Fourier transforms in digital signal processing
Real space: numbers at points in time or space

1D array

7 2 9 10

shape: (4,)

2D array

5.2 3.0 4.5
9.1 0.1 0.3

shape: (2, 3)

3D array

shape: (4, 3, 2)

Induced voltage

Time

Image 1

Image 2

DkTx
Real space is an intuitive representation of data

How fast is the signal decaying?
How fast is the signal oscillating?
How many oscillators make up the signal?

Where are the particles located?
What is the shape of the particles?
Is this image blurred and distorted?
How does this image relate to the structure?

...but some questions are harder to pose
(1) There many ways of representing numerical data

(2) Fourier space is an alternative representation based on waves of different frequency

(3) Many find Fourier space initially unintuitive

(4) Many hard problems become easier to understand and solve in Fourier space
Linear combinations

Most functions can be represented as weighted sums of other functions.

\[ S(x) = w_1 F_1(x) + w_2 F_2(x) + \ldots + w_N F_N(x) = \sum_{i=1}^{N} w_i F_i(x) \]

Some function of the variable \( x \)

Basis functions:
- a set of functions that can be combined to form other functions.
- A.k.a. components, dimensions
**Linear combinations**

Most functions can be represented as weighted sums of other functions.

\[ S(x) = w_1 F_1(x) + w_2 F_2(x) + \ldots + w_N F_N(x) = \sum_{i=1}^{N} w_i F_i(x) \]

- **Some function of the variable** \( x \)
- **Basis functions**: a set of functions that can be combined to form other functions.
- **Coefficients**: how much \( F_i(x) \) is in \( S(x) \)?
  - A.k.a. **weights or coordinates**
  - A.k.a. **components, dimensions**
Reminder of dot product and inner product

Discrete

dot product: \[ c = \sum_k u_k v_k = u \cdot v \]

Example:
\[ x = [1, 4, 1] \]
\[ y = [7, 2, 6] \]
\[ \text{dot\_product}(x, y) = 1*7 + 4*2 + 1*6 = 21 \]

Continuous

inner product \[ c = \int f(t) g(t) \, dt = (f, g) \]
Orthonormal linear basis functions

Orthonormality conditions
Dot product of F1 and F2 in a basis is always zero.
Dot product of F1 and F1 is always one'
ex. \( \mathbf{x} = [1 \ 0 \ 0] \), \( \mathbf{y} = [0 \ 1 \ 0] \), and \( \mathbf{z} = [0 \ 0 \ 1] \)

\[
\begin{align*}
\text{dot}_\text{product}(\mathbf{x},\mathbf{y}) &= 1*0 + 0*1 + 0*0 = 0 \\
\text{dot}_\text{product}(\mathbf{x},\mathbf{z}) &= 1*0 + 0*0 + 0*1 = 0 \\
\text{dot}_\text{product}(\mathbf{y},\mathbf{z}) &= 0*0 + 1*0 + 0*1 = 0 \\
\text{dot}_\text{product}(\mathbf{x},\mathbf{x}) &= 1*1 + 0*0 + 0*0 = 1
\end{align*}
\]

If basis vectors are mutually orthonormal, we can determine the coefficients for a function \( S \) simply by taking the dot product of the function and the basis functions:

ex. \( S = [3 \ 2 \ 6] \)
\[
\begin{align*}
\mathbf{w}_x &= \text{dot}_\text{product}(S,\mathbf{x}) = 3*1 + 2*0 + 6*0 = 3 \\
\mathbf{w}_y &= \text{dot}_\text{product}(S,\mathbf{y}) = 3*0 + 2*1 + 6*0 = 2 \\
\mathbf{w}_z &= \text{dot}_\text{product}(S,\mathbf{z}) = 3*0 + 2*0 + 6*1 = 6 \\
\mathbf{S} &= \mathbf{w}_x * \mathbf{x} + \mathbf{w}_y * \mathbf{y} + \mathbf{w}_z * \mathbf{z} = [3 \ 0 \ 0] + [0 \ 2 \ 0] + [0 \ 0 \ 6] = [3 \ 2 \ 6]
\end{align*}
\]

Geometric interpretation:
\( \text{dot}_\text{product}(\mathbf{x},\mathbf{y}) = 0 \) means \( \arccos(x,y) = \pi / 2 = 90^\circ \)
“a right angle between \( x \) and \( y \)”
The linear orthogonal basis for Fourier space:
Waves with different frequencies

Square wave function
The linear orthogonal basis for Fourier space:

Waves with different frequencies

<table>
<thead>
<tr>
<th>Wave frequency</th>
<th>Wave phase (offset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_k(x) = \sin(2\pi k x + \phi)$</td>
<td></td>
</tr>
</tbody>
</table>

Square wave function
Properties of waves

Waves are represented with sine and cosine functions over time or space.

