Summary of previous lecture

- Given a sinusoidal carrier signal, I can add information by changing one of the sine signal parameters: Amplitude, frequency or phase
- The spectrum of any signal always has a mirrored negative side (that has no physical meaning)
- Modulating a signal, moves the baseband spectrum around the frequency of the carrier spectrum
  - The bandwidth is at least double, as it also brings up the negative side into the positive part of the x axis.
- Some reasons/applications of modulation at higher frequency:
  - More capacity available at higher frequencies
  - Higher efficiency of antennas / smaller dimensions
  - **Multiplexing**

More on spectrum

Summary of current lecture

- Meaning of spectrum bandwidth
- From discrete to continuous spectrum
- Spectrum analysis in digital calculators
- Introduction to signal digitalisation
- Calculating spectrum in Matlab

Spectrum review

- We mentioned that a periodic continuous signal, can be represented in the frequency domain with a Fourier Series:

$$s(t) = C_0 + \sum_{n=1}^{\infty} C_n \cdot \sin(2\pi nft + \varphi_n)$$

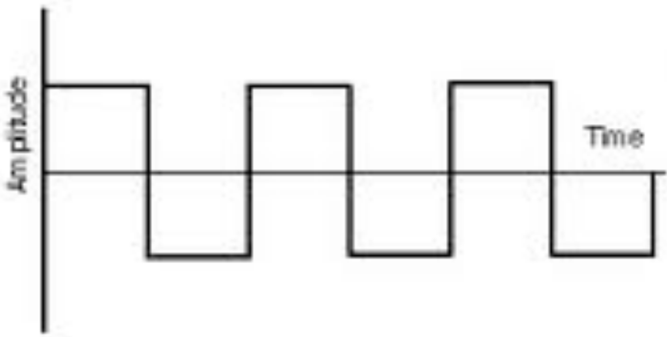
- This can also be written in its complex form:

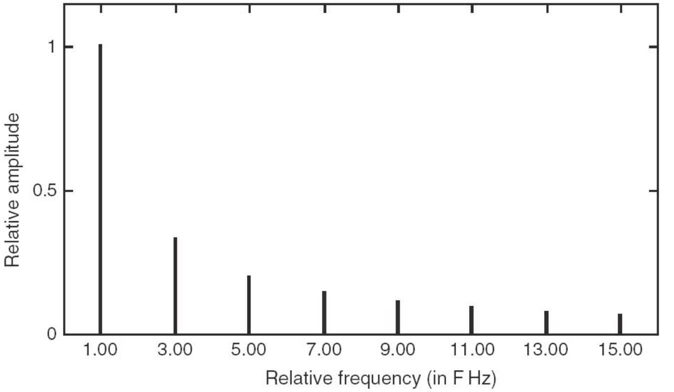
$$s(t) = \sum_{n=-\infty}^{\infty} C_n \cdot e^{-i2\pi \mathrm{nft}}$$

$$e^{i2\pi \mathrm{nft}} = \cos(2\pi \,\mathrm{nf}\, t) + i \sin(2\pi \,\mathrm{nf}\, t)$$

Fourier Series

- With the Fourier Series we can see the spectrum of a periodic signal in the frequency domain as a series of lines

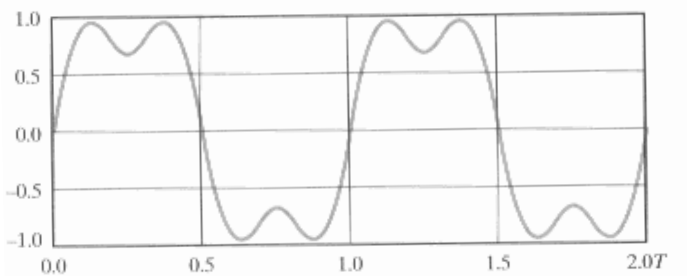




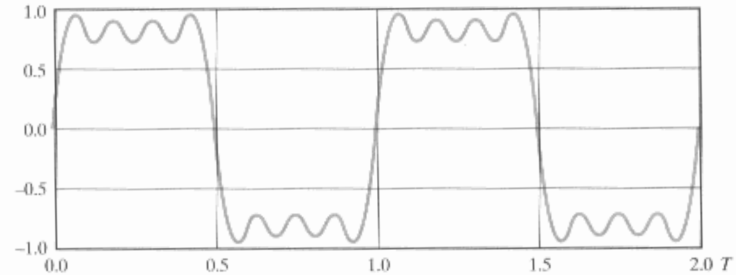
- Notice that the value of the first harmonic (which also indicates the spacing between the spectrum lines) is the inverse to the period of the signal $f = \dfrac{1}{T}$

Meaning of spectrum bandwidth

- The spectrum analysis tells me about the spectral component of a signal.
- The lower parts of the spectrum tell me about the slowly varying parts of the signal, the higher parts tell me about the fast varying part of the signal.
- If you think again in terms of sinusoids, you can see this concept clearly



(c) $\sin(2\pi f_1 t) + 1/3 \sin(2\pi(3f_1)t)$

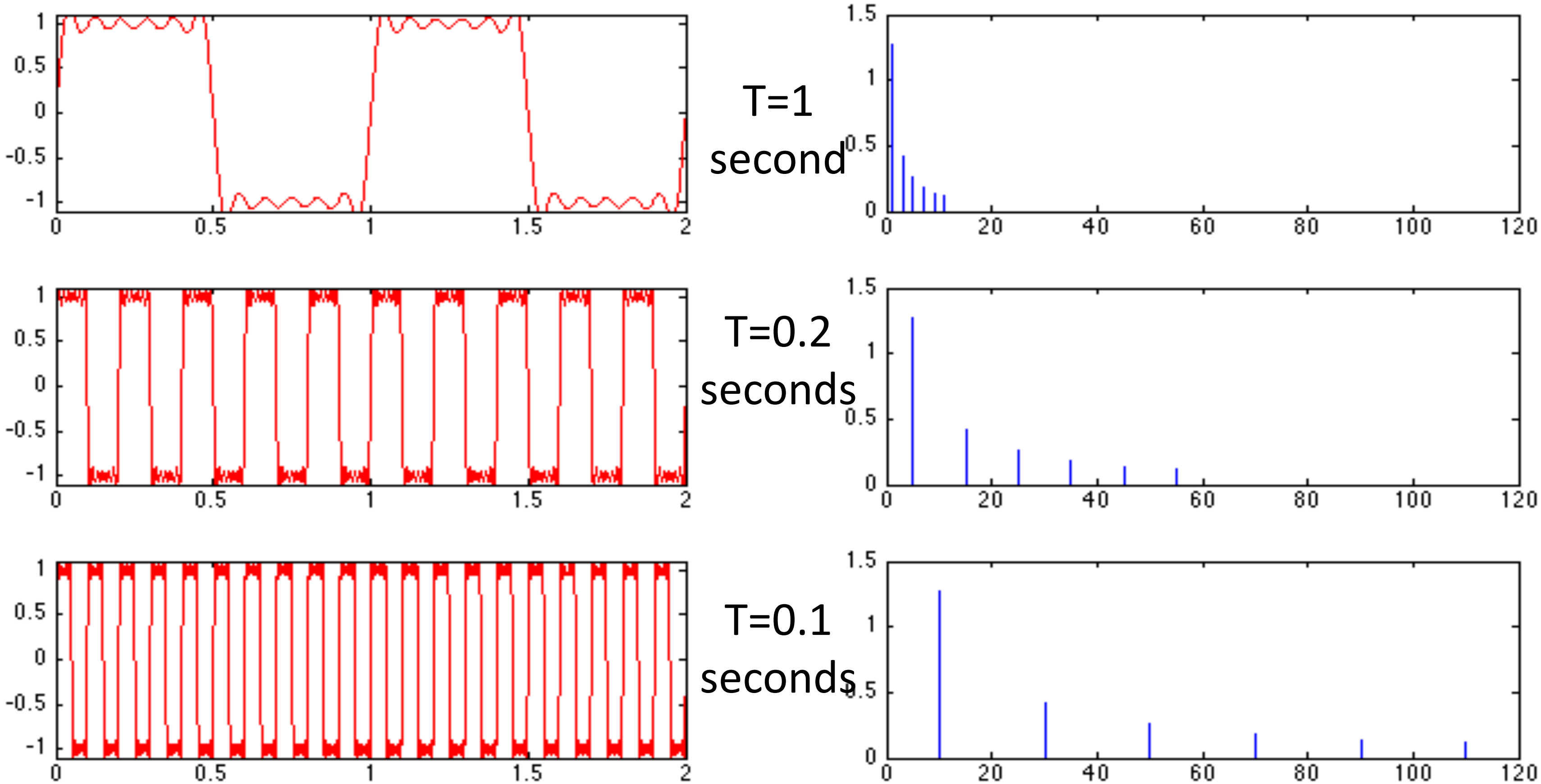


(b) $\sin(2\pi f_1 t) + 1/3 \sin(2\pi(3f_1)t) + 1/5 \sin(2\pi(5f_1)t) + 1/7 \sin(2\pi(7f_1)t)$

