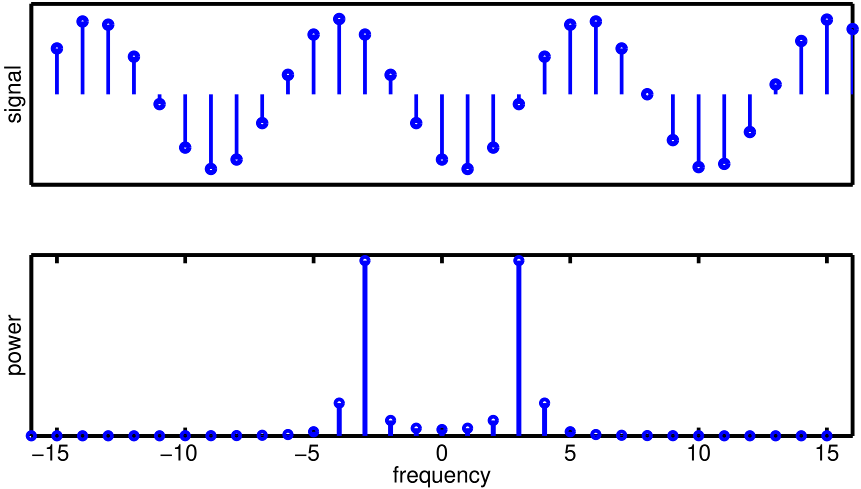
$$\begin{aligned} Y(0) &= X(0) \\ Y(2) &= X(1) \\ Y(4) &= X(2) \\ Y(6) &= X(3) \end{aligned}$$

So the relationship $Y(k) = X(k/2)$ holds, with $K = 2$, for even values of $k$.

Note we cannot derive $Y(k)$ for odd values of $k$, or if $K$ is not an integer, but the relationship still tells us how to scale the frequency units, when we pad.

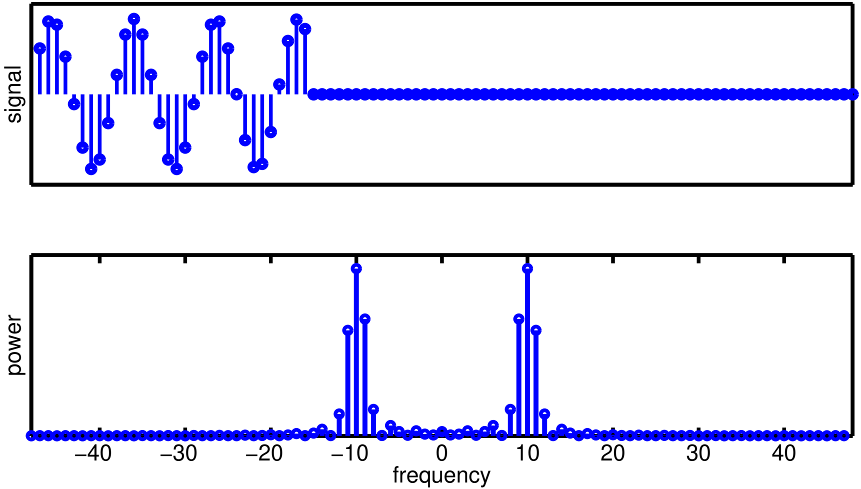
# Padding (packing) example

Original data length $N = 32$ (frequency $= 3.333$)



# Padding (packing) example

$K = 3$, new sequence length $KN = 96$. (frequency $= 10/\mathrm{K}$)



# DFT properties: similarity

We can interleave a sequence with zeros, e.g.

$$y(n) = \begin{cases} x(n/K), & \text{if } n = 0, K, 2K, \ldots, (N-1)K \\ 0, & \text{otherwise} \end{cases}$$

The resulting DFT is

$$\mathcal{F}\{y\} = Y(k) = \begin{cases} X(k) & k = 0, \ldots, N-1 \\ X(k-N) & k = N, \ldots, 2N-1 \\ \vdots & \\ X(k-(K-1)N) & k = (K-1)N, \ldots, KN-1 \end{cases}$$

# Similarity example (ii)

Data $x(n) = (0, 1, 0, 0)$ with transform $X(k) = (1, -i, -1, i)$
Interleave zeros to get $y(n) = (0, 0, 1, 0, 0, 0, 0, 0)$ then

$$
\begin{array}{rclclcl}
Y(k) & = & \sum_{n=0}^{N-1} y(n) e^{-i2\pi kn/N} & & & & \\
Y(0) & = & e^{-i2\pi 0/8} & = & 1 & & \\
Y(1) & = & e^{-i2\pi 2/8} & = & e^{-i\pi/2} & = & -i \\
Y(2) & = & e^{-i2\pi 4/8} & = & e^{-i\pi} & = & -1 \\
Y(3) & = & e^{-i2\pi 6/8} & = & e^{-i\pi 3/2} & = & i \\
Y(4) & = & e^{-i2\pi 8/8} & = & e^{-i2\pi} & = & 1 \\
Y(5) & = & e^{-i2\pi 10/8} & = & e^{-i\pi 5/2} & = & -i \\
Y(6) & = & e^{-i2\pi 12/8} & = & e^{-i\pi 3} & = & -1 \\
Y(7) & = & e^{-i2\pi 14/8} & = & e^{-i\pi 7/2} & = & i
\end{array}
$$

So $Y(k) = (1, -i, -1, i, 1, -i, -1, i)$ (or $X(k)$ repeated twice)

# Similarity application

Practical use: upsampling (interpolation)

> *We have a sequence sampled every $t_s$ seconds, e.g. at a rate $f_s = 1/t_s$, but we need a sequence sampled at rate $Kf_s$.*

Approach: produce a new sequence with $K - 1$ zeros interleaved between each original data point.

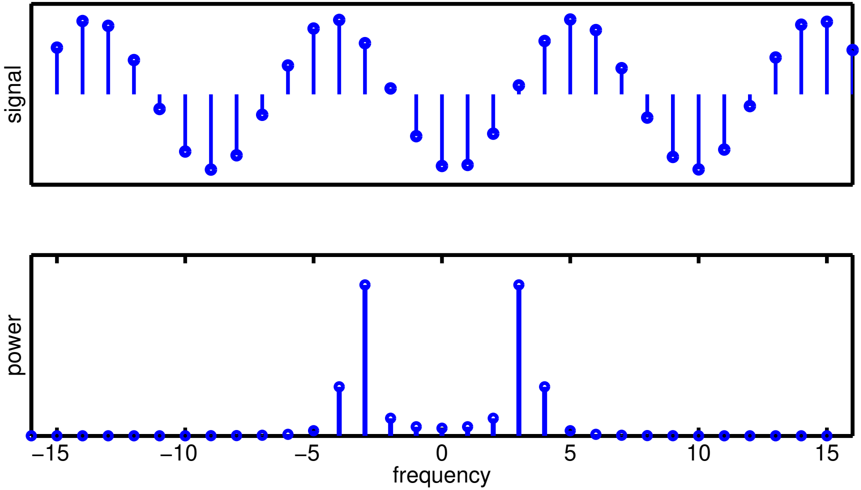
# Similarity application: upsampling

iven $K-1$ zeros interleaved between each original sample.

- max frequency in original data is $f_s/2$, with frequency resolution $f_s/N$, and $N/2$ points in frequency domain.
- upsampled data has max frequency $Kf_s/2$, with frequency resolution $f_s/N$, and $KN/2$ points in frequency domain.
- the frequency resolution doesn't change, but now we have $K$ repeats of the original spectrum at intervals $f_s/N$.
- to get a signal with the same original band-limited power-spectrum, we apply a low-pass filter, smoothing the data.

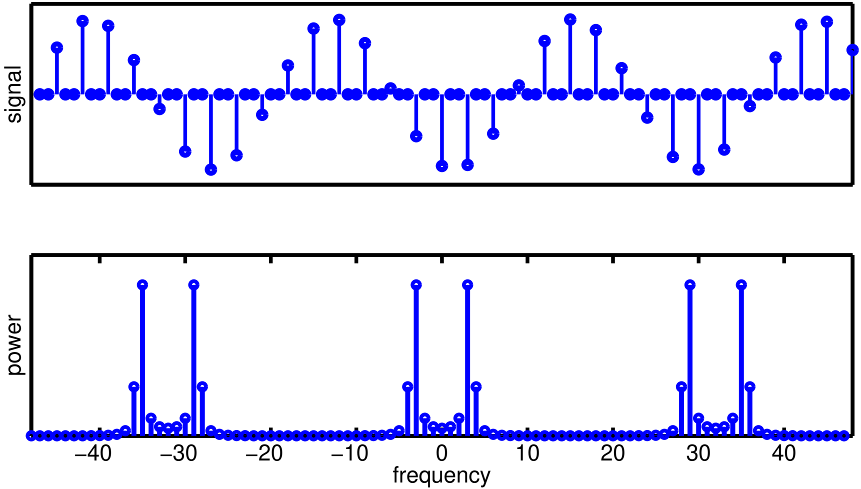
# Upsampling example

32 samples (frequency 3.4 cycles)