Wave frequency

Multiplying the argument of the sine function changes the frequency of the wave
Properties of waves

Sine waves can have different phases

The wave phase (offset) is given by

$$F_i(x) = \sin(2\pi k x + \phi_i)$$

Adding to the argument of the sine function adds an offset to the wave called the ‘phase’.
Properties of waves

Phase shifts are less than $2\pi$

\[
\sin(\theta + \pi) = -\sin \theta
\]
\[
\sin(\theta + 2\pi) = +\sin \theta
\]

Sines and cosines are related by a $90^\circ (\pi/2)$ phase shift

\[
\sin(\theta + \frac{\pi}{2}) = +\cos \theta
\]
\[
\cos(\theta + \frac{\pi}{2}) = -\sin \theta
\]
Properties of waves

Sine waves can have **different amplitudes**: these will be coefficients of our combination linear combinations

\[ \omega_i F_i(x) = A_i \sin(2\pi kx + \phi_i) \]

\[ A(\text{Louder}) > A(\text{Softer}) \]
Properties of waves

**Polar coordinates**
‘Amplitude - phase’ coordinates

\[ M \cos(x + \theta) \]

**Rectangular coordinates**
‘Sine - cosine’ coordinates

\[ A \cos(x) + B \sin(x) \]

**Convenient identities**
\[
M = \left( A^2 + B^2 \right)^{\frac{1}{2}} \\
\theta = \arctan(B/A) \\
A = M \cos(\theta) \\
B = M \sin(\theta)
\]

Scientists prefer to think in **polar coordinates**

Computer programs typically output **rectangular coordinates**.
Waves are also commonly represented by exponential functions using Euler’s formula.

$$e^{ix} = \cos x + i \sin x$$
Waves of different frequency form an orthonormal basis

\[ \frac{1}{L} \int_{-L}^{L} \cos\left(\frac{n \pi}{L} t\right) \cos\left(\frac{m \pi}{L} t\right) \, dt = \begin{cases} 1 & n = m \neq 0 \\ 0 & n \neq m \\ 2 & n = m = 0 \end{cases} \]

\[ \frac{1}{L} \int_{-L}^{L} \cos\left(\frac{n \pi}{L} t\right) \sin\left(\frac{m \pi}{L} t\right) \, dt = 0 \]

\[ \frac{1}{L} \int_{-L}^{L} \sin\left(\frac{n \pi}{L} t\right) \sin\left(\frac{m \pi}{L} t\right) \, dt = \begin{cases} 1 & n = m \neq 0 \\ 0 & n \neq m \end{cases} \]

In words: “The inner product (dot product) of two sine or cosine functions is zero if they have different frequencies.”

Proof of the orthogonality relations: This is just a straightforward calculation using the periodicity of sine and cosine and either (or both) of these two methods:

Method 1: use \( \cos at = \frac{e^{iat} + e^{-iat}}{2} \), and \( \sin at = \frac{e^{iat} - e^{-iat}}{2i} \).

Method 2: use the trig identity \( \cos(a) \cos(b) = \frac{1}{2}(\cos(a + b) + \cos(a - b)) \), and the similar trig identities for \( \cos(a) \sin(b) \) and \( \sin(a) \sin(b) \).
Fourier’s big idea

Any periodic function can be represented by a linear combination of sine and cosine wave functions.

Continuous functions may require infinite waves. Discrete functions (real-world data) can be exactly represented with a finite sum of waves. (N/2+1 sines and N/2+1 cosines)
The Fourier synthesis equation

The function $S(x)$ is equal to a weighted sum of sines and cosines of increasing frequency, $k$. The weights are the coefficients $a_k$ and $b_k$

\[
S(x) = \frac{1}{N} \sum_{k=0}^{N-1} a_k \sin\left(\frac{2\pi k x}{N}\right) + i b_k \cos\left(\frac{2\pi k x}{N}\right) = \frac{1}{N} \sum_{k=0}^{N-1} A_k e^{i2\pi k x + \phi_k}
\]

Rectangular coordinates  
Exponential coordinates
The Fourier analysis equation

We can solve for the coefficients $a_k$ and $b_k$ by calculating the dot product of $S(x)$ with a wavefunction at the frequency $k$.

$$X(k) = a_k + ib_k = \sum_{x=0}^{N-1} S(x)e^{-i2\pi k x / N}$$

$X(k)$ is a **frequency-domain** representation of the **real-space** function $S(x)$. You might also hear **Fourier-space** or **reciprocal space** representation.
Shannon's sampling theorem

For a discrete FT, what are the frequencies $k$?

