CM2202: Scientific Computing and Multimedia Applications
Fourier Transform 1:
Digital Signal and Image Processing
Fourier Theory

Prof. David Marshall

School of Computer Science & Informatics

Fourier Transform

Moving into the Frequency Domain

The **Frequency domain** can be obtained through the transformation, via **Fourier Transform (FT)**, from

- one (**Temporal (Time)** or **Spatial**) domain

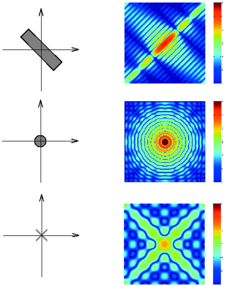
  to the other

- **Frequency** Domain
  - We do not think in terms of signal or pixel intensities but rather underlying sinusoidal waveforms of varying frequency, amplitude and phase.

Applications of Fourier Transform

Numerous Applications including:

- Essential tool for Engineers, Physicists, Mathematicians and Computer Scientists
- Fundamental tool for Digital Signal Processing and Image Processing
- Many types of Frequency Analysis:
    - **Filtering**
    - **Noise Removal**
    - Signal/Image Analysis
    - Simple implementation of **Convolution**
    - **Audio** and Image **Effects Processing**.
    - Signal/Image Restoration — *e.g.* **Deblurring**
    - Signal/Image Compression —– **MPEG** (Audio and Video), **JPEG** use related techniques.
    - Many more . . . . . .



Introducing Frequency Space

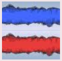
1D Audio Example

Lets consider a 1D (e.g. Audio) example to see what the different domains mean:

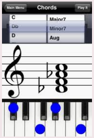
Consider a **complicated sound** such as the a **chord** played on a **piano** or a **guitar**.

We can describe this sound in two related ways:

Temporal Domain : Sample the **amplitude** of the sound many times a second, which gives an approximation to the sound as a **function** of **time**.



Frequency Domain : **Analyse** the sound in terms of the **pitches** of the notes, or **frequencies**, which make the sound up, recording the **amplitude** of **each frequency**.



Fundamental Frequencies

D♭	: 554.40Hz
F	: 698.48Hz
A♭	: 830.64Hz
C:	1046.56Hz

plus harmonics/partial frequencies ....

Back to Basics

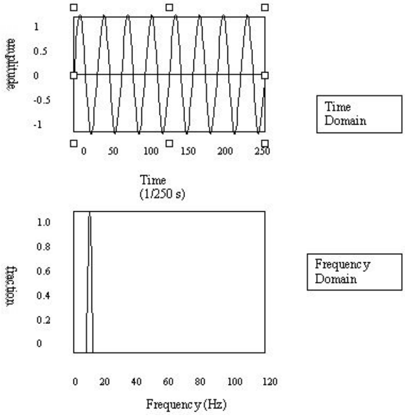
An 8 Hz Sine Wave

A signal that consists of a **sinusoidal** wave at **8 Hz**.

- 8 Hz means that wave is completing 8 cycles in 1 second
- The **frequency** of that wave is 8 Hz.

From the **frequency domain** we can see that the composition of our signal is

- **one peak** occurring with a frequency of 8 Hz — there is only one sine wave here.
  - with a **magnitude/fraction** of **1.0** i.e. it is the **whole signal**.



2D Image Example

What do Frequencies in an Image Mean?

Now images are no more complex really:

- **Brightness** along a **line** can be recorded as a set of **values** measured at **equally** spaced **distances apart**,
- **Or** equivalently, at a **set** of **spatial frequency values**.
- Each of these frequency values is a **frequency component**.
- An image is a 2D array of pixel measurements.
- We form a 2D grid of spatial frequencies.
  - A given frequency component now specifies what contribution is made by data which is changing with specified x and y direction spatial frequencies.

Frequency components of an image

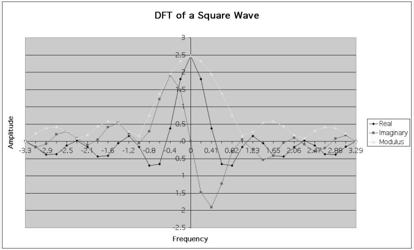
What do Frequencies in an Image Mean? (Cont.)

- Large values at **high** frequency components then the data is changing rapidly on a short distance scale.
  - *e.g.* a **page of text**
  - **However**, **Noise** contributes (very) **High Frequencies** also
- Large **low** frequency components then the large scale features of the picture are more important.
  e.g. a single fairly simple object which occupies most of the image.

