

# MAS212 Scientific Computing and Simulation

Dr. Sam Dolan

School of Mathematics and Statistics,  
University of Sheffield

Autumn 2016

<http://sam-dolan.staff.shef.ac.uk/mas212/>

G30 Hicks Building  
s.dolan@sheffield.ac.uk

# Today's lecture

- The **Discrete Fourier Transform** (DFT)
- **Motivation:** looking for hidden signals in experiments
- Definition of DFT and 3 examples
- The inverse DFT
- DFT and **Fourier series**
- **Example:** Gibbs' phenomenon
- **Filtering**

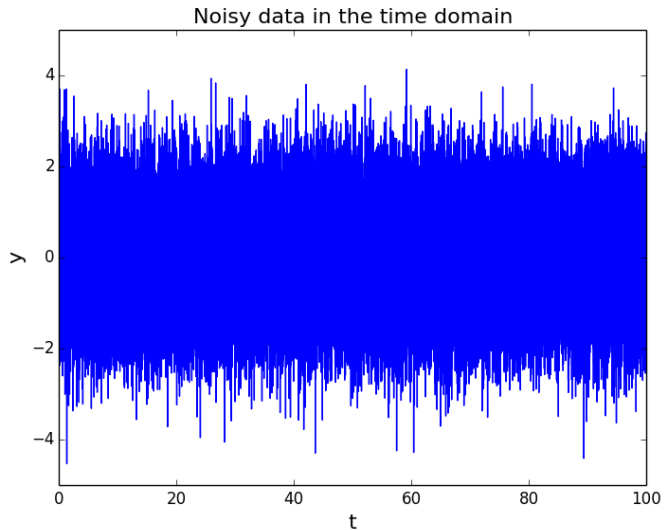
# Motivation

- **Toy Model:** Imagine a scientific instrument (e.g. a seismometer or radio telescope) measuring noisy data with a hidden signal.
- For example,

$$y(t) = 0.1 \sin(3t) + 0.06 \cos(7t) + n(t)$$

- Here blue = signal and red = noise.
- Suppose  $n(t)$  is noise from a Gaussian distribution with standard deviation  $\sigma = 1$ .

# Motivation: Noisy data with hidden signal

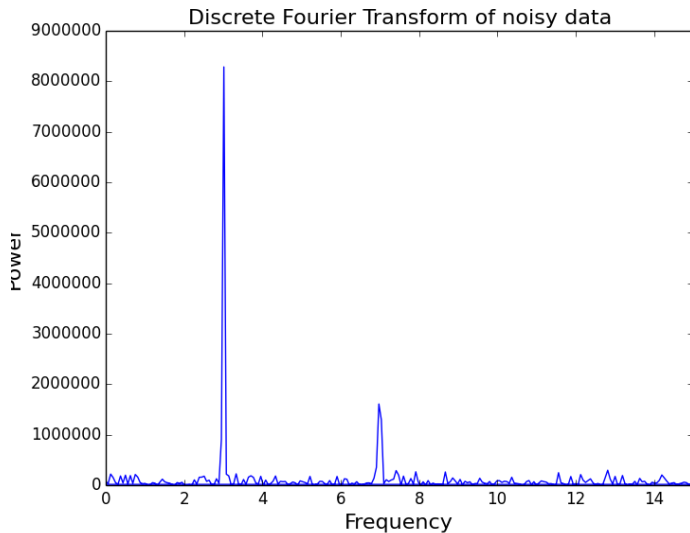


(a) time domain

Motivation: Noisy data with hidden signal

... now find the Discrete Fourier Transform ...

## Motivation: Noisy data with hidden signal



(b) frequency domain.