- Adding higher spectral components allow the signal to vary more rapidly

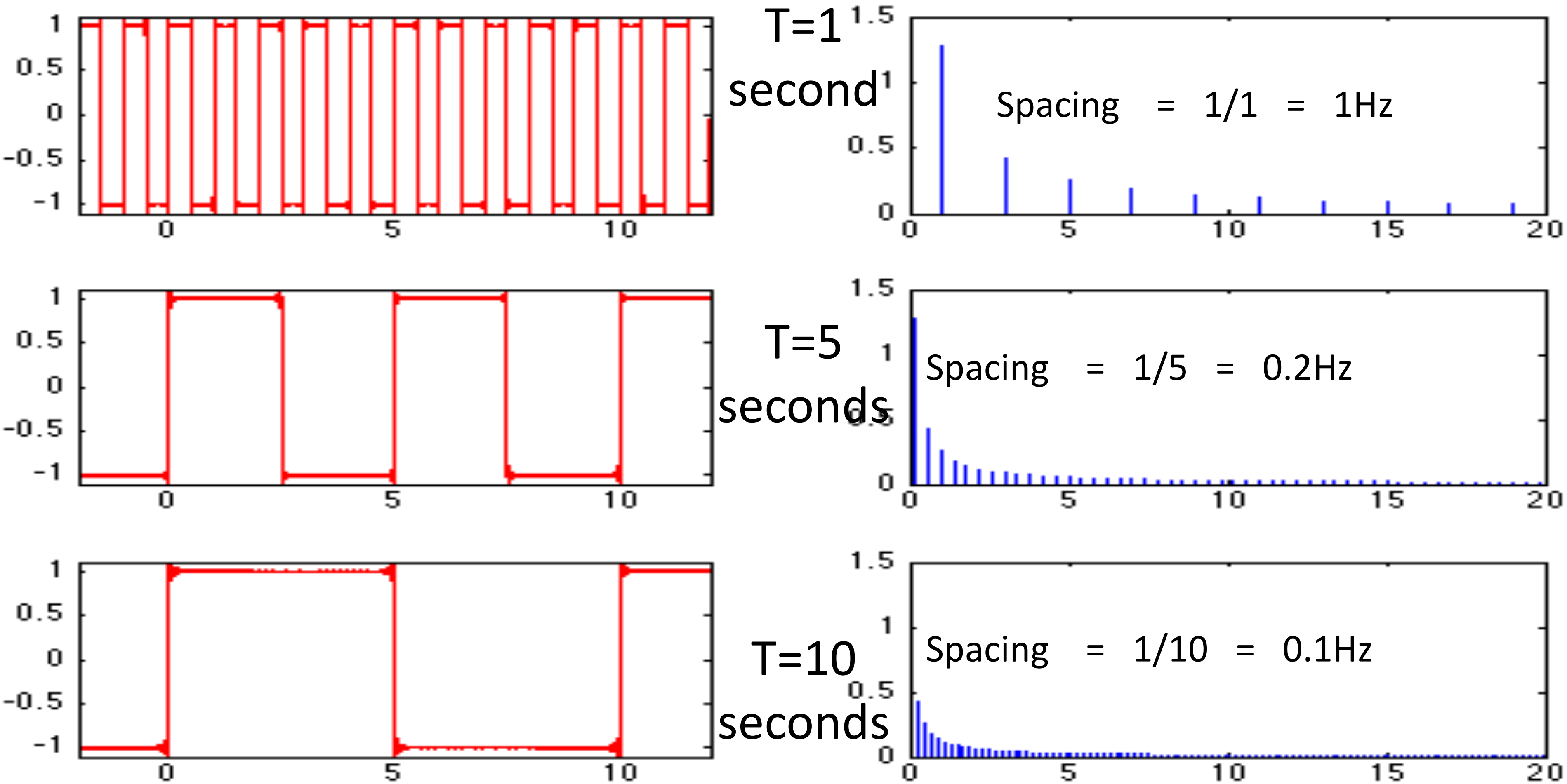
More capacity on higher bandwidth

- A signal that can vary more rapidly can transport more information. Think of a square wave with decreasing period:



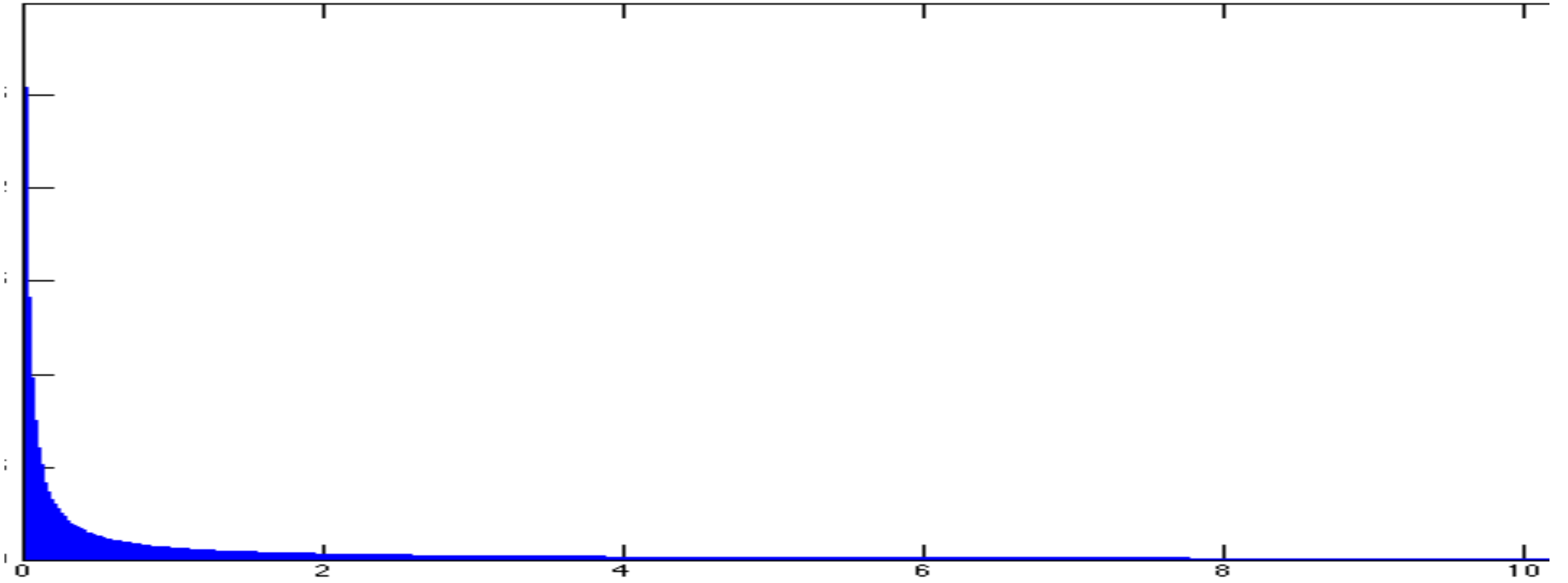
From periodic to non-periodic signal

- If we increase the period of the signal, the lines gets more and more packed together



Increasing the period

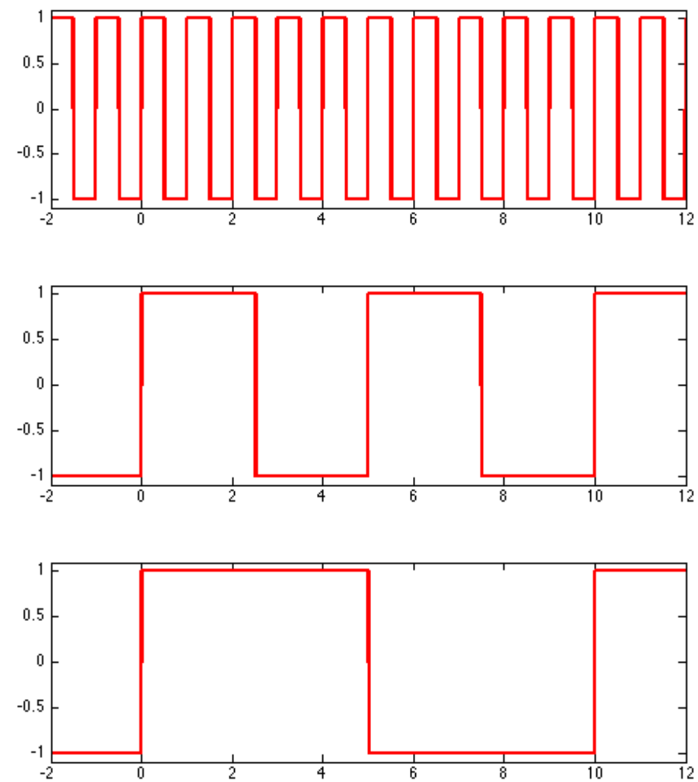
- If we keep increasing the period, the frequency lines will tend towards a continuous spectrum



Non-periodic signal and continuous spectrum

- Having a large period means that the signal repeats after longer and longer times

- A nonperiodic signal can be seen as a periodic signal with extremely large period (infinite).
- So the signal never actually repeats and its spectrum is continuous



Fourier Transform

- A nonperiodic signal can be seen as a periodic signal with a very large period, tending to infinite
- So for analogy of a periodic signal with large period, its spectrum will be continuous.
- The spectrum of a nonperiodic signal is:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-2\pi i \omega t}\, dt$$

Fourier overview

Input signal: time domain	Fourier operation	Output signal: frequency domain
Continuous periodic	Series: $F(\omega) = \sum_{n=-\infty}^{\infty} s_n \cdot e^{-i2\pi\omega n}$	Discrete
Continuous non periodic	Integral: $F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-2\pi i\omega t}\, dt$	Continuous

Fourier overview

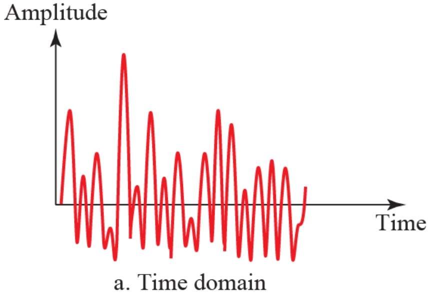
This can only be used for theoretical analysis, because the input signal is continuous

Input signal: time domain	Fourier operation	Output signal: frequency domain
Continuous periodic	Series: $F(\omega) = \sum_{n=-\infty}^{\infty} s_n \cdot e^{-i2\pi\omega n}$	Discrete
Continuous non periodic	Integral: $F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-2\pi i\omega t}\, dt$	Continuous

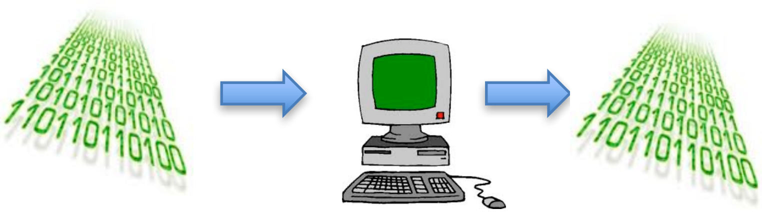
Fourier in the digital era

- This is all mathematical and theoretical, but today we use computers to process signals, not mathematics…
- … and we want to process nonperiodic signals that are continuous in nature

- Computers work on discrete points, which I can store in a memory and process one after the other…
- … but a continuous signal has infinite number of points, so how do we match the two things?



continuous signal

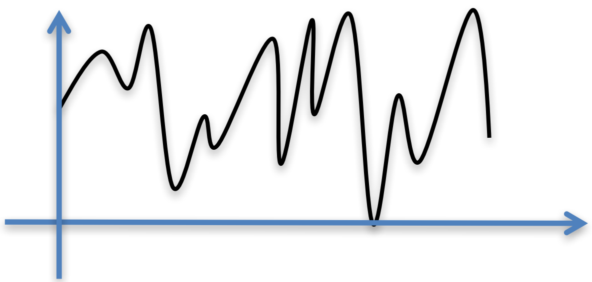


discrete input -- processing -- discrete output

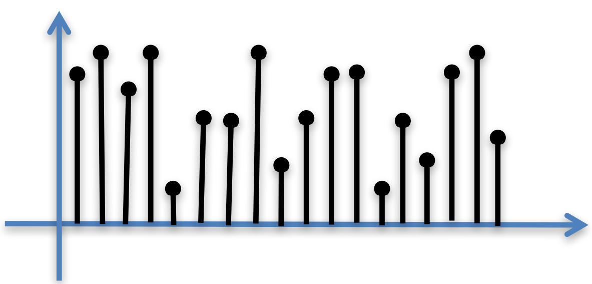
- The solution is that both our signal and the spectrum need to be converted to the discrete world

Converting from analogue to digital

- Converting a signal from analogue (continuous) to digital (discrete), means taking a value every so often (sampling).



Analogue (continuous) signal



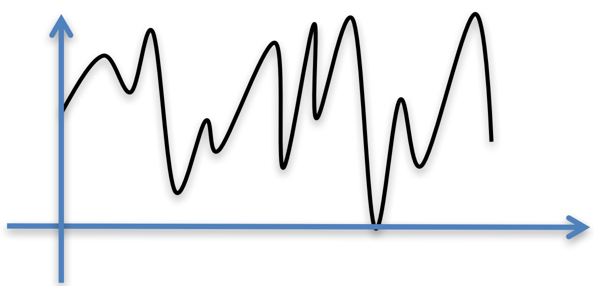
Digital (discrete) signal

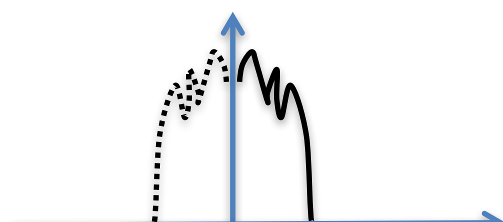
- The big question is: how often should I sample the analogue signal not to loose any information?

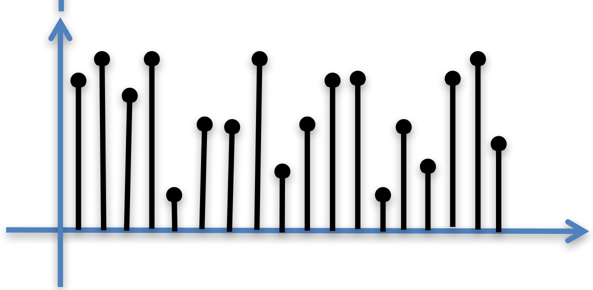
How often should I sample an analogue signal not to loose any information?

Nyquist-Shannon sampling theorem:
if you sample at a rate at least twice the signal bandwidth, then you don’t loose any information!

- Nyquist rate $f_N=2f_{max}$







Nyquist rate f_N=2ooHz
➔ If I take **<u>at least</u>** 200 samples per second, then I don’t loose any information

Digitalization of Fourier

- In order for a computer to be able to do spectrum calculations we need to do 2 actions (**<u>both of them</u>**):
  1. Design a new Fourier transform that can work with a discrete (i.e. digitalized) input signal. This will give us a continuous spectrum
  2. Take the output of action 1) (a continuous spectrum) and transform it into a discrete spectrum

Digitalization of Fourier-step I: DTFT

- The first step is considering that our input signal is a sequence of points.
- This transforms the Fourier integral into the Discrete-time Fourier Transform (DTFT)

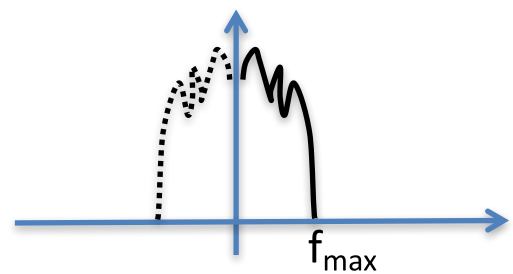
$$F(\omega) = \sum_{n=-\infty}^{\infty} s[n] \cdot e^{-i2\pi\omega n}$$

- Notice that while the signal in the time domain s(n) is now discrete, F(w) (the spectrum) is still continuous.

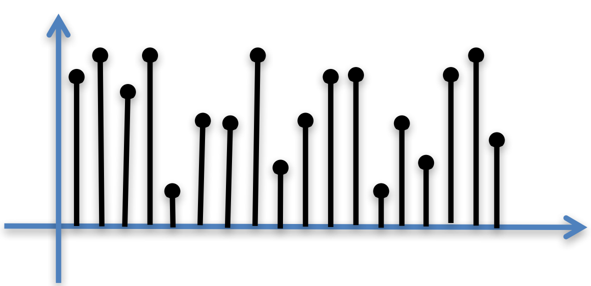
DTFT

- Sampling a signal changes its spectrum substantially!
- If I sample a signal at a frequency f_s, the spectrum becomes periodic, with period f_s

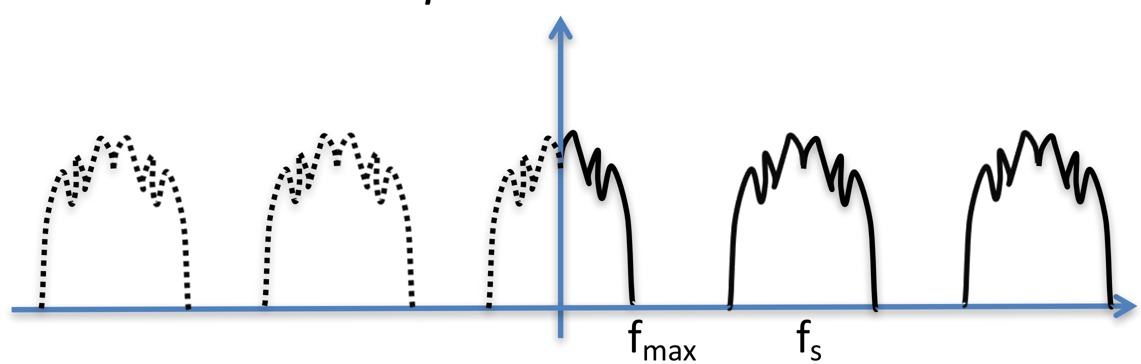
Continuous signal



Spectrum



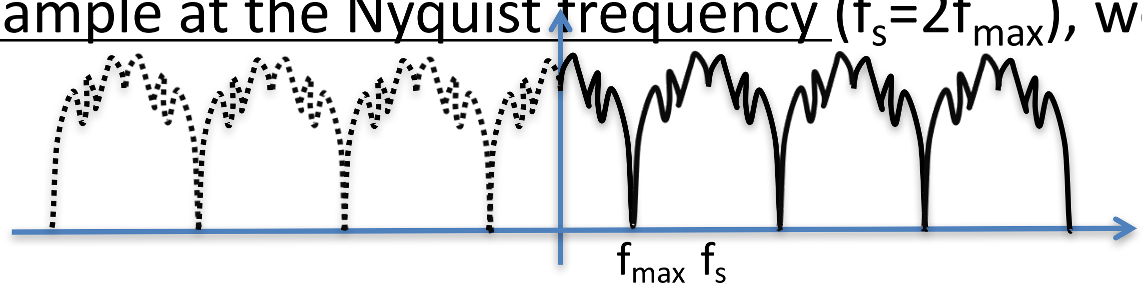
Discrete signal



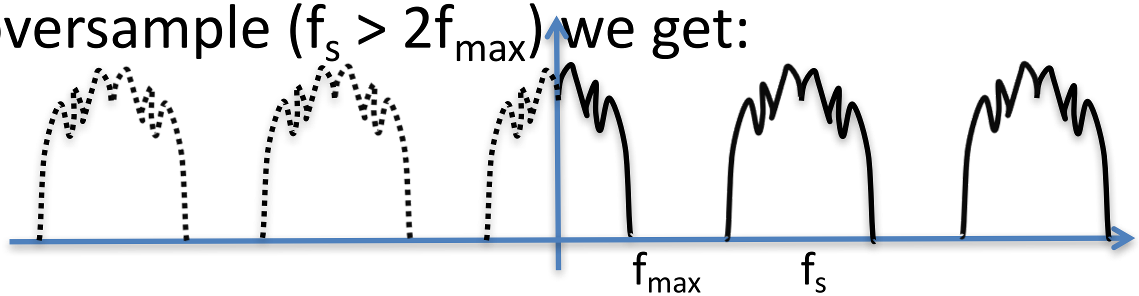
Spectrum

Meaning of Nyquist sampling rate

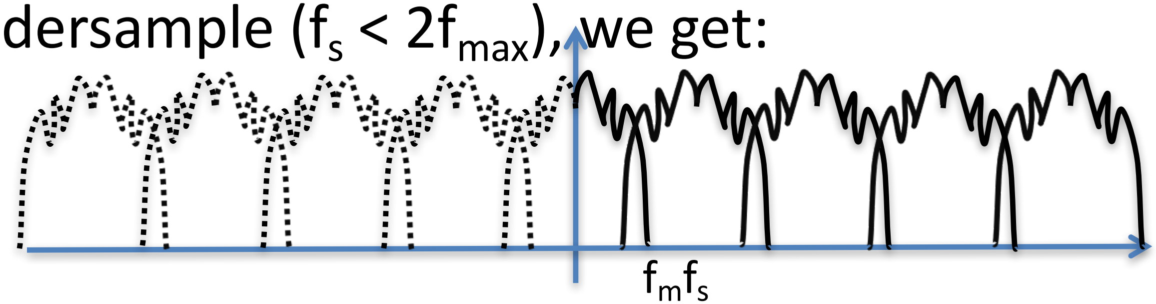
- This result is quite interesting as it shows what happens if you sample at a frequency different than the Nyquist frequency.
- If we sample at the Nyquist frequency ($f_s=2f_{max}$), we get:



- If we oversample ($f_s > 2f_{max}$) we get:

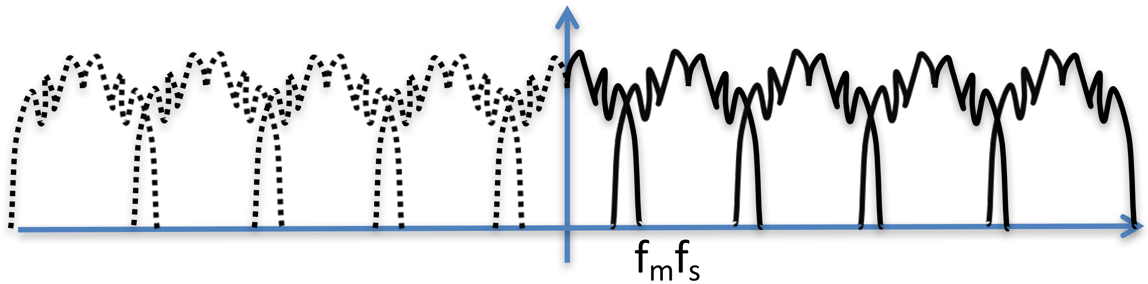


- If we undersample ($f_s < 2f_{max}$), we get:

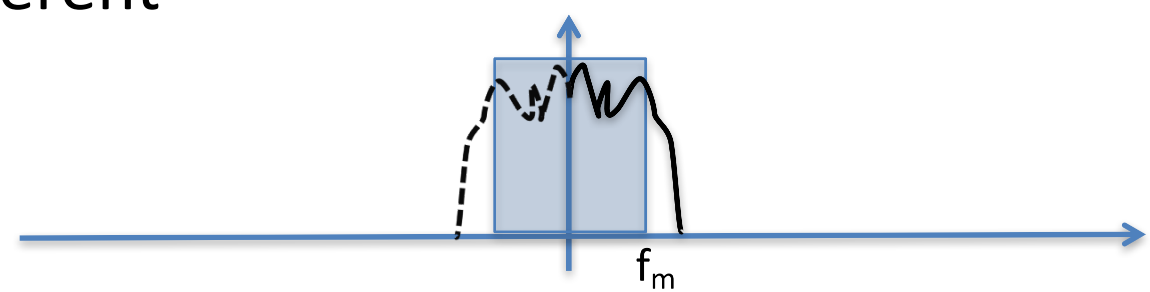


Meaning of undersampling

- From a spectrum perspective, undersampling overlaps the copies of the original spectrum.



- Thus undersampling will distort the signal: if I try to filter out my original spectrum, its shape will be different



Fourier overview

This can only be used for theoretical analysis, because the input signal is continuous

Input signal: time domain	Fourier operation	Output signal: frequency domain
Continuous periodic	Series: $F(\omega) = \sum_{n=-\infty}^{\infty} s_n \cdot e^{-i2\pi\omega n}$	Discrete
Continuous non periodic	Integral: $F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-2\pi i\omega t}\, dt$	Continuous
Discrete	DTFT: $F(\omega) = \sum_{n=-\infty}^{\infty} s[n] \cdot e^{-i2\pi\omega n}$	Continuous

This can be used for computation because the input signal is discrete

However the output signal is continuous, so it cannot be used in a digital machine

Digitalization of Fourier-step II: DFT and FFT

- To represent the DTFT on a computer we need to sample the spectrum we have obtained.
- This transforms the DTFT into the Discrete Fourier Transform (DFT):

$$F_k = \sum_{n=0}^{N-1} s[n] \cdot e^{-i2\pi \frac{k}{N} n}$$

- This F_k are discrete points that can be processed by a PC.
- **In practice the DFT is operated by a much faster algorithm called Fast Fourier Trasnform (FFT)**
  - *The output of FFT is **exactly the same** as DFT, but the computation time is much shorter*

Fourier overview

This can only be used for theoretical analysis, because the input signal is continuous

Input signal: time domain	Fourier operation	Output signal: frequency domain
Continuous periodic	Series: $F(\omega) = \sum_{n=-\infty}^{\infty} s_n \cdot e^{-i2\pi\omega n}$	Discrete
Continuous non periodic	Integral: $F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-2\pi i\omega t}\, dt$	Continuous
Discrete	DTFT: $F(\omega) = \sum_{n=-\infty}^{\infty} s[n] \cdot e^{-i2\pi\omega n}$	Continuous
Discrete	DFT: $F_k = \sum_{n=0}^{N-1} s[n] \cdot e^{-i2\pi \frac{k}{N} n}$	Discrete

This can be used for computation because the input signal is discrete

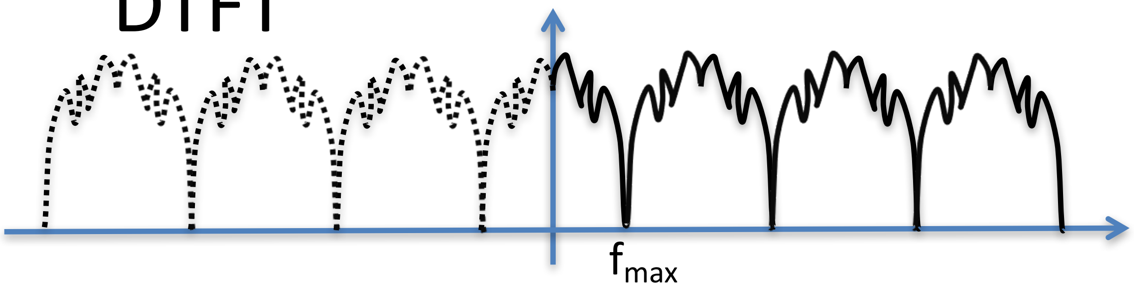
However the output signal is continuous, so it cannot be used in a digital machine

This is discrete both at the input and at the output, so it can be used for computation on a digital machine!

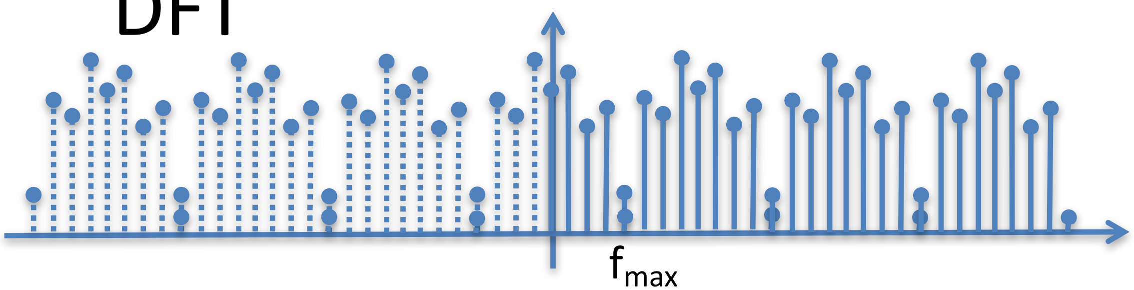
Discrete Fourier Transform (DFT)

- The DFT is a DTFT sampled at certain points.

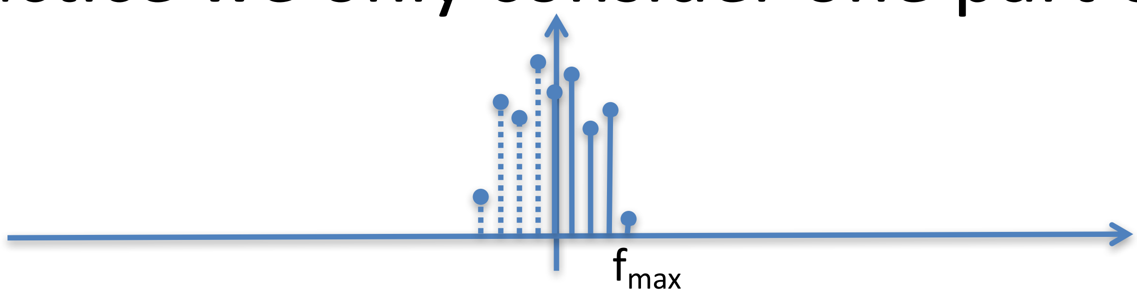
DTFT



DFT

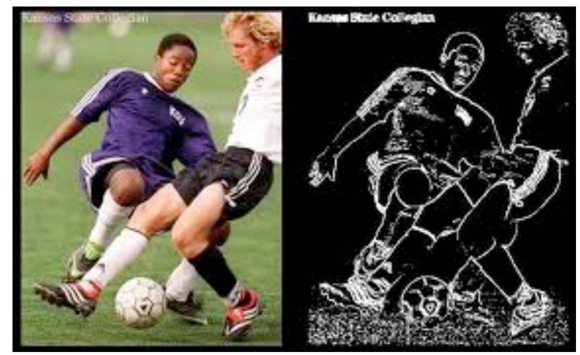


In practice we only consider one part of the DFT

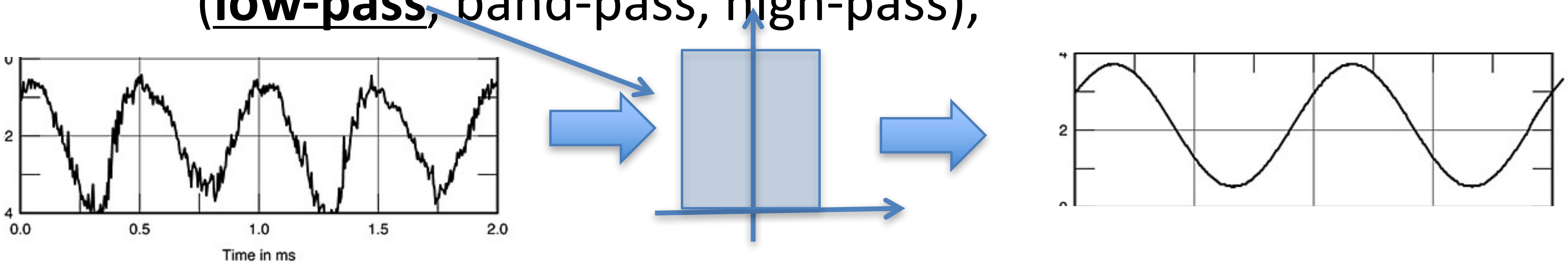


DFT (/FFT) applications

- As we live in a digital world, virtually all practical work that involves processing in the frequency domain is done using the FFT.
- There are numerous applications for this:
  - Imaging: find the edges of a picture, apply digital filters…



  - Signal processing: build a filter (**<u>low-pass</u>**, band-pass, high-pass),



  - … many other…

Learning to use the FFT

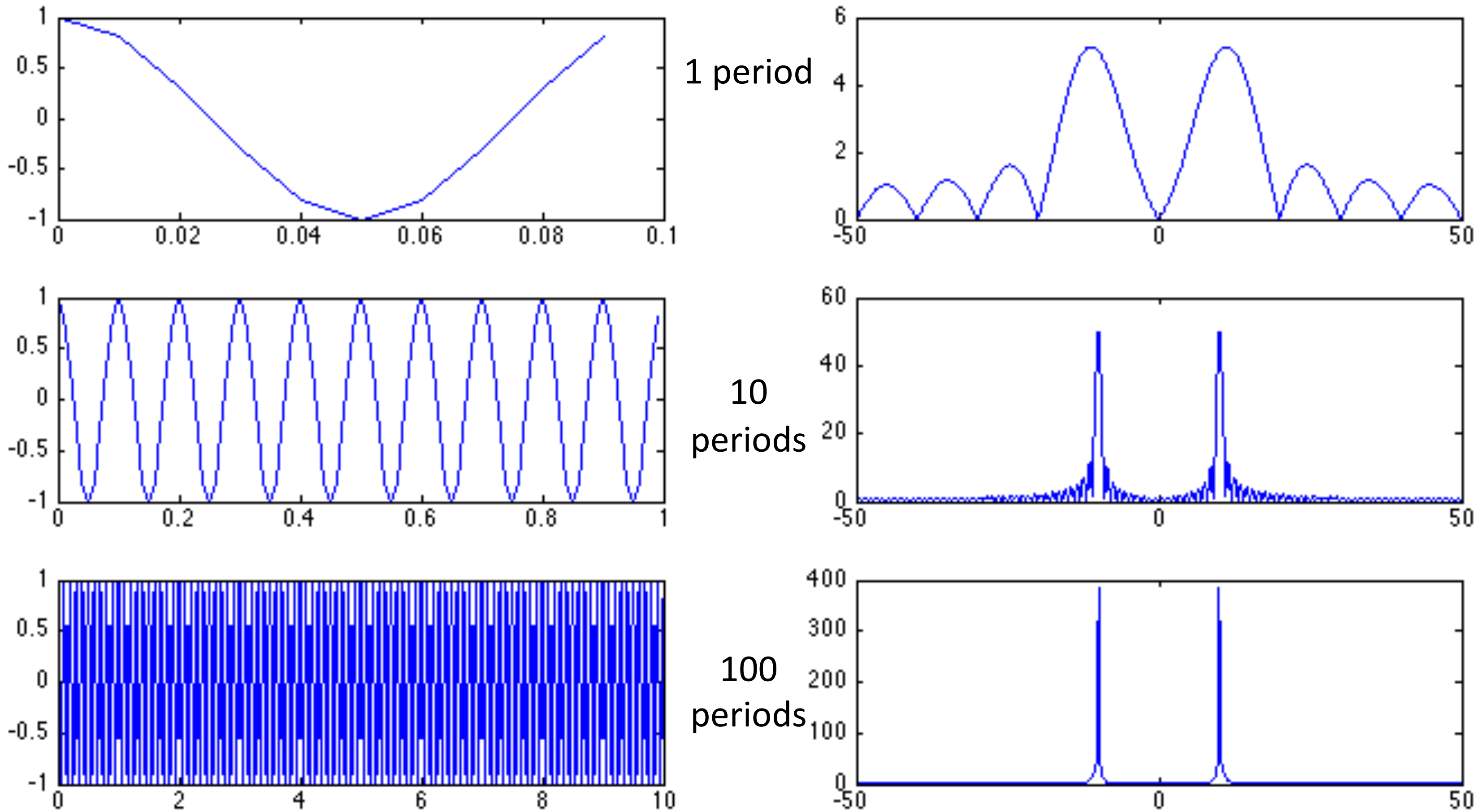
- We now have a look at how to use the FFT to find the spectrum of a signal.
- We will implement this using Matlab in the labs.
- There are three important issues to take care of when using the FFT:
    1. The length of the time-domain signal I consider
    2. Frequency at which I sample my time-domain signal
    3. Number of points of the FFT (related to the frequency at which I sample my spectrum)

Issue 1: length of a periodic signal

- We saw that the theoretical spectrum of a sine wave is one line at the frequency of the sine wave.
- However a sine wave theoretically never stops, while when we do an FFT we need to work with a finite number of elements
- Thus we need to “cut” the signal after a certain number of samples

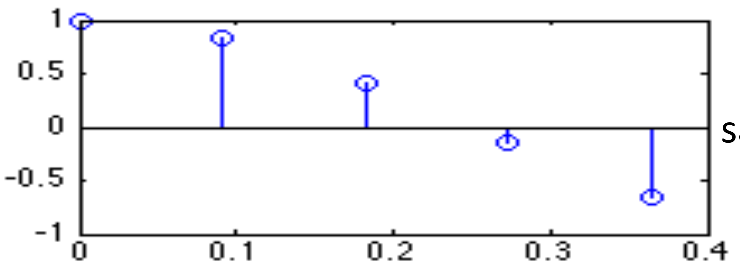
Issue 1: length of a periodic signal

- Cutting a periodic signal has an effect on its spectrum: **<u>the larger the number of periods I consider the more the spectrum will be ideal</u>**

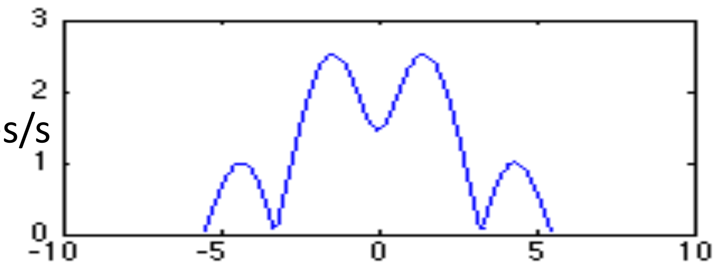


Issue 2: number of samples of my signal

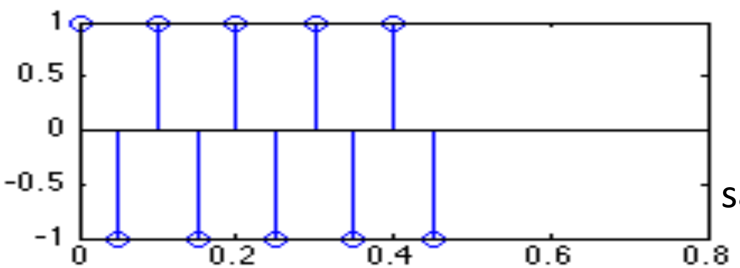
- The number of samples I use for my signal also affect my FFT spectrum.



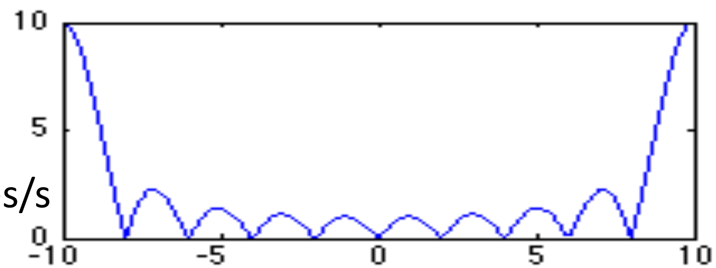
11 samples/s



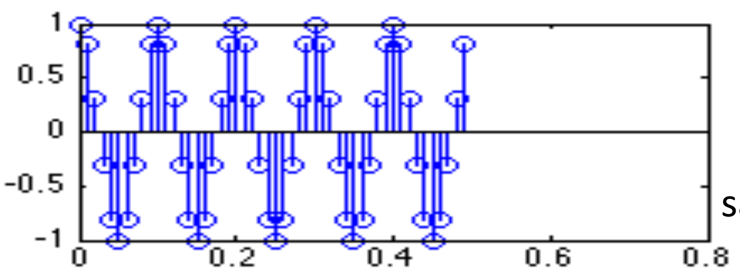
Below Nyquist sampling frequency



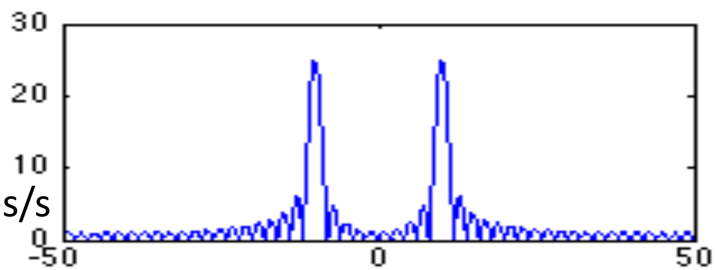
20 samples/s



Exactly at Nyquist sampling frequency



100 samples/s



Above Nyquist sampling frequency

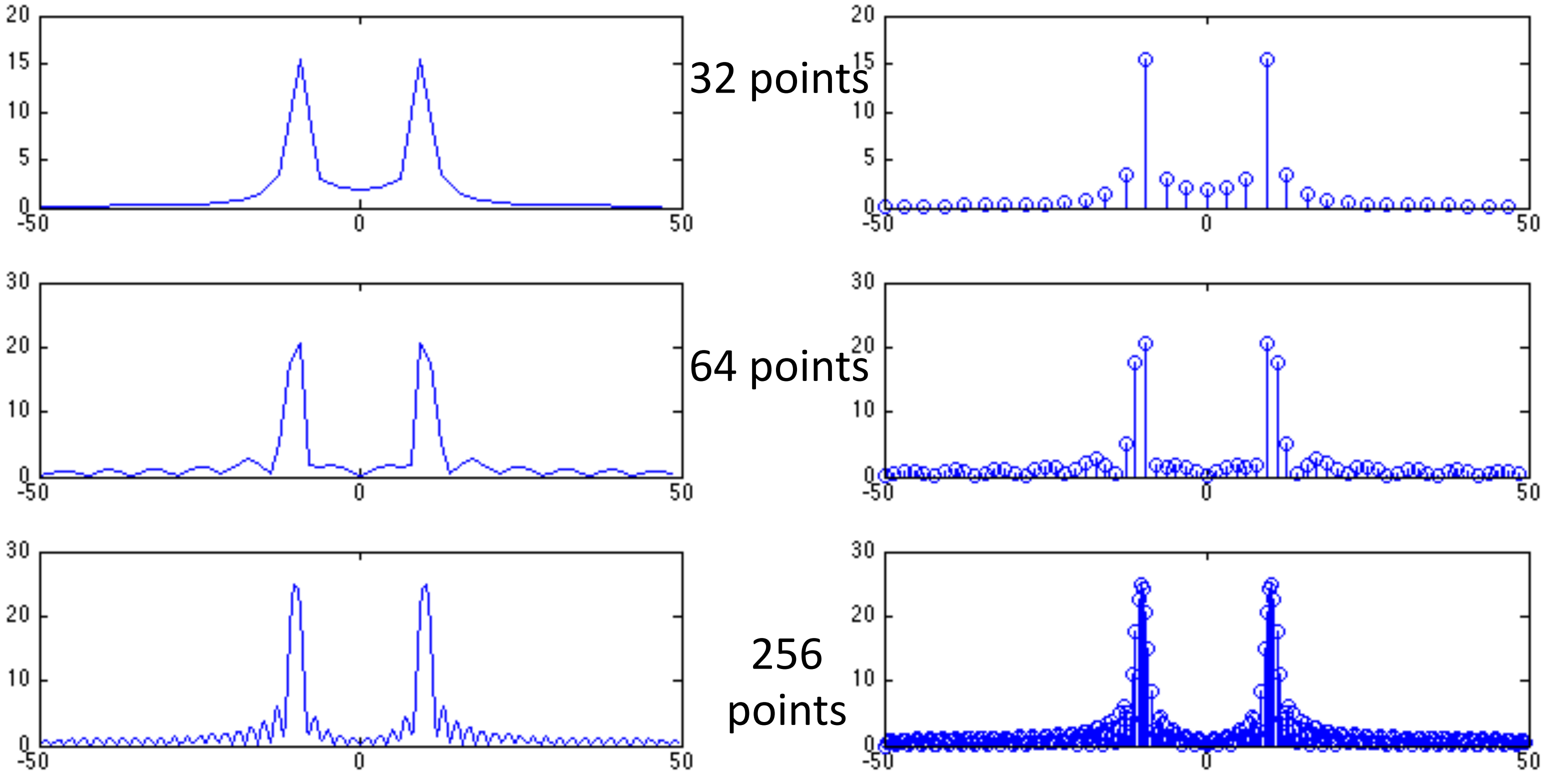
Issue 2: number of samples of my signal

- Thus it is important to select a proper sampling frequency when plotting functions.
- When we write a function in Matlab, the number of samples is given by the number of x values I use to plot my function.
  - For example if I write x=0:0.1:1, and y=sin(2*pi*1*x), I have a frequency of 1Hz for the sine, and I take 10 points in the array x within one period. Thus I have a sampling frequency of 10Hz. This is OK because my sampling frequency is >2*1 Hz (i.e. > Nyquist sampling frequency).
- It is a good idea to insert the sampling frequency in the selection of the array x.
- For example:
  - frequency=20; %frequency of my sine wave
  - sam= 3*frequency; %set sampling frequency at more than twice the sine %frequency (because of approximation errors). Even %sam=2.01*frequency would work
  - p=3; %number of periods I want to calculate
  - x=[0:1/sam:p]; %this gives me the total number of samples I will have
  - y=sin(2*pi*frequency*x); %this is my function

Issue 3: number of points for the FFT

- This indicates the number of points at which I sample my FFT:
  - The higher the number of points the more detailed the result…
  - …but the longer it takes to calculate the FFT.

FFT plot | FFT stem plot



Summary of FFT issues

1. If you have a periodic signal, the larger the number of periods you examine the closer your FFT will be to the ideal spectrum

2. Make sure your sampling rate is larger than the Nyquist sampling rate.

3. You should use a number of points for the FFT not smaller than the number of samples you use
   a. In addition, the FFT algorithm works most efficiently when the number of points is a power of 2, i.e, $N=2^n$: 32, 64, 128, 256, 512, 1024,…

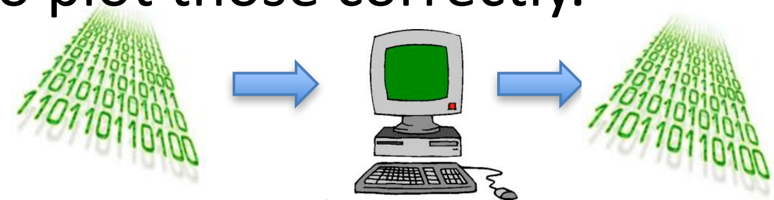
FFT in Matlab

- The FFT function in matlab is part of the signal processing toolbox.
- It is called by the command: fft(y,N):
  - y is the vector containing my time-domain samples
  - N is the <u>number of points </u>over which the fft is calculated.
- The fft gives me back complex numbers (i.e. amplitude and phase)
- For our work we are only interested in the amplitude, so we will apply the function abs()

➔ abs(fft(y,N));

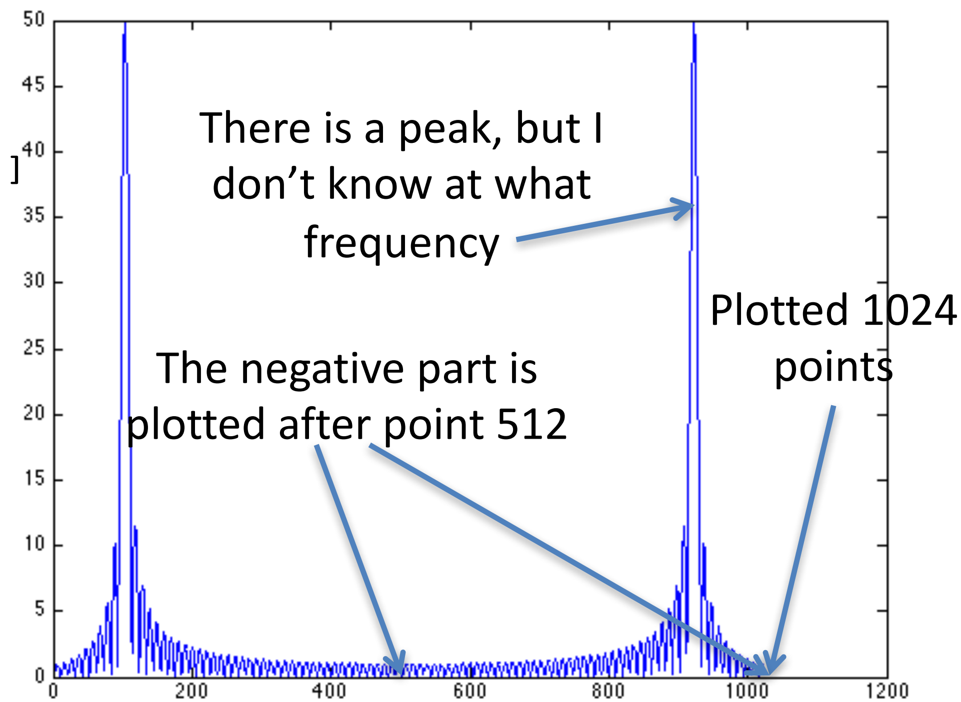
FFT in Matlab

- The fft function makes a calculation on a number of input values, and returns a number of output values, but it's up to the user to plot those correctly.



- For example if I take cos(2π*10) for 1 second, I sample it at 100 Hz and calculate the fft on 1024 points I obtain:

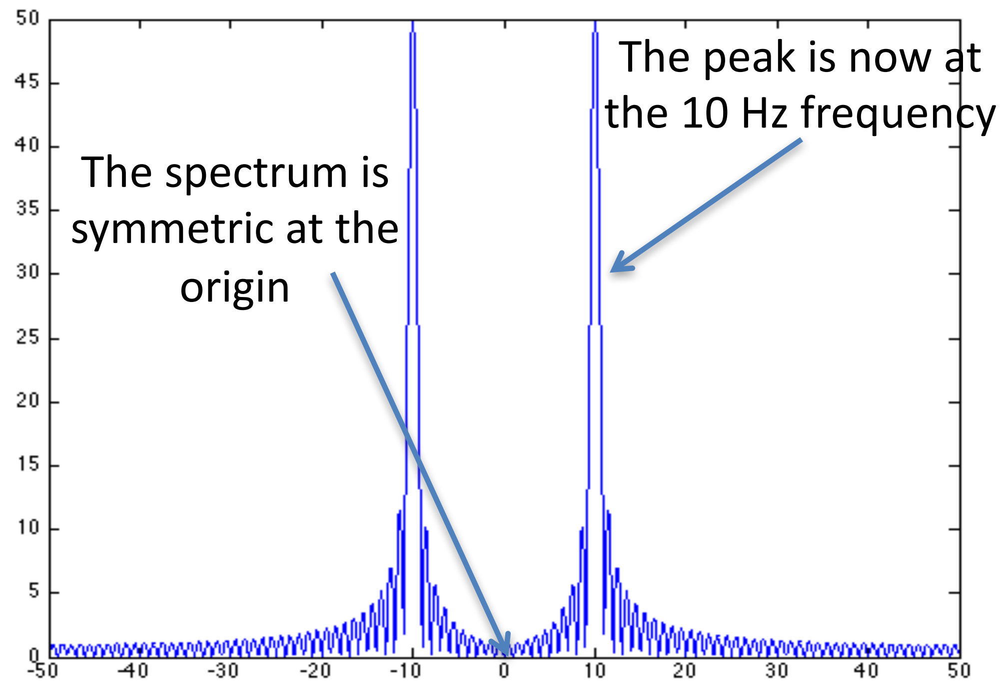
```
fs=100;  %sampling frequency [Hz]
frequency=10; %frequency of sine wave [Hz]
time=1; %how many seconds to consider
x=[0:1/fs:time-1/fs]; %points in x axis
y=cos(2*pi*x*frequency); %your signal

N=1024; %number of FFT samples
F=abs(fft(y,N)); %frequency spectrum of
             your signal

Plot(F); %plot the frequency spectrum
```



Scaling the FFT

- In order to show the classical spectrum that is:
  1. symmetric at the origin
  2. showing the right frequency scale in the X axis
- I need to:
  1. Shift the points, using the command **fftshift()**
  2. Scale the frequencies with respect to f_s and N

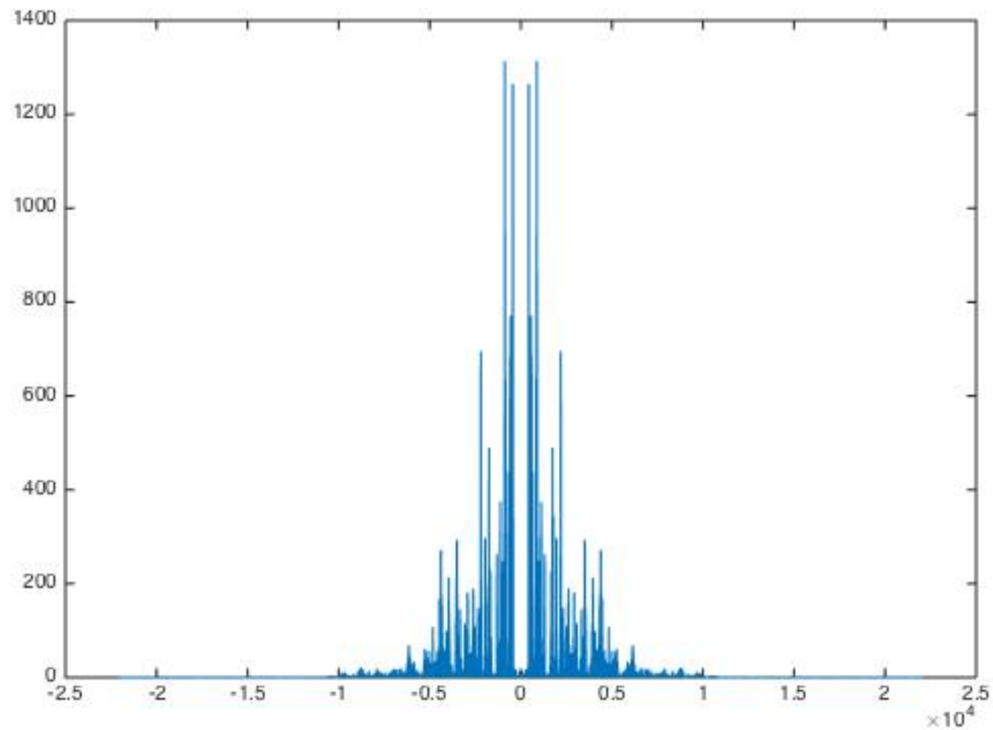
```
fs=100; %sampling frequency[Hz]
frequency=10; %frequency of sine wave [Hz]
time=1; %how many seconds to consider
x=[0:1/fs:time-1/fs]; %points in x axis
y=cos(2*pi*x*frequency); %your signal

N=1024;  %number of FFT samples
F=fftshift(abs(fft(y,N))); %frequency
                         spectrum of your signal

newX=-fs/2:fs/N:fs/2-fs/N; %new x axis for
                                   spectrum plot

Plot(newX,F); %plot the frequency spectrum
```



Example of applying FFT

- This is a short sa

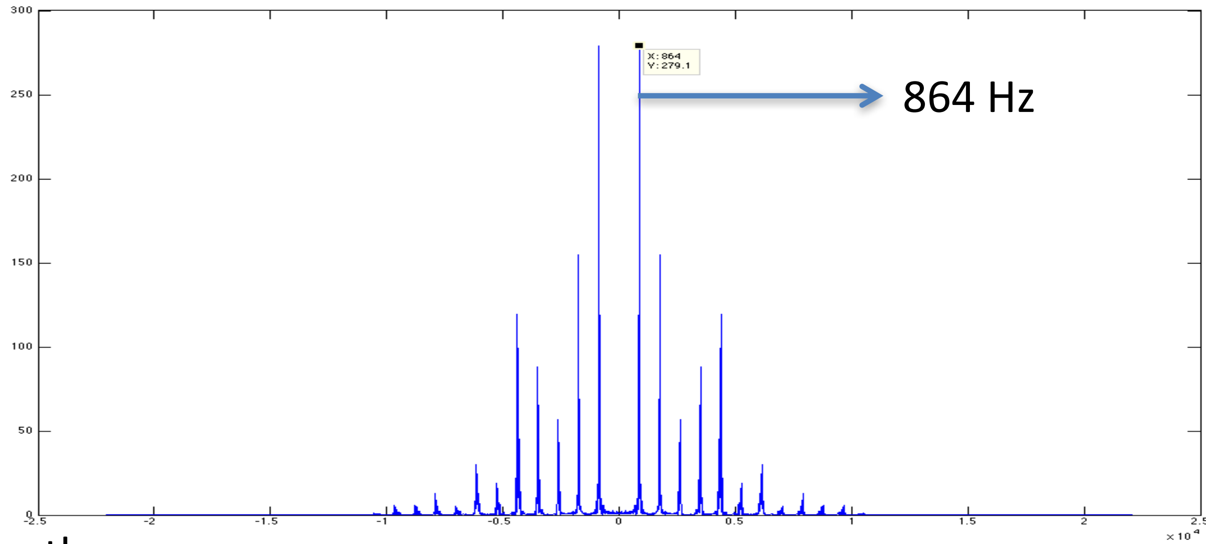
- In Matlab it's an
  considering a sa
  duration is appr



- If I separate individual notes and do a spectral analysis I can find out what note is being played

Example of applying FFT

- First note:

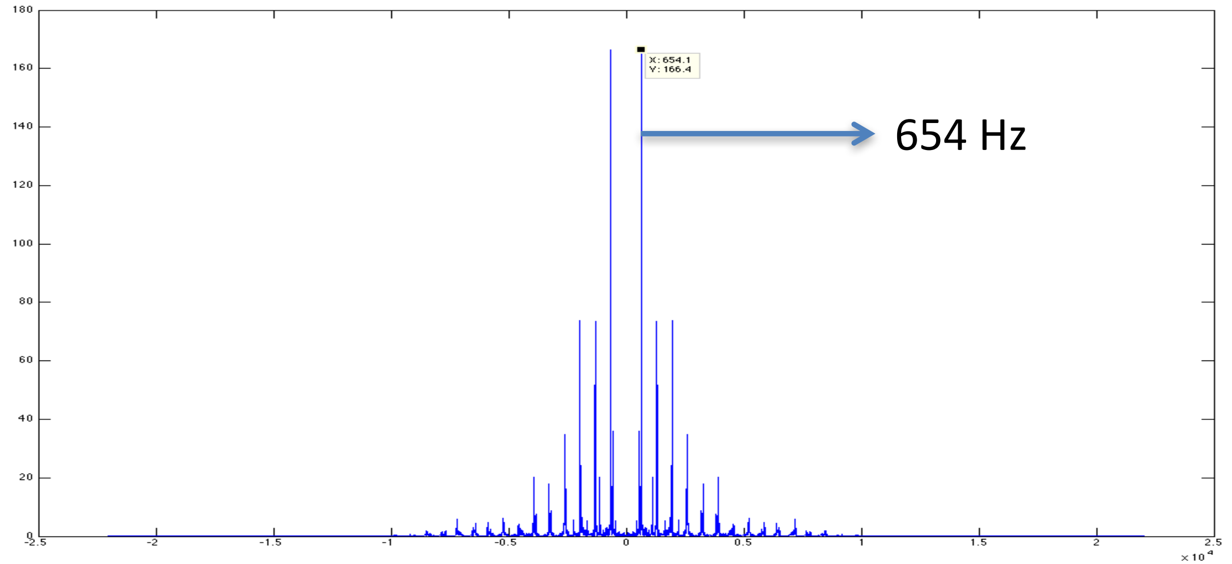


Note	Frequency	Wavelength
D_5	587.33	58.7
$D^{\#}_5/E^{b}_5$	622.25	55.4
E_5	659.26	52.3
F_5	698.46	49.4
$F^{\#}_5/G^{b}_5$	739.99	46.6
G_5	783.99	44.0
$G^{\#}_5/A^{b}_5$	830.61	41.5
A_5	880.00	39.2

864Hz ➔ A_5

Example of applying FFT

- Second note:



Note	Frequency	Wavelength
D_5	587.33	58.7
$D^{\#}_5/E^{b}_5$	622.25	55.4
E_5	659.26	52.3
F_5	698.46	49.4
$F^{\#}_5/G^{b}_5$	739.99	46.6
G_5	783.99	44.0
$G^{\#}_5/A^{b}_5$	830.61	41.5
A_5	880.00	39.2

654Hz ➔ E_5