Visualising Frequency Domain Transforms

Sinusoidal Decomposition

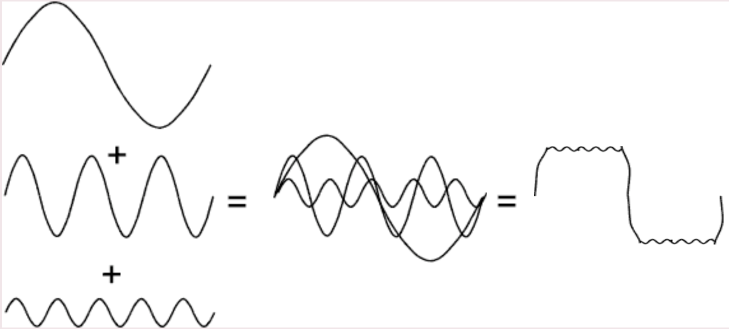
- **Any digital signal** (function) can be **decomposed** into purely **sinusoidal components**
  - Sine waves of different size/shape — varying **amplitude**, **frequency** and **phase**.
- When **added** back **together** they **reconstitute** the **original signal**.
- The **Fourier transform** is the tool that performs such an operation.



Summing Sine Waves. Example: to give a Square(ish) Wave

Digital signals are composite signals made up of many sinusoidal frequencies

- A 200Hz digital signal (**square(ish) wave**) may be a composed of 200, 600, 1000, *etc.* sinusoidal signals which sum to give:



Summary so far

So What Does All This Mean?

Transforming a signal into the frequency domain allows us

- **To see what sine waves make up our underlying signal**
- **E.g.**
  - One part sinusoidal wave at 50 Hz and
  - Second part sinusoidal wave at 200 Hz.
  - *Etc.*
- More **complex** signals will give more complex decompositions but the idea is exactly the same.

How is this Useful then?

Basic Idea of Filtering in Frequency Space

Filtering now involves **attenuating** or **removing** certain frequencies — **easily performed**:

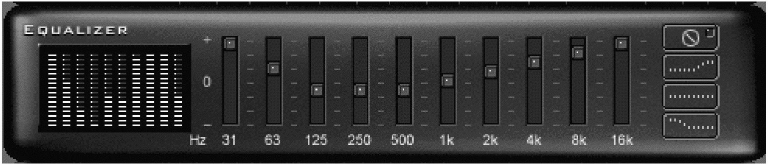
- **Low pass filter** —
  - **Ignore high frequency** noise components — make **zero** or a **very low value**.
  - Only store lower frequency components
- **High Pass Filter** — **opposite of above**
- **Bandpass Filter** — only **allow** frequencies in a **certain range**.

CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

Visualising the Frequency Domain

Think Graphic Equaliser

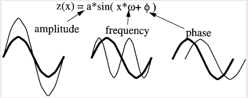
An easy way to visualise what is happening is to think of a graphic equaliser on a stereo system (or some software audio players, *e.g. iTunes*).



So are we ready for the Fourier Transform?

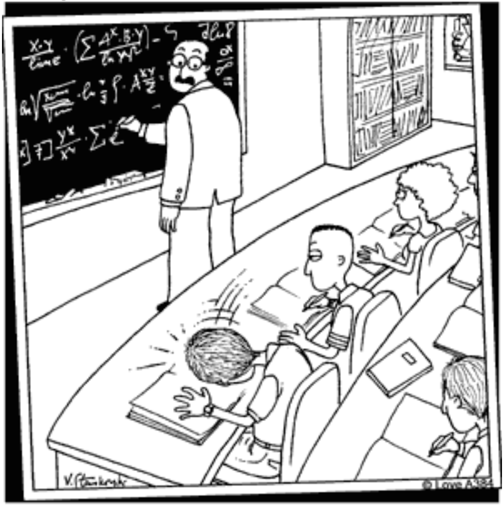
We have all the Tools....

- This lecture, so far, (hopefully) set the context for Frequency decomposition.
- Past Maths Lectures:
  - **Odd/Even Functions**: $\sin(-x) = -\sin(x)$, $\cos(-x) = \cos(x)$
  - **Complex Numbers**: **Phasor Form** $re^{i\phi} = r(\cos\phi + i\sin\phi)$
  - Calculus **Integration**: $\int e^{kx}\,dx = \frac{e^{kx}}{k}$
- Digital Signal Processing:
  - Basic Waveform Theory. Sine Wave $y = A.sin(2\pi.n.F_w/F_s)$ where: A = amplitude, F_w = wave frequency, F_s = sample frequency, n is the sample index.
  - Relationship between Amplitude, Frequency and Phase:



    - Cosine is a Sine wave 90° out of phase
  - Impulse Responses
- DSP + Image Proc.: Filters and other processing, Convolution

Snapshots at jasonlove.com



Professor Herman stopped when he heard that unmistakable thud -- another brain had imploded.