## Code for this example

Generating the noisy data:

```
n = 2**15
tmax = 100
ts = np.linspace(0, tmax, n)
h = ts[1]-ts[0]
ys = 0.1*np.sin(3*ts) + 0.06*np.cos(7.0*ts) +
      np.random.normal(size=n)
plot(ts, ys, 'b-')
```

Taking the Discrete Fourier Transform:

```
zs = np.fft.fft(ys)
ws = np.fft.fftfreq(ts.size, h)
plt.plot(2.0*np.pi*ws, zs.real**2 + zs.imag**2, 'r-')
plt.xlim(0, 15)
```

# The Discrete Fourier Transform (DFT)

## Definition:

- Suppose we have a data set made up of a sequence of complex numbers:  $x_0, x_1, \dots, x_{n-1}$
- The discrete Fourier transform (DFT) of  $x_j$  is the sequence of **complex numbers**  $\tilde{X}_k$  defined by

$$\tilde{X}_k \equiv \sum_{j=0}^{n-1} x_j \exp(-i 2\pi jk/n), \quad k \in \mathbb{Z},$$

where  $i$  is the unit imaginary.

- The sequence  $\tilde{X}_k$  is  **$n$ -periodic** (as  $e^{2i\pi} = 1$ ):

$$\tilde{X}_{k \pm n} = \tilde{X}_k$$

- $\tilde{X}_k$  are called the **Fourier coefficients** of  $x_j$ .



# The Discrete Fourier Transform (DFT)

## Examples:

Consider the following sequences  $x_j$ ,  
where  $n = 100$  and  $j = 0, 1, \dots, n - 1$  and  $t_j = j/n$ :

(a)  $x_j = \sin(10\pi t_j) + \frac{1}{2} \cos(20\pi t_j)$

(b)  $x_j = t_j^2$ .

(c)  $x_j = \exp\left(-200\left(t_j - \frac{1}{2}\right)^2\right)$

- 1 Plot these sequences.
- 2 Compute the Fourier coefficients  $\tilde{X}_k$  and plot.

## Solution:

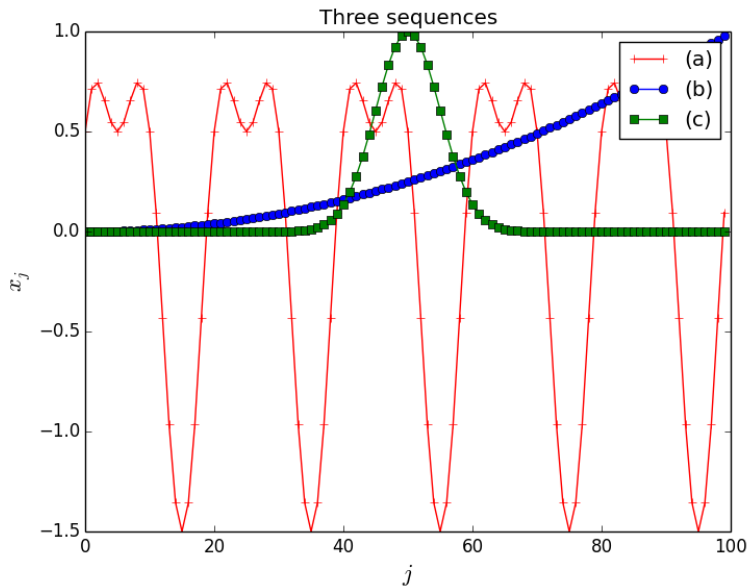
(1) Generate the sequences and plot:

```
import numpy as np
import matplotlib.pyplot as plt
```

```
n = 100
js = np.arange(n)
ts = js / float(n)
x_a = np.sin(10*np.pi*ts) + 0.5*np.cos(20*np.pi*ts)
x_b = ts**2
x_c = np.exp(-200*(ts-0.5)**2)
```

```
plt.plot(js, x_a, 'r+-', label='(a)')
plt.plot(js, x_b, 'bo-', label='(b)')
plt.plot(js, x_c, 'gs-', label='(c)')
```

# Solution (1):



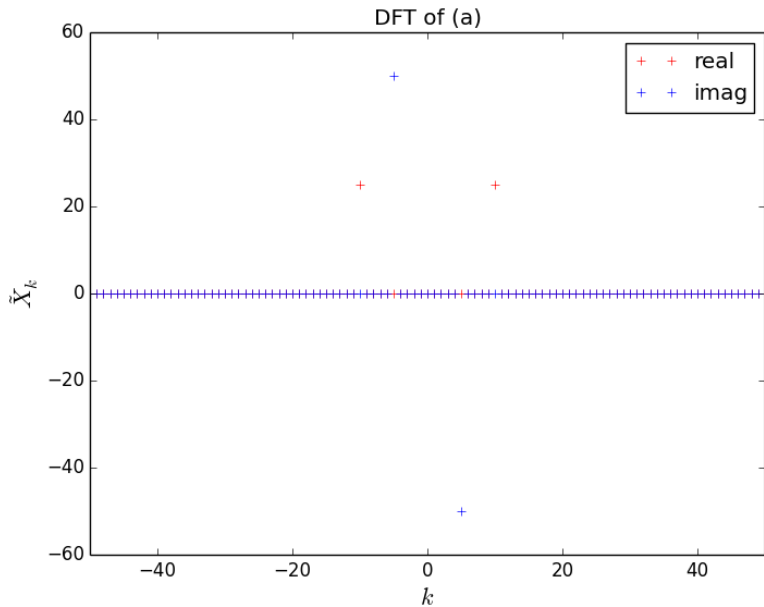
## Solution (2):

- Python has **two** modules for DFTs:  
numpy.fft and scipy.fftpack.
- FFT = **Fast** Fourier Transform (efficient algorithm).
- Remember that  $\tilde{X}_k$  are **complex numbers**.
- We want to plot  $\tilde{X}_k$  **symmetrically** about  $k = 0$ .

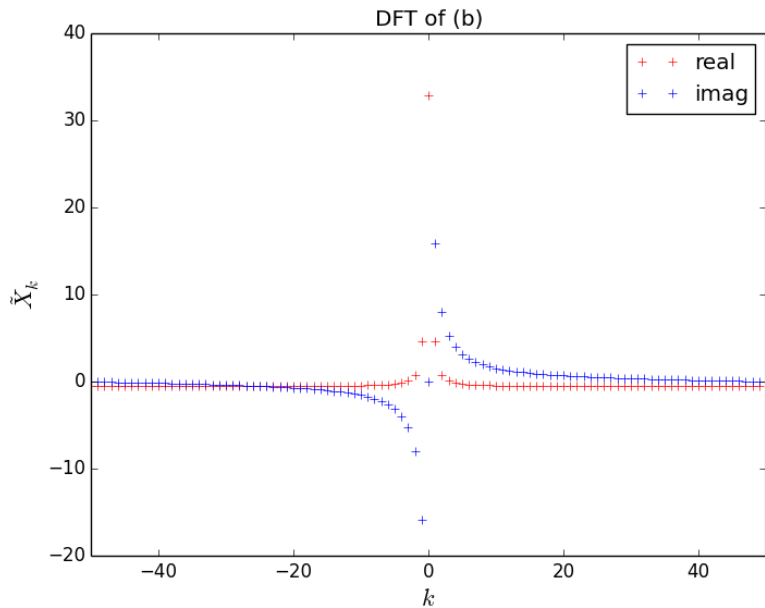
```
Xt_a = np.fft.fft(x_a)      # compute the DFT
ks = np.fft.fftfreq(n, d=1.0/n) # get the array of k values
plt.plot(ks, Xt_a.real, 'r+', label='real')
plt.plot(ks, Xt_a.imag, 'b+', label='imag')
plt.legend()
plt.show()
```

- For real  $x_j$ , the DFT is **Hermitian**:  $X_{-k} = X_k^*$ .