First we need the real-space **sampling rate**, $d$

elements

- $d = 1$ second / sample (temporal signal, like a sound)
- $d = 1$ angstrom / pixel (spatial signal, like an image)

We also need the **number of samples**, $N$

The FT will have $N/2+1$ **frequencies**, $k$. The units will be $1/d$ and they will run linearly from 0 to $1/2d$.

The frequency $k=1/(2d)$ is the **Nyquist frequency**. It is the highest possible frequency sinusoid that can be correctly represented at sampling rate $d$. 
Examples of discrete wavefunctions

\( \cos(k=0) \) is a constant value. Its coefficient is the mean of the real-space data. Sometimes it’s called the DC component.

\( \sin(k=0) \) is always zero.

The component at the nyquist frequency, \( \cos(k=1/(2d)) \) is a function that alternates each pixel between -1 and 1.

\( \sin(k=1/(2d)) \) is always zero.
Real-space representation of data:
Values at points in time/space

Fourier-space representation of data:
Coefficients of waves of different frequencies.

The Fourier analysis equation,
aka. **The Fourier Transform**

The Fourier synthesis equation,
aka. **The Inverse Fourier Transform**
Fourier transforms can also be calculated for 2D functions like images: $S(x,y)$

$$X(k, l) = \sum_{x=0}^{N_x-1} \sum_{y=0}^{N_y-1} S(x, y) e^{-i2\pi(kx/N_x + ly/N_y)}$$
Usually when we represent a 2D Fourier transform, we put low spatial frequencies near the center, high spatial frequencies farther away.
Real-space image

Amplitudes of Fourier transform

Fourier transform of images

Higher frequency waves
Higher resolution details
Fourier transform of images

Amplitudes of Fourier transform

Same frequency waves but different directions
Fourier transform of images

Amplitudes of Fourier transform

Higher frequency waves in different directions
Fourier transform of images

The Phase Problem: Animal Magic

A duck

A cat
Fourier transform of images

The Phase Problem: Animal Magic

Combine the magnitudes from the Duck transform with the phases from the Cat transform.
Fourier transform of images

The Phase Problem: Animal Magic

Combine the magnitudes from the Duck transform with the phases from the Cat transform

Combine the magnitudes from the Cat transform with the phases from the Duck transform

The phase contains the bulk of the information!

Kevin Cowtan  http://www.ysbl.york.ac.uk/~cowtan/
John Tukey’s Fast Fourier Transform (FFT) algorithm
One of the greatest algorithm of all time

For the discrete Fourier transform, we convert an array with N elements to N/2+1 sines and cosines. Each sine/cosine pair requires the dot product over all N elements, we require N*(N/2) operations. The FFT solves the same problem in N*\log(N) operations, making it ‘cheap’ even for very large N.
Scientific computing libraries have highly optimized implementations of the FFT.

**real FFT:**
Produces $N/2-1$ coefficients

**complex FFT:**
Produces $N$ coefficients, but one side of the FFT is exactly the same as the other side.
Spectral analysis: which waves are in a signal?

The fastest wave that can be represented with 500 samples has 250 oscillations.

Nyquist frequency = 0.5 = 250

The frequency of each plotted wave is:

\[
\text{wave1} = \frac{100}{500} = 0.2 \\
\text{wave2} = \frac{50}{500} = 0.1 \\
\text{wave3} = \frac{80}{500} = 0.16
\]

Can we determine these values from the data itself using the FFT?
Spectral analysis: which waves are in a signal?

In a realistic case, we’ll also have noise or other processes occurring.

We can thus add random noise to our wave sum to simulate these processes.

Now there’s no way you could guess the frequencies from just looking!
Spectral analysis: which waves are in a signal?

**Power** of a signal at frequency $k = \text{squared amplitude}$ at frequency $k$

The **power spectrum** is the power as a function of $k$

Calculate the power of a Fourier coefficient by multiplying $a+ib$ by its complex conjugate $a-ib$:

$$(a + bi)(a - bi) = a^2 + b^2$$

Welch’s algorithm for estimating the power spectrum:
1. Divide signal up into overlapping patches.
2. Calculate the FT of each patch
3. Calculate $PS = \text{FT} \ast \text{FT.conj}()$ (python syntax)
4. Average all the PS together

Using python’s implementation of Welch’s algorithm, we see peaks at 0.1, 0.16, and 0.2 as expected!
Why would we want to do all this work?

FT applications in cryo-EM
Linear systems and convolution

Real-space signal $f(x)$ of three events/objects at sharp points

Point-spread function $p(x)$

Describes how a point is transformed by a linear system. Also called the impulse-response function or the convolution kernel.

Convolution of $f(x)$ with $p(x)$:
Each point is multiplied by the point-spread function:

$$f(x) * p(x) = y(x)$$

* stands for convolution, not multiplication.
The Fourier convolution theorem

\[ f(x) * p(x) = y(x) \]

Convolution in real-space is computationally challenging.

\[ F(k) = \text{FT}(f(x)) \]
\[ P(k) = \text{FT}(f(x)) \]

However, in Fourier space, convolution is an elementwise multiplication of the FT of the signal and FT of the PSF.

\[ F(k)P(k) = Y(k) \]

In cryo-EM, the FT of the PSF is called the **Contrast Transfer Function** (CTF).

\[ \text{IFT}(Y(k)) = y(x) \]

The CTF corresponds to the way the electron optical system distorts...
Fourier transforms let us do efficient convolutions.
Fourier transforms let us do efficient convolutions

\[ \text{FT} \xrightarrow{\ast} \text{FT} \xrightarrow{\times} \text{FT}^{-1} = \]
Scattering and lensing are like Fourier transforms

The linear Fourier imaging model

\[ f(x) = \text{projection image of electrons through the specimen} \]

\[ F(k) = \text{FT}( f(x) ) \]

\[ Y(k) = F(k) \text{CTF}(k) + \text{noise}(k) \]

\[ y(x) = \text{IFT}( Y(k) ) \]

\[ y(x) = \text{image we record on our electron cameras} \]
The 2D Fourier transform lets us separate information at different scales in images.
The 2D Fourier transform lets us separate information at different scales in images.
The 2D Fourier transform lets us separate information at different scales in images.

\[ f(x,y) \]

\[ |F(u,v)| \]
The 2D Fourier transform lets us separate information at different scales in images.
The 2D Fourier transform lets us separate information at different scales in images.
Fourier transforms let us align noisy images

The **matched-filter** or **cross-correlation** algorithm:

$$\text{FT}^{-1}[ \text{FT}(M) \times \text{FT}(T)^* ] = \text{cross-correlation map}$$

- **M**: a noisy, off-center particle image
- **T**: a clean, centered template image
- The cross-correlation map with localization peak

The better the match, the larger the peak
We can divide an image into resolution shells and compute cross-correlation for each shell.

If we have two images, we can use the **Fourier Shell Correlation (FSC)** curve to find the resolution where they become inconsistent with each other.

FSC curves:
- resolution-dependent cross-correlation
- resolution-dependent consistency metric for structures
- commonly used as measure of resolution

\[
FSC(r) = \frac{\sum_{r_i \in r} F_1(r_i) \cdot F_2(r_i)^*}{\sqrt{\sum_{r_i \in r} |F_1(r_i)|^2 \cdot \sum_{r_i \in r} |F_2(r_i)|^2}}
\]
Fourier transforms let us average 2D images in 3D.

In electron microscopy, we want to average 2D projection images together to form a 3D volumetric ('density') image… but how?

**Forward process:** e- beam forms projection image from 3D structure

**Inverse problem:** Can we recover 3D structure from projection images?
Fourier transforms let us average 2D images in 3D

Images can be averaged in real-space or averaged in reciprocal space and then inverse Fourier transformed

\[ S_1(x) + S_2(x) = FT^{-1}(X_1(k) + X_2(k)) \]
Fourier transforms let us average 2D images in 3D

The projection-slice theorem:
If I take a 2D projection through a 3D object and fourier transform it, I get a 2D slice of the object’s 3D fourier transform.

If we project from a different viewpoint, the slice we get is from that viewpoint.
Fourier transforms let us average 2D images in 3D.

This gives leads to a **reconstruction algorithm** for solving EM structures:

1. Project an initial 3D density from many views
   *use the projection-slice theorem*

2. Match experimental images to projections
   *use the cross-correlation algorithm*

3. Calculate a new 3D density from aligned images
   *use the projection-slice theorem*

4. Iterate this process until 3D density stops improving
Fourier transforms let us average 2D images in 3D.