Fourier Theory

Introducing The Fourier Transform

The tool which **converts** a **spatial** or **temporal** (real space) **description** of **audio**/**image** data, for example, into one in terms of its **frequency components** is called the **Fourier transform**

The new version is usually referred to as the **Fourier space description** of the data.

We then essentially process the data:

- *E.g.* for **filtering** basically this means attenuating or setting certain frequencies to zero

We then need to **convert data back** (or **invert**) to **real audio**/imagery to use in our applications.

The corresponding **inverse** transformation which turns a Fourier space description back into a real space one is called the **inverse Fourier transform**.

1D Fourier Transform

1D Case (*e.g.* Audio Signal)

Considering a **continuous** function $f(x)$ of a single variable x representing distance (or time).

The **Fourier transform** of that function is denoted $F(u)$, where u represents **spatial** (or **temporal**) **frequency** is defined by:

$$F(u) = \int_{-\infty}^{\infty} f(x)\mathbf{e}^{-\mathbf{2}\pi \mathbf{ixu}}\, dx.$$

Note: In general $F(u)$ will be a complex quantity *even though* the original data is purely **real**.

- The meaning of this is that not only is the **magnitude** of each **frequency** present important, but that its **phase relationship** is **too**.
- Recall **Phasors** from **Complex Number Lectures**.
  - $e^{-2\pi ixu}$ above is a **Phasor**.

Inverse Fourier Transform

Inverse 1D Fourier Transform

The **inverse Fourier transform** for regenerating $f(x)$ from $F(u)$ is given by

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{\mathbf{2\pi i x u}} \, du,$$

which is rather similar to the (forward) Fourier transform

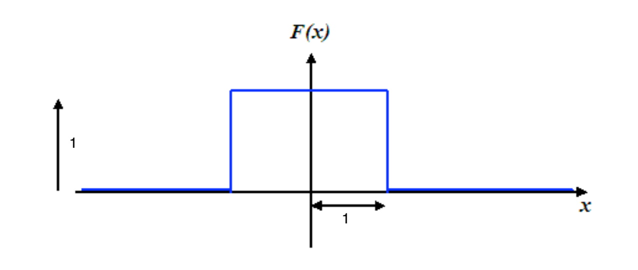
- except that the **exponential term has the opposite sign.**
- It is **not negative**

Fourier Transform Example

Fourier Transform of a Top Hat Function

Let's see how we compute a Fourier Transform: consider a particular function $f(x)$ defined as

$$f(x) = \begin{cases} 1 & \textbf{if } |x| \leq 1 \\ 0 & \textbf{otherwise,} \end{cases}$$



The Sinc Function (1)

We derive the Sinc function

So its Fourier transform is:

$$\begin{aligned} \mathbf{F(u)} &= \int_{-\infty}^{\infty} f(x) e^{-2\pi i x u}\, dx \\ &= \int_{-1}^{1} 1 \times e^{-2\pi i x u}\, dx \\ &= \frac{-1}{2\pi i u}(e^{2\pi i u} - e^{-2\pi i u}) \end{aligned}$$

Now (refer to **Complex Numbers** Lectures/**Maths Formula Sheet** Handout)

$$\begin{aligned} \sin\theta &= \frac{\mathbf{e^{i\theta} - e^{-i\theta}}}{\mathbf{2i}}, \text{ So:} \\ \mathbf{F(u)} &= \frac{\sin \mathbf{2\pi u}}{\mathbf{\pi u}}. \end{aligned}$$

In this case, $F(u)$ is purely real, which is a consequence of the original data being **symmetric** in x and $-x$.

- $f(x)$ is an **even** function.

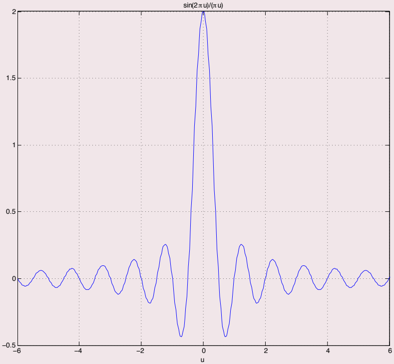
A graph of $F(u)$ is shown overleaf.

This function is often referred to as the **Sinc function**.