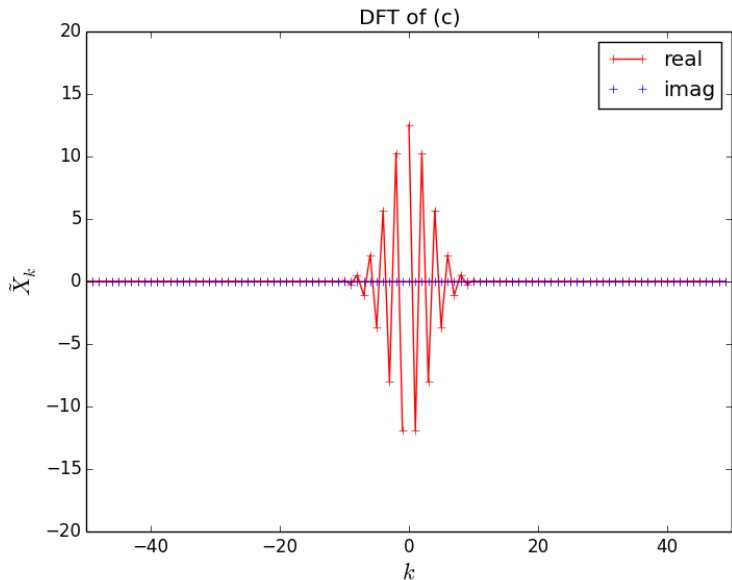
Solution (2): (a)  $x_j = \sin(10\pi t_j) + \frac{1}{2} \cos(20\pi t_j)$



Solution (2): (b)  $x_j = t_j^2$



Solution (2): (c)  $x_j = \exp\left(-200\left(t_j - \frac{1}{2}\right)^2\right)$



# The Inverse Transform

## Inverse Discrete Fourier Transform

- The inverse DFT **goes the other way**:

$$x_j = \frac{1}{n} \sum_{k=0}^{n-1} \tilde{X}_k \exp(+i2\pi jk/n)$$

- Compare with

$$\tilde{X}_k \equiv \sum_{j=0}^{n-1} x_j \exp(-i2\pi jk/n)$$

- For the inverse transformation, use `numpy.fft.ifft()`.



## Proof of inverse formula:

- Substitute definition of  $\tilde{X}_k$  into inverse formula:

$$\begin{aligned} & \frac{1}{n} \sum_{k=0}^{n-1} \{ \tilde{X}_k \} \exp(+i2\pi jk/n) \\ &= \frac{1}{n} \sum_{k=0}^{n-1} \left\{ \sum_{j'=0}^{n-1} x_{j'} \exp(-i2\pi j'k/n) \right\} \exp(+i2\pi jk/n) \\ &= \sum_{j'=0}^{n-1} x_{j'} \times \frac{1}{n} \sum_{k=0}^{n-1} \exp(+i2\pi(j-j')k/n) \\ &= \sum_{j'=0}^{n-1} x_{j'} \delta_{jj'} \\ &= x_j \quad \text{as required.} \end{aligned}$$

- N.B.  $\delta_{jj'}$  is 1 if  $j' = j$ , and 0 otherwise.

## Summary:

$$x_j = \frac{1}{n} \sum_{k=0}^{n-1} \tilde{X}_k e^{+i2\pi jk/n} \quad \Leftrightarrow \quad \tilde{X}_k \equiv \sum_{j=0}^{n-1} x_j e^{-i2\pi jk/n}$$

## Interpreting the DFT

- Suppose  $\{x_j\}$  is a data set of length  $n$ , sampled at regular time intervals  $\Delta t$ :

$$t_j \equiv j\Delta t.$$

- We can think of the DFT as a map from the ‘time domain’ to the ‘frequency domain’:

$$(t_j, x_j) \longrightarrow (\omega_k, \tilde{X}_k)$$

where

$$\omega_k = k\Delta\omega, \quad \Delta\omega = \frac{2\pi}{n\Delta t},$$

such that  $2\pi jk/n = \omega_k t_j$ .

- The frequency interval  $\Delta\omega$  is inversely proportional to the total duration  $n\Delta t$ .
- The maximum frequency ( $\omega_{\max} = 2\pi/\Delta t$ ) is inversely proportional to the sampling interval  $\Delta t$ .

## Interpreting the DFT

Suppose  $x_j$  are **real** measurements at times  $t_j = j\Delta t$ .

- $x_j$  are real  $\Leftrightarrow \tilde{X}_k$  are **Hermitian**:  $\tilde{X}_k^* = \tilde{X}_{-k}$
- $\Rightarrow \tilde{X}_0$  is **real**:  $\tilde{X}_0 = \sum x_j = n\bar{x}$
- Let  $\tilde{X}_k = a_k - ib_k$ . Then

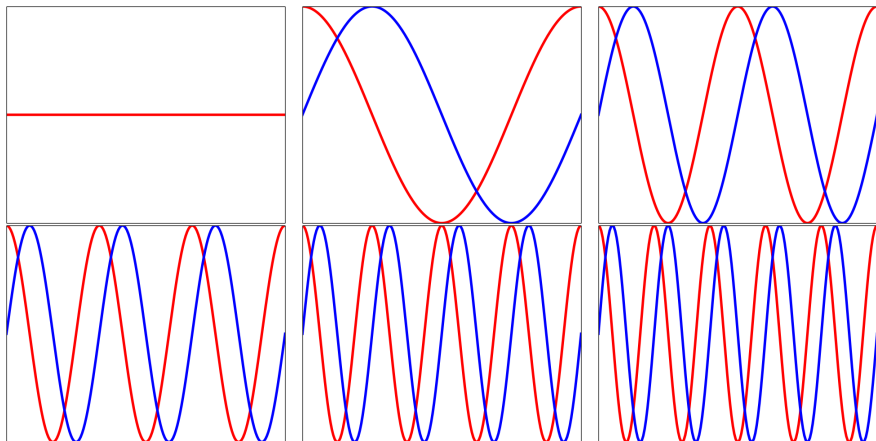
$$x_j = \frac{1}{2}a_0 + \frac{2}{n} \sum_{k=1}^{n-1} \{a_k \cos(\omega_k t_j) + b_k \sin(\omega_k t_j)\}$$

where  $a_k$  and  $b_k$  are real coefficients

$$a_k = \operatorname{Re} \sum_{j=0}^{n-1} x_j e^{-i2\pi jk/n} = \sum_{j=0}^{n-1} x_j \cos(\omega_k t_j)$$
$$b_k = -\operatorname{Im} \sum_{j=0}^{n-1} x_j e^{-i2\pi jk/n} = \sum_{j=0}^{n-1} x_j \sin(\omega_k t_j)$$

# Interpreting the DFT

- $x_j$  is made up of a sum of harmonics:
- $a_k$  and  $b_k$  are the amplitudes of the harmonics



# Fourier series

- The **Fourier series expansion** of a real function  $x(t)$  in the domain  $t_0 \leq t < t_0 + T$  is:

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty} \{a_k \cos(\omega_k t) + b_k \sin(\omega_k t)\}.$$

where  $\omega_k = \frac{2\pi k}{T}$ .

- The Fourier coefficients  $a_k$  and  $b_k$  are:

$$a_k = \frac{2}{T} \int_{t_0}^{t_0+T} x(t) \cos(\omega_k t) dx$$

$$b_k = \frac{2}{T} \int_{t_0}^{t_0+T} x(t) \sin(\omega_k t) dx.$$

## Example:

Consider the **'step function'**:

$$x(t) = \begin{cases} 1, & t \geq 0, \\ 0, & t < 0, \end{cases}$$

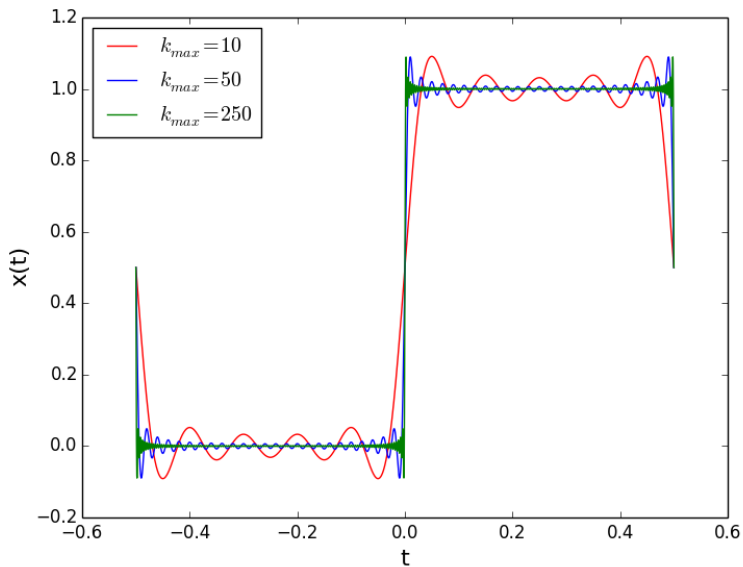
on the domain  $-T/2 \leq t \leq T/2$ .

(a) Show that the Fourier representation of  $x(t)$  is

$$\frac{1}{2} + \sum_{k \text{ odd}}^{\infty} \left( \frac{2}{\pi k} \right) \sin(\omega_k t).$$

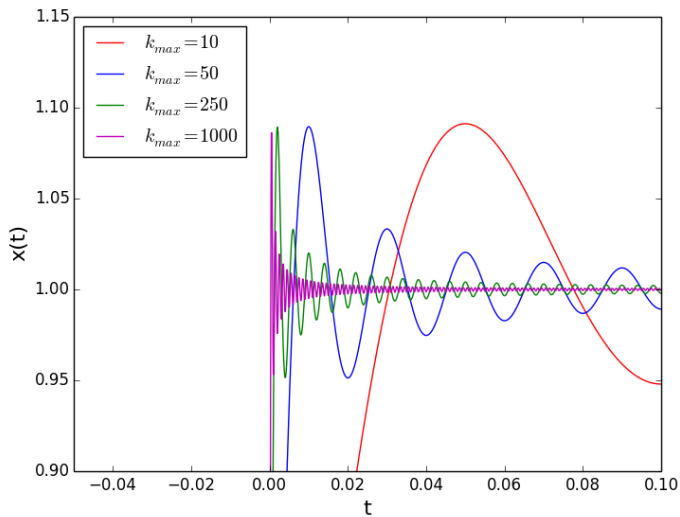
(b) Set  $T = 1$  and compute the Fourier sum up to (i)  $k = 10$ , (ii)  $k = 50$ , (iii)  $k = 250$ . What do you notice?

# Solution:





## Solution:



- The overshoot of  $\sim 9\%$  **does not disappear** as  $k_{max} \rightarrow 0$ .

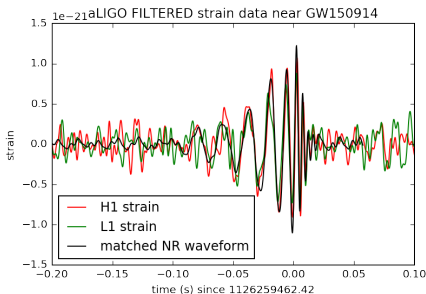
# Gibbs' Phenomenon

If  $x(t)$  is a function with discontinuities:

- Its Fourier series will overshoot at each discontinuity
- The overshoot will **not** disappear as the number of terms in the sum increases.

# Filtering

- Many signal processing algorithms work as follows:
  - Apply a DFT
  - Apply a filter (e.g. remove high or low frequencies)
  - Apply an inverse DFT.
- **Example:** For a tutorial in signal processing for **gravitational-wave detectors**, see [https://lsc.ligo.org/s/events/GW150914/GW150914\\_tutorial.ipynb](https://lsc.ligo.org/s/events/GW150914/GW150914_tutorial.ipynb)



# Filtering

- Suppose I wanted to use a **low-pass filter** to remove the high frequencies from a data set  $a_j$ .
- I can define a **filter** in terms of its Fourier components  $\tilde{F}_k$ ,
- then define a **filtered dataset**  $b_j$  via

$$b_j = IDFT(\tilde{B}_k), \quad \tilde{B}_k = \tilde{A}_k \tilde{F}_k$$

- A naive low-pass filter would be:

$$\tilde{F}_k = \Theta(k, k_0) \equiv \begin{cases} 0 & |k| > k_0 \\ 1 & \text{otherwise} \end{cases}, \quad -\frac{n}{2} \leq k \leq \frac{n}{2}.$$

- But this is not suitable. **Why not?** First we need to understand the **convolution theorem**.

# Convolution

## Definition:

- The convolution of two  $n$ -periodic sequences  $a_j$  and  $b_j$  is defined by

$$(a \otimes b)_j = \sum_{j'=0}^{n-1} a_{j'} b_{j-j'}$$

## Notes:

- Convolution is like **smearing** one sequence with another.

# The Convolution Theorem

- Suppose  $a_j$  and  $b_j$  are sequences with Fourier coefficients  $\tilde{A}_k$  and  $\tilde{B}_k$ . Now consider the sequence  $c_j$  whose Fourier coefficients are  $\tilde{C}_k = \tilde{A}_k \tilde{B}_k$ . What are  $c_j$ ?

$$\begin{aligned}c_j &= \frac{1}{n} \sum_{k=0}^{n-1} (\tilde{A}_k \tilde{B}_k) \exp(i 2\pi j k / n) \\&= \frac{1}{n} \sum_k \sum_p \sum_q a_p e^{-i 2\pi p k / n} b_q e^{-i 2\pi q k / n} e^{i 2\pi j k / n} \\&= \sum_p \sum_q a_p b_q \times \frac{1}{n} \sum_k \exp\left(i \frac{2\pi k}{n} (j - p - q)\right) \\&= \sum_p a_p b_{j-p} = (a \otimes b)_j\end{aligned}$$

- That is,  $c_j$  is the **convolution** of  $a_j$  and  $b_j$ .

# The Convolution Theorem

- This works both ways:

$$\tilde{C}_k = \tilde{A}_k \tilde{B}_k \quad \Leftrightarrow \quad c_j = (a \otimes b)_j$$

and

$$\tilde{C}_k = (\tilde{A} \otimes \tilde{B})_k \quad \Leftrightarrow \quad c_j = a_j b_j.$$

## Practical application

- Naive low-pass filter:

$$\tilde{F}_k = \Theta(k, k_0) \equiv \begin{cases} 0 & |k| > k_0 \\ 1 & \text{otherwise} \end{cases} \quad \text{for} \quad -\frac{n}{2} \leq k \leq \frac{n}{2}.$$

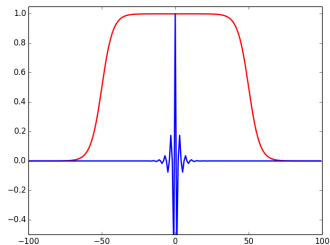
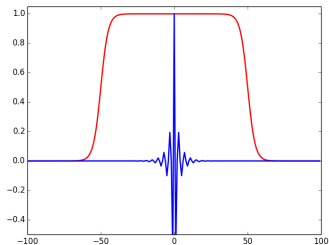
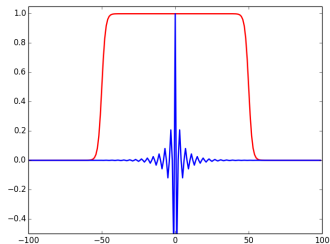
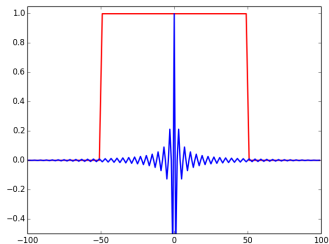
- Defining  $c = \pi j/n$ ,

$$\begin{aligned} f_j &= \frac{1}{n} \sum_{k=-n/2}^{n/2-1} \Theta(k, k_0) e^{2ick} \\ &= \frac{1}{n} \sum_{k=-k_0}^{k_0} (e^{2ic})^k \\ &= \frac{1}{n} \frac{e^{ic(2k_0+1)} - e^{-ic(2k_0+1)}}{e^{ic} - e^{-ic}} \\ &= \frac{1}{n} \frac{\sin((2k_0 + 1)\pi j/n)}{\sin(\pi j/n)} \end{aligned}$$

- This is **wide** and oscillatory. Not suitable.



# Filters and their Inverse DFTs



Red = filter in frequency domain, Blue = time domain profile.