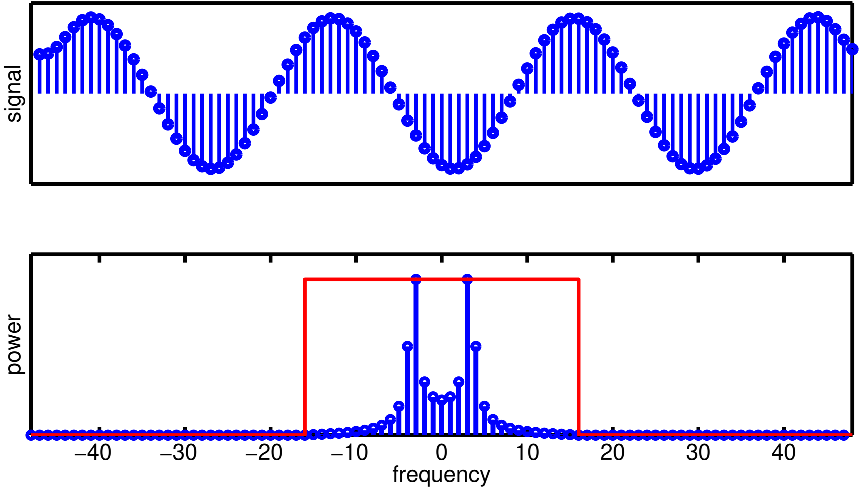
# Upsampling example

3 $\times$'s upsampled (96 samples)



# Upsampling example

low pass filter, then IDFT



# Upsampling tricks

Trick of the day: low-pass before upsampling.

- notionally, the filtering occurs after upsampling
- If filtering in the time domain however, $K - 1/K$ proportion of multiplies in the filter are by zero.
- can ignore these, but this is the same as low-pass before upsampling.

Let's revisit this later (after discussing filtering in more detail).

# Upsampling applications: audio

Oversampling CD or DVD players

- digital components are cheap
- analogue components are more expensive
- Digital to Analogue Conversion (DAC) is required in CD player
- want to make this as cheap as possible (for a given quality)

The trick

- upsample in the digital domain (where it is cheap)
- when we convert to analogue, we can use a simpler, cheaper analogue filter, to get the same results

# The DFT in 2D

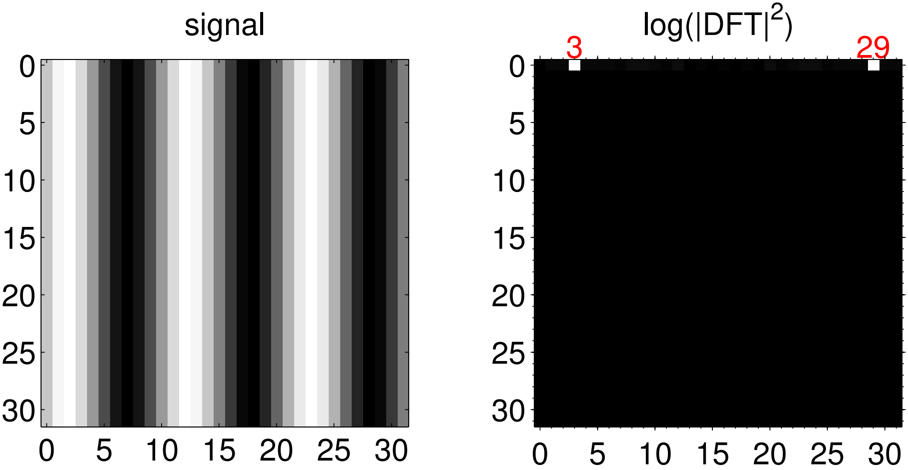
DFT

$$X(k_1,k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1,n_2) e^{-i2\pi k_1 n_1/N_1} e^{-i2\pi k_2 n_2/N_2},$$

- To compute it efficiently:
  1. compute 1D FFT along the rows
  2. then do a 1D FFT along the columns
- Called **row-column** algorithm
  - note that the order could change.
- naturally generalizes to higher dimensions

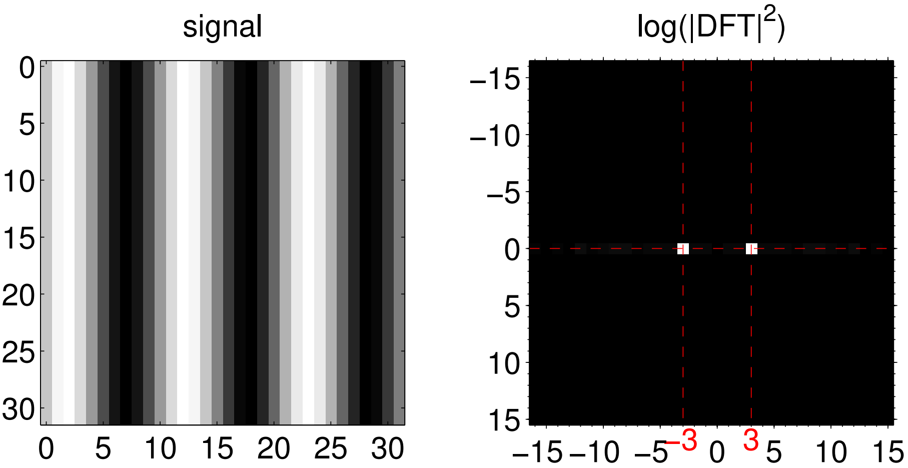
# Examples (i)

$x(n, k) = \sin\left(2\pi 3k / N\right)$



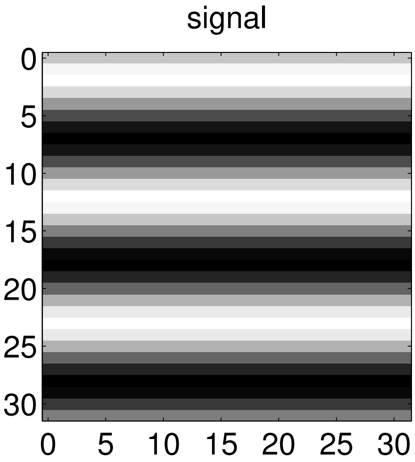
# Examples (i): `fftshift`

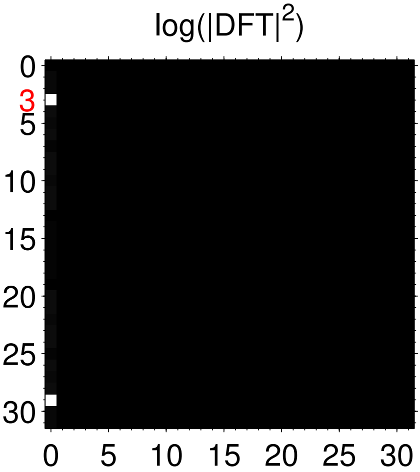
$x(n, k) = \sin(2\pi 3k / N)$



# Examples (ii)

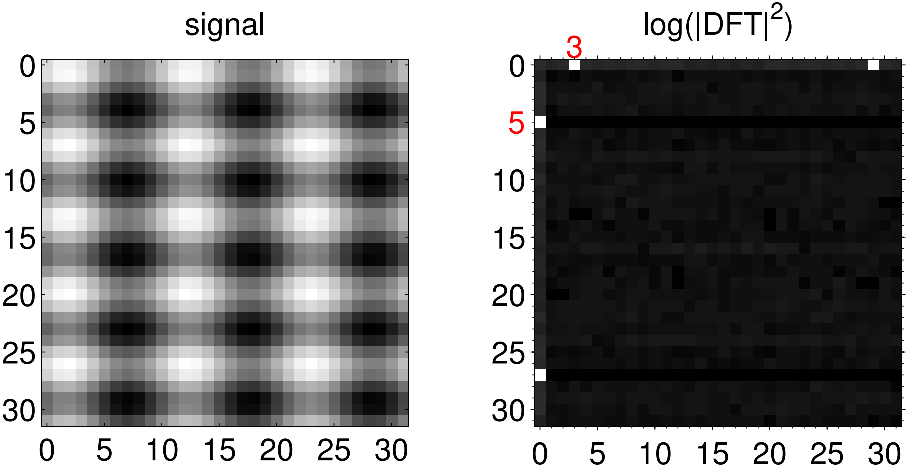
$x(n,k) = \sin(2\pi 3n/N)$



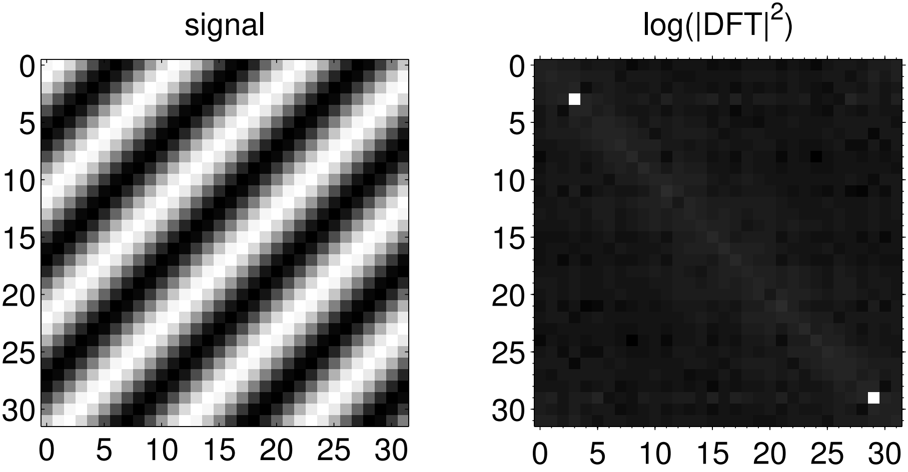
# Examples (iii): superposition

$x(n,k) = \sin(2\pi 5n/N) + \sin(2\pi 3k/N)$



# Examples (iv)

$x(n,k) = \sin\left(2\pi 3(n+k)/N\right)$



# DFT and symmetry

The symmetry of the 2D FT depends on the symmetry of the function.

$$
\begin{aligned}
F(-s,-v) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{i2\pi(sx+ty)}\, dx\, dy \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(-x,-y) e^{-i2\pi(sx+ty)}\, dx\, dy \\
&= \mathcal{F}\{f(-x,-y)\}
\end{aligned}
$$

As before (in 1D), but now we reflect through the origin.

- similar result to before relating complex conjugates etc.

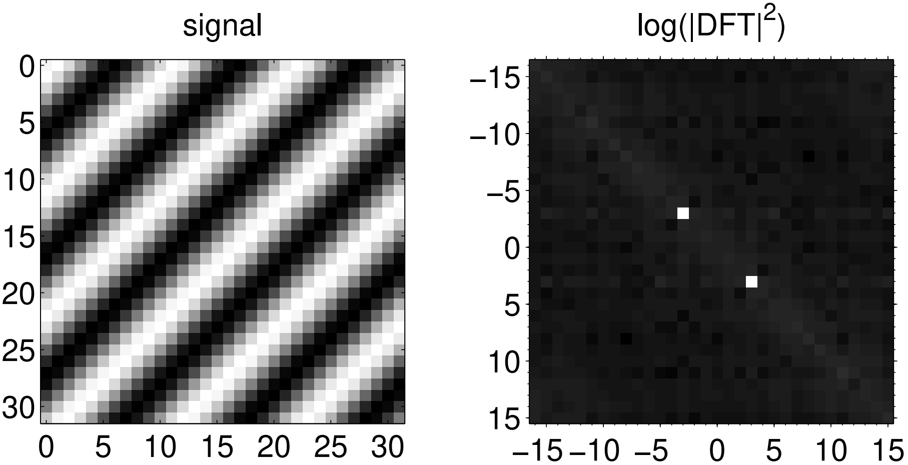
# DFT and symmetry

Power-spectrum of 2D DFT will be symmetric about the center (zero frequency).

- Equivalent to real time series produces even power-spectrum.
- In matlab, use `fftshift` to see the plots this way.

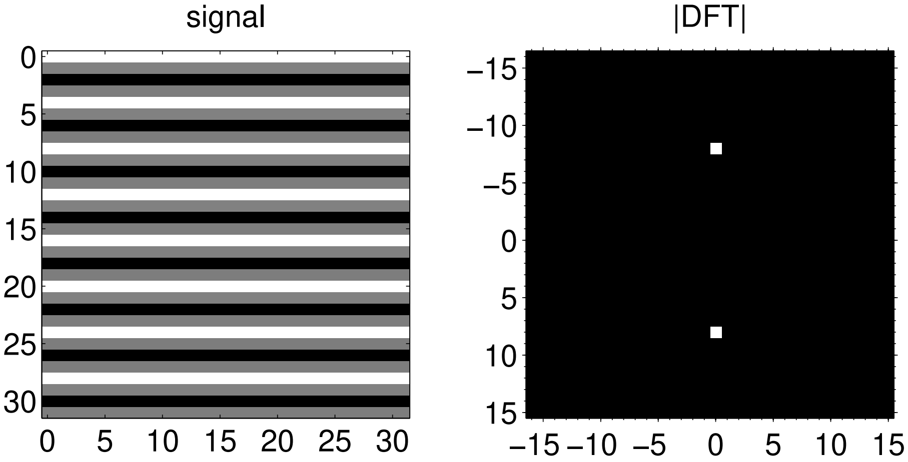
# Examples (iv-b)

as before using `fftshift`



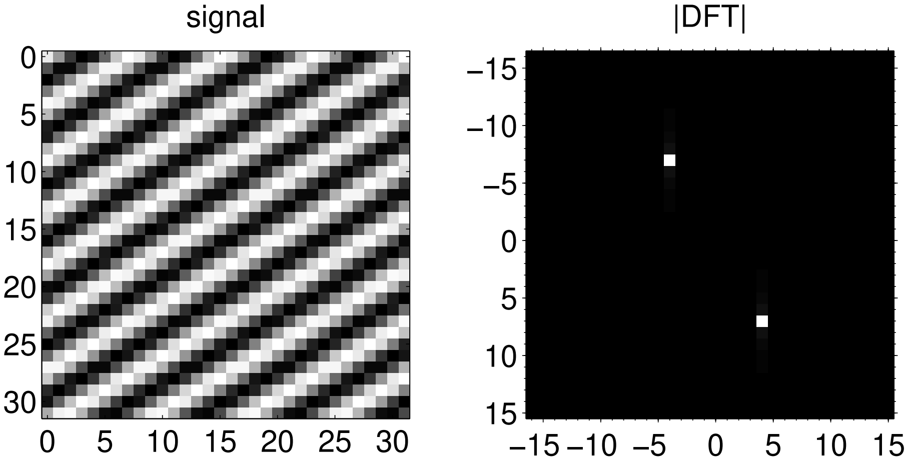
# Examples (v)

$x(n, k) = \sin(2\pi 8n/N)$ with `fftshift`



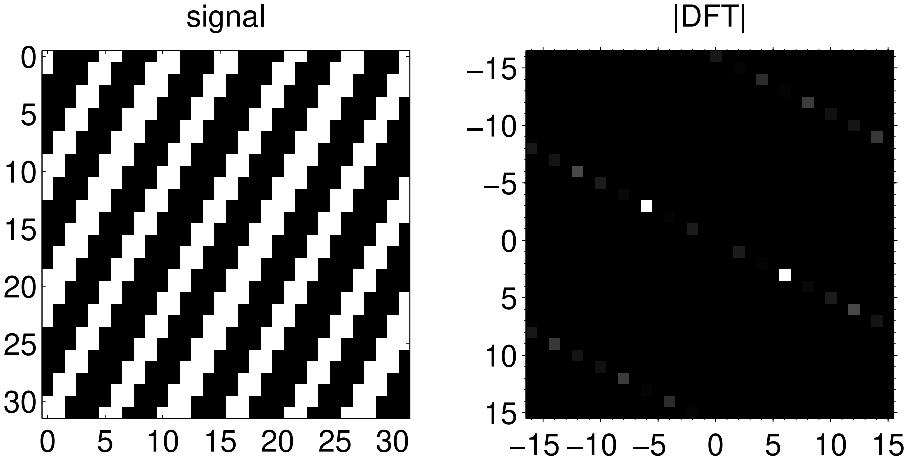
# Examples (v-b)

$x(n,k) = \sin\left(2\pi 8(\cos(\theta)n + \sin(\theta)k)/N\right)$ with `fftshift`



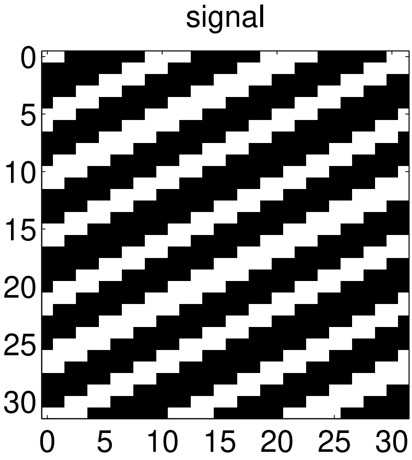
# Examples (vi)

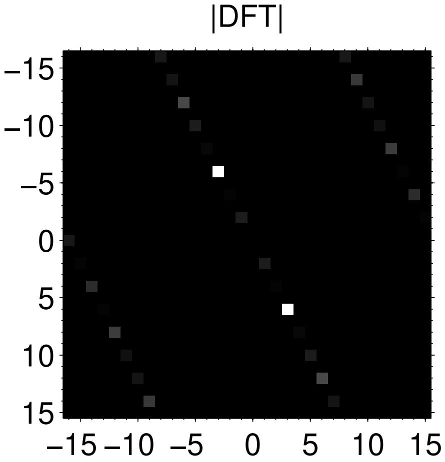
$x(n,k) = I\left\{\sin\left(2\pi(n+2k)/N\right) > 0.2\right\}$ with `fftshift`



# Examples (vii)

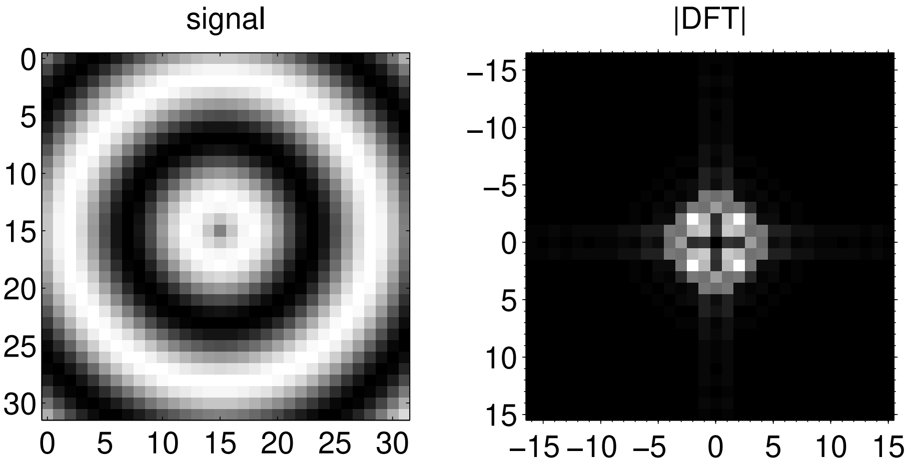
$x(n,k) = I\left\{\sin\left(2\pi(2n+k)/N\right) > 0.2\right\}$ with `fftshift`



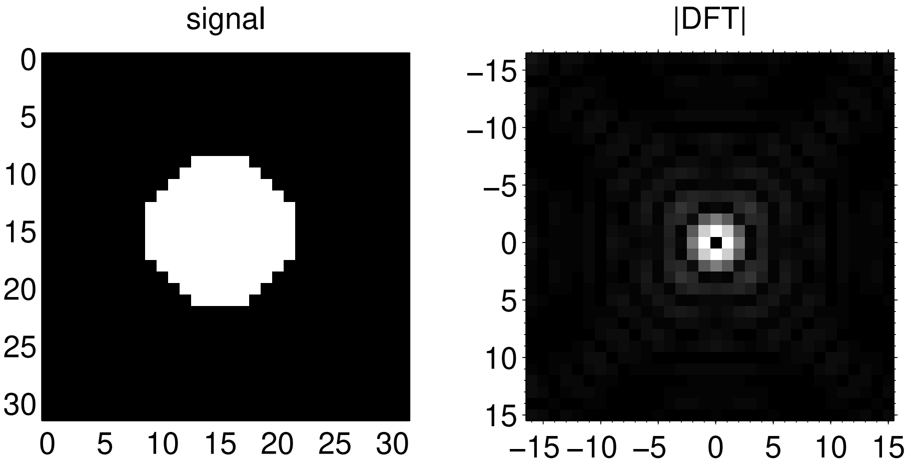
# Examples (viii)

$x(n,k) = \sin\left(2\pi\sqrt{(n/N - 1/2)^2 + (k/N - 1/2)^2}\right)$ with `fftshift`



# Examples (viiib)

$x(n,k) = I\left\{\sqrt{(n/N-1/2)^2+(k/N-1/2)^2} < 0.2\right\}$ with `fftshift`



# Radial symmetry

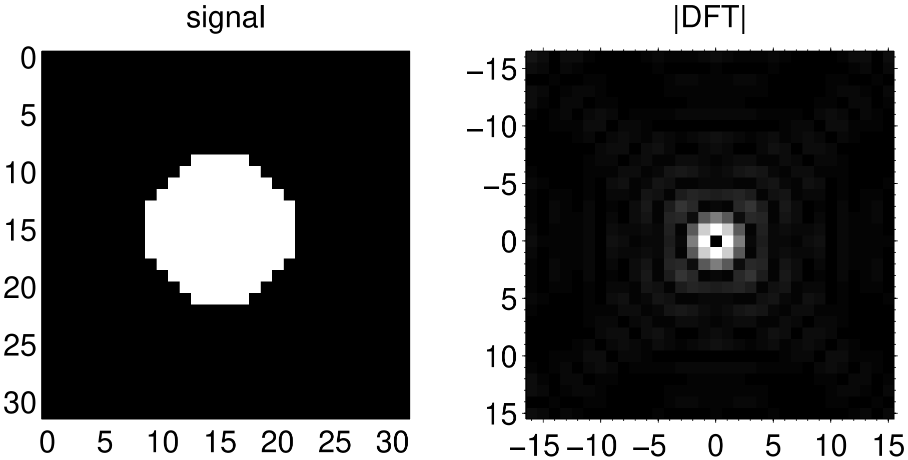
Radially symmetric signal produces radially symmetric DFT

- we know that a rotation in space domain, causes equivalent rotation in frequency domain.
- rotation doesn't change $f(x, y)$, so $F(s, t)$ must also be invariant.
- Remember discretization effects limit radial symmetry.

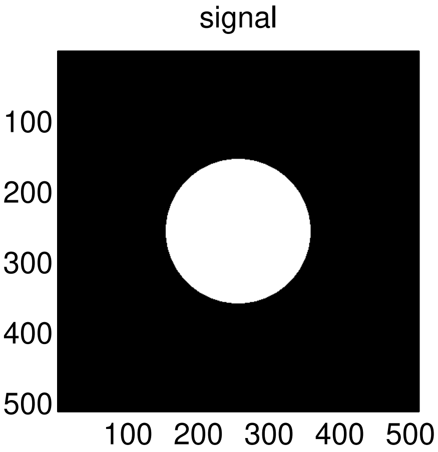
Given radial symmetry can get Hankel transform:

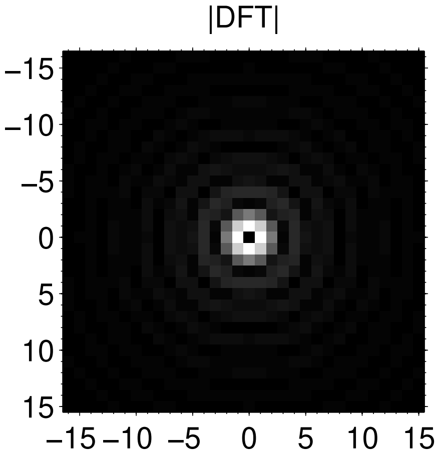
- useful where the system has radial symmetry
- e.g. optical systems, such as lenses.

# Jaggies

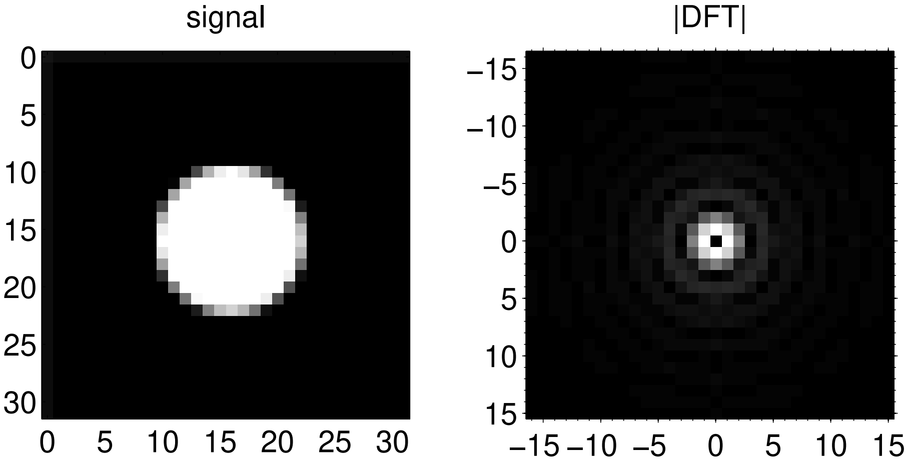

signal
0
5
10
15
20
25
30
0 5 10 15 20 25 30
|DFT|
−15
−10
−5
0
5
10
15
−15 −10 −5 0 5 10 15

# Jaggies (reduced by enhanced resolution)



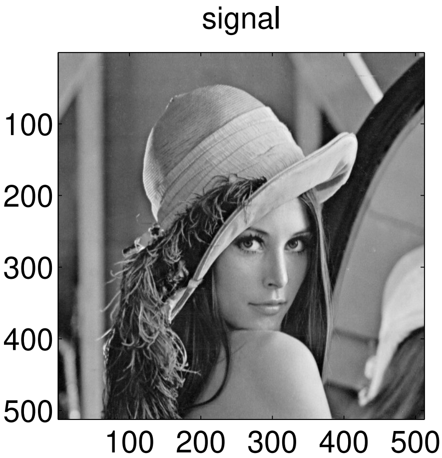



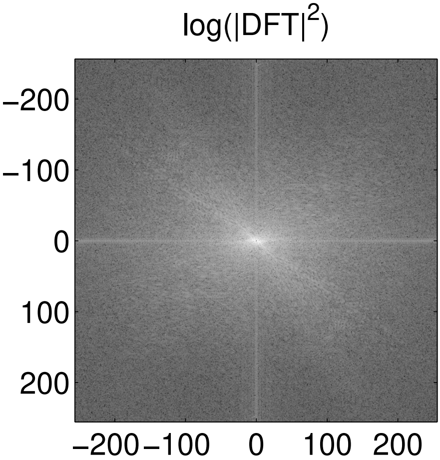
# Jaggies (reduced by pre-filtering)



# Examples (Lena)

Lena image and power-spectra plotted using `fftshift`




http://ndevilla.free.fr/lena/.