The Sinc Function Graph

The Sinc Function

The Fourier transform of a top hat function, the **Sinc function**:



The 2D Fourier Transform

2D Case (*e.g.* Image data)

If $f(x, y)$ is a function, for example **intensities** in an **image**, its **Fourier transform** is given by

$$F(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\mathbf{e}^{-2\pi \mathbf{i}(\mathbf{xu}+\mathbf{yv})}\, dx\, dy,$$

and the **inverse transform**, as might be expected, is

$$f(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u, v)e^{\mathbf{2}\pi \mathbf{i}(\mathbf{xu}+\mathbf{yv})}\, du\, dv.$$

The Discrete Fourier Transform

But All Our Audio and Image data are Digitised!!

Thus, we need a *discrete* formulation of the Fourier transform:

- **Assumes** **regularly spaced** data values, and
- **Returns** the **value** of the Fourier transform for a set of values in frequency space which are **equally spaced**.

This is done quite naturally by replacing the integral by a summation, to give the *discrete Fourier transform* or **DFT** for short.

1D Discrete Fourier transform

1D Case:

In 1D it is convenient now to assume that x goes up in steps of 1, and that there are N samples, at values of x from 0 to $N-1$.

So the DFT takes the form

$$F(u) = \frac{\mathbf{1}}{\mathbf{N}} \sum_{x=0}^{N-1} f(x) e^{-2\pi i x u/\mathbf{N}},$$

while the inverse DFT is

$$f(x) = \sum_{x=0}^{N-1} F(u) e^{2\pi i x u/\mathbf{N}}.$$

NOTE: Minor changes from the continuous case are a factor of $1/N$ in the **exponential** terms, and also the factor $1/N$ in front of the forward transform which **does not appear** in the **inverse** transform.

2D Discrete Fourier transform

2D Case

The **2D DFT** works is similar.

So for an $N \times M$ grid in x and y we have

$$F(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{1}}{\mathbf{NM}} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) e^{-2\pi i (x\mathbf{u}/\mathbf{N} + y\mathbf{v}/\mathbf{M})},$$

and

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u, v) e^{2\pi i (xu/\mathbf{N} + yv/\mathbf{M})}.$$

Balancing the 2D DFT

Most Images are Square

Often $N = M$, and it is then it is more convenient to redefine $F(u, v)$ by multiplying it by a factor of N, so that the **forward** and **inverse** transforms are more **symmetric**:

$$F(u, v) = \frac{\mathbf{1}}{\mathbf{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-2\pi i (xu+yv)/\mathbf{N}},$$

and

$$f(x, y) = \frac{\mathbf{1}}{\mathbf{N}} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{2\pi i (xu+yv)/\mathbf{N}}.$$

Fourier Transforms in MATLAB

fft() and fft2()

MATLAB provides functions for 1D and 2D **Discrete Fourier Transforms** (**DFT**):

- fft(X) is the 1D discrete Fourier transform (DFT) of **vector** X. For **matrices**, the FFT operation is applied to **each column** — **NOT** a 2D DFT transform.
- fft2(X) returns the 2D Fourier transform of matrix X. If X is a vector, the result will have the same orientation.
- fftn(X) returns the N-D discrete Fourier transform of the **N-D array X**.
- Inverse DFT **ifft()**, **ifft2()**, **ifftn()** perform the **inverse** DFT.

See appropriate MATLAB **help/doc** pages for **full details**.

Plenty of examples to Follow.

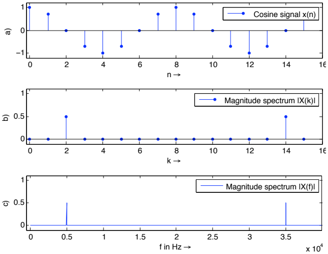
See also: **MALTAB Docs Image Processing → User's Guide → Transforms → Fourier Transform**

Visualising the Fourier Transform

Visualising the Fourier Transform

Having computed a DFT it might be useful to visualise its result:

- It's useful to visualise the Fourier Transform
- Standard tools
- Easily plotted in MATLAB



The Magnitude Spectrum of Fourier Transform

Recall that the Fourier Transform of our **real** audio/image data is always **complex**

- **Phasors**: This is how we encode the **phase** of the underlying signal's **Fourier Components**.

How can we visualise a complex data array?

Back to Complex Numbers:

Magnitude spectrum **Compute the absolute value of the complex data**:

$$|F(k)| = \sqrt{F_R^2(k) + F_I^2(k)} \textbf{ for } k = 0, 1, \ldots, N-1$$

where $F_R(k)$ is the **real** part and $F_I(k)$ is the **imaginary** part of the N sampled Fourier Transform, $F(k)$.

Recall MATLAB: `Sp = abs(fft(X,N))/N;`
(**Normalised form**)

The Phase Spectrum of Fourier Transform

The Phase Spectrum

Phase Spectrum

The Fourier Transform also represent phase, the **phase spectrum** is given by:

$$\varphi = \arctan \frac{F_I(k)}{F_R(k)} \textbf{ for } k = 0, 1, \ldots, N-1$$

Recall MATLAB: `phi = angle(fft(X,N))`

Relating a Sample Point to a Frequency Point

When **plotting graphs** of *Fourier Spectra* and doing other DFT processing we may wish to **plot** the x-axis in **Hz** (**Frequency**) rather than **sample point** number $k = 0, 1, \ldots, N - 1$

There is a **simple relation** between the two:

- The sample points go in steps $k = 0, 1, \ldots, N - 1$
- For a given sample point k the frequency relating to this is given by:

$$f_k = k\frac{f_s}{N}$$

  where f_s is the *sampling frequency* and N the **number** of samples.
- Thus we have **equidistant frequency steps** of $\frac{f_s}{N}$ ranging from 0 Hz to $\frac{N-1}{N} f_s$ Hz

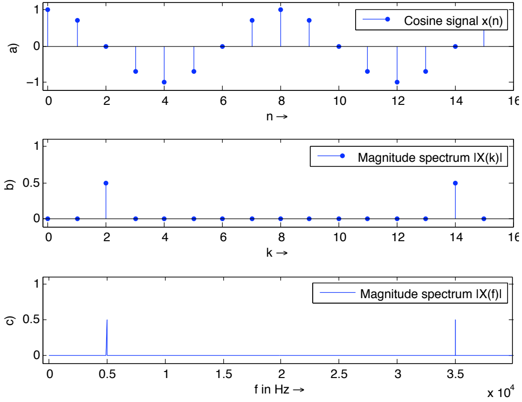
MATLAB Fourier Frequency Spectra Example

fourierspectraeg.m

```
N=16;
x=cos(2*pi*2*(0:1:N-1)/N)';

figure(1)
subplot(3,1,1);
stem(0:N-1,x,'.');
axis([-0.2 N -1.2 1.2]);
legend('Cosine signal x(n)');
ylabel('a)');
xlabel('n \rightarrow');

X=abs(fft(x,N))/N;
subplot(3,1,2);stem(0:N-1,X,'.');
axis([-0.2 N -0.1 1.1]);
legend('Magnitude spectrum |X(k)|');
ylabel('b)');
xlabel('k \rightarrow')

N=1024;
x=cos(2*pi*(2*1024/16)*(0:1:N-1)/N)';

FS=40000;
f=((0:N-1)/N)*FS;
X =abs(fft(x,N))/N;
subplot(3,1,3);plot(f,X);
axis([-0.2*44100/16 max(f) -0.1 1.1]);
legend('Magnitude spectrum |X(f)|');
ylabel('c)');
xlabel('f in Hz \rightarrow')

figure(2)
subplot(3,1,1);
plot(f,20*log10(X./(0.5)));
axis([-0.2*44100/16 max(f) ...
-45 20]);
legend('Magnitude spectrum |X(f)| ...
in dB');
ylabel('|X(f)| in dB \rightarrow');
xlabel('f in Hz \rightarrow')
```

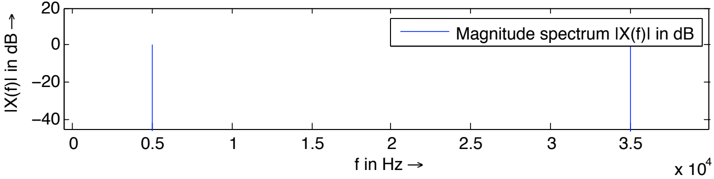
MATLAB Fourier Frequency Spectra Example Output

fourierspectraeg.m produces the following:



Magnitude Spectrum in dB

Note: It is common to plot both spectra magnitude (also frequency ranges not show here) on a dB/log scale:
(Last Plot in fourierspectraeg.m)



Time-Frequency Representation: Spectrogram

Spectrogram

It is often **useful** to look at the **frequency distribution** over a **short-time**:

- Split signal into N segments
- Do a **windowed Fourier Transform** — **Short-Time Fourier Transform** (**STFT**)
  - Window needed to reduce *leakage* effect of doing a shorter sample SFFT.
  - Apply a **Blackman**, **Hamming** or **Hanning** Window
- MATLAB function does the job: `Spectrogram` — see `help spectrogram`
- See also MATLAB's `specgramdemo`

MATLAB spectrogram Example

spectrogrameg.m

```
load('handel')
[N M] = size(y);
figure(1)
spectrogram(y,512,20,1024,Fs);
```

Produces the following: