ENGR 510 Signals Homework

Anthony Su

December 6, 2024

1 Building an Orthogonal Cosine Basis

1.a

Let $f(x) = \cos(\pi m x)$ and let $g(x) = \cos(\pi n x)$ where $m \neq n$ and $m, n \in \mathbb{Z}$. Then, on the interval $[0, 1]$,

$$\begin{aligned}
\langle f, g\rangle &= \int_0^1 f(x)g(x)\mathrm{d}x \\
\langle f, g\rangle &= \int_0^1 \cos(\pi m x)\cos(\pi n x)\mathrm{d}x \\
\langle f, g\rangle &= \frac{1}{2\pi}\left(\frac{\sin(\pi x(m-n))}{m-n} + \frac{\sin(\pi x(m+n))}{m+n}\right)\Bigg|_0^1 \\
\langle f, g\rangle &= \frac{1}{2\pi}\left(\frac{\sin(\pi(m-n))}{m-n} + \frac{\sin(\pi(m+n))}{m+n} - \frac{\sin(0)}{m-n} - \frac{\sin(0)}{m+n}\right)
\end{aligned}$$

Since m and n are integers, $\sin(\pi(m \pm n)) = 0$. Simplifying,

$$\langle f, g\rangle = 0$$

$$\boxed{\text{Thus, } \cos(\pi m x) \text{ and } \cos(\pi n x) \text{ are orthogonal for } m \neq n \text{ and } m, n \in \mathbb{Z}.}$$

1.b

```python
x_vec = np.arange(0, 1.01, 0.01)
fig, ax = plt.subplots(figsize=(5, 2.5), layout='constrained')
for m in range(6):
    y_vec = np.cos(np.pi * m * x_vec)
    ax.plot(x_vec, y_vec, label=f'm={m}')
ax.set_xlabel('x')
ax.set_ylabel('f(x)')
ax.legend()
ax.grid(True)
fig.savefig('1bfig1.png', dpi=300)
plt.show()
```

Listing 1: Python code to generate Figure 1

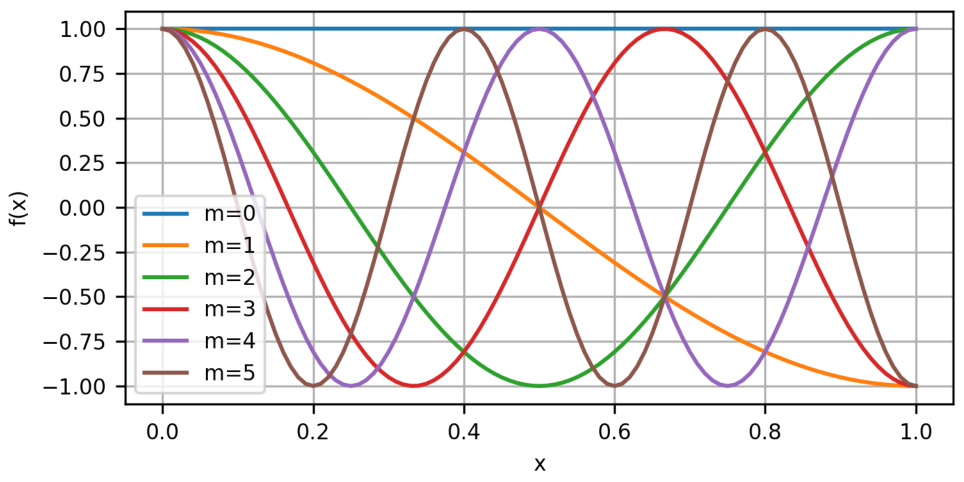

Figure 1: $f(x) = \sin(\pi m x)$ for d$x = 0.01$ and $m = \{0, 1, 2, 3, 4, 5\}$

```python
mn_pairs = [(1, 4), (2, 6), (3, 15)]
for m, n in mn_pairs:
    x = np.arange(0, 1.01, 0.01)
    f = np.sin(m * np.pi * x)
    g = np.sin(n * np.pi * x)
    inner_prod = integrate.trapezoid(y=f*g, x=x)
    print(f"Inner product of m={m} and n={n} is: {inner_prod}")
```

```
Inner product of m=1 and n=4 is: -1.0963994469259664e-17
Inner product of m=2 and n=6 is: -1.3227266504323154e-17
Inner product of m=3 and n=15 is: -2.439454888092385e-18
```

Listing 2: Python code to compute inner products

2 Inner Product, Revisited

2.a

Sine and cosine waves are orthogonal.

$\langle a, b \rangle$ is zero

2.b

Although dissimilar, the signals are in phase.

$\langle b, c \rangle$ is close to one

2.c

$a(x)$ is odd and $d(x)$ is even; they are orthogonal.

$\boxed{\langle a, d \rangle \text{ is zero}}$

2.d

$f(x)$ is a rescaled $a(x)$. The inner product is just the product of the magnitudes.

$\boxed{\langle a, f \rangle \text{ is greater than one}}$

3 Image Compression

3.b

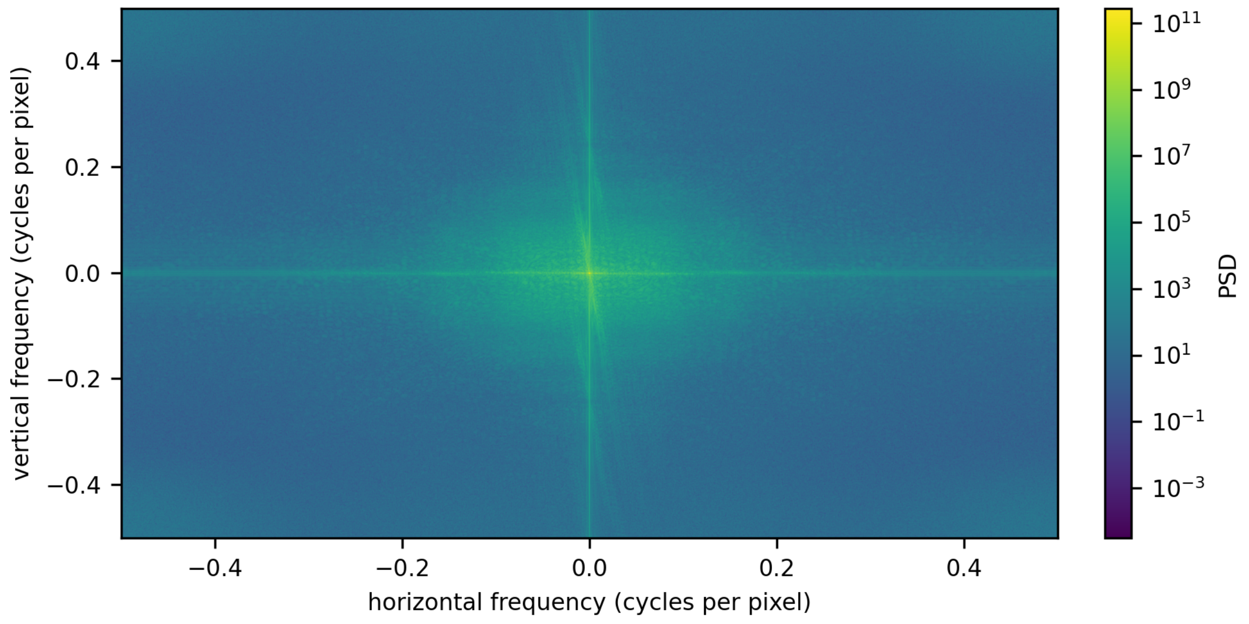

Figure 2: Power spectral density of Fourier coefficients of image

3.c

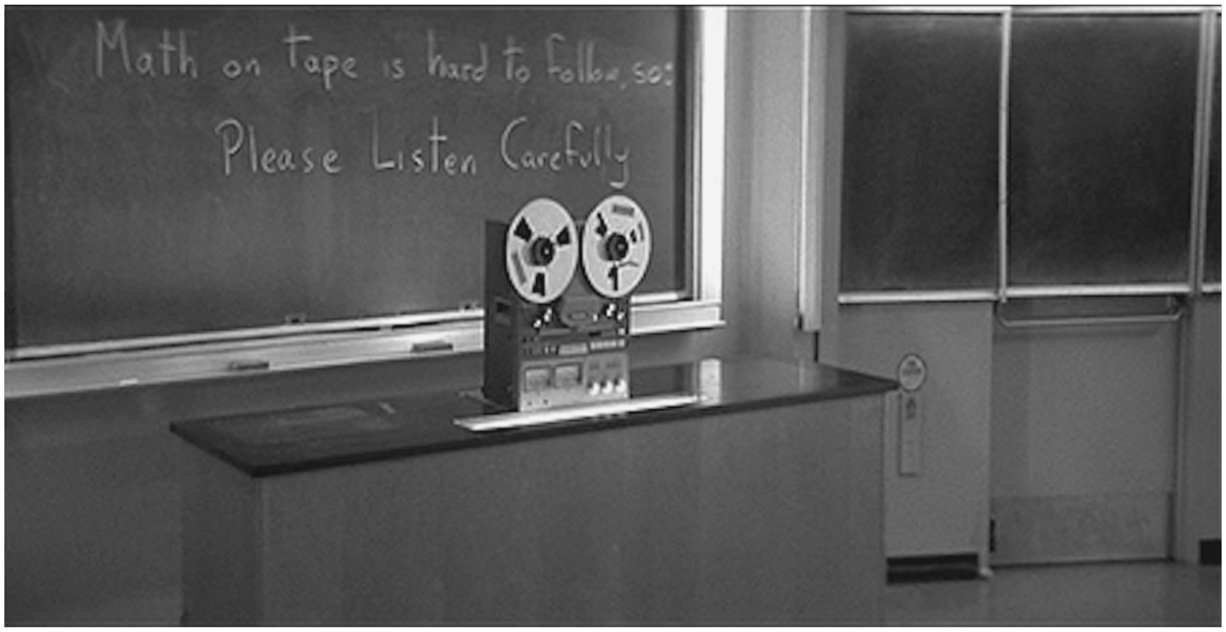

Figure 3: Image with 10% of original Fourier coefficients

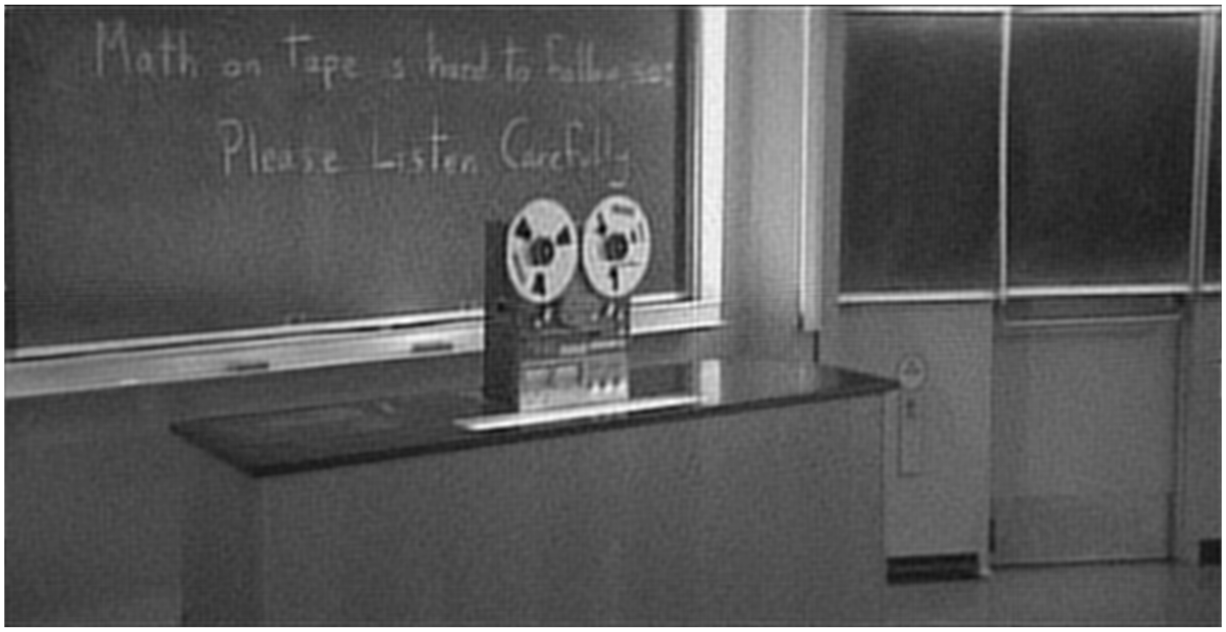

Figure 4: Image with 1% of original Fourier coefficients

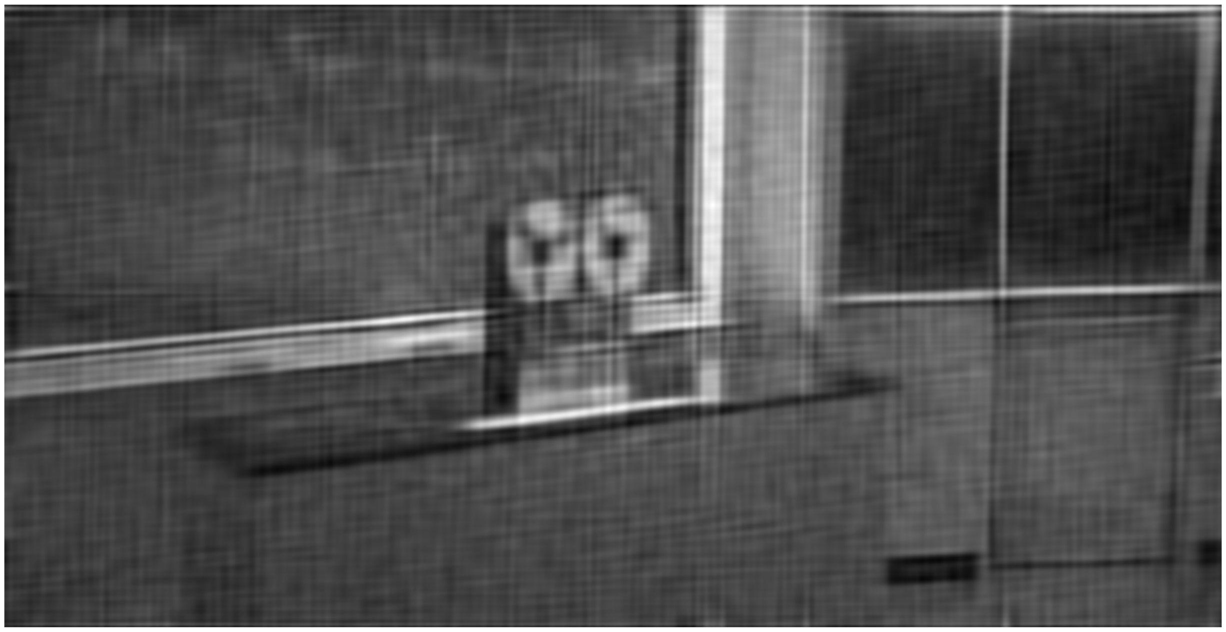

Figure 5: Image with 0.2% of original Fourier coefficients

3.d

Keeping 10% of the Fourier coefficients yields an image which is visually indiscernible from the original because the magnitude of the truncated Fourier coefficients is negligible.

Keeping 1% of the Fourier coefficients yields an image which is visually less detailed and more noisy than the original because the magnitude of the truncated Fourier coefficients is significant.

Keeping 0.2% of the Fourier coefficients yields an image which is lacking major details/features because large Fourier coefficients have been truncated, significantly altering the result. Stripe-like artifacts appear which result from the interference patterns of the remaining low-frequency components.

4 Fourier Series vs Fourier Transform

4.a

The Fourier series representation of $f(x)$ on the interval with width $L = 4$ is

$$f(x) = \frac{A_0}{2} + \sum_{k=1}^{\infty} \left[A_k \cos\left(\frac{2\pi k}{L} x\right) + B_k \sin\left(\frac{2\pi k}{L} x\right) \right]$$

where A_0 is given by

$$\begin{aligned}
A_0 &= \frac{2}{L}\int_{L/2}^{L/2} f(x)\mathrm{d}x \\
A_0 &= \frac{2}{4}\int_{-2}^{2} f(x)\mathrm{d}x \\
A_0 &= \frac{1}{2}\left[\int_{-2}^{-1} 0\mathrm{d}x + \int_{-1}^{1} 1\mathrm{d}x + \int_{1}^{2} 0\mathrm{d}x\right] \\
A_0 &= 1
\end{aligned}$$

and where A_k is given by

$$\begin{aligned}
A_k &= \frac{2}{L}\int_{L/2}^{L/2} f(x)\cos\left(\frac{2\pi k}{L}x\right)\mathrm{d}x \\
A_k &= \frac{2}{4}\left[\int_{-2}^{-1} 0\cdot\cos\left(\frac{2\pi k}{4}x\right)\mathrm{d}x + \int_{-1}^{1} 1\cdot\cos\left(\frac{2\pi k}{4}x\right)\mathrm{d}x + \int_{1}^{2} 0\cdot\cos\left(\frac{2\pi k}{4}x\right)\mathrm{d}x\right] \\
A_k &= \frac{1}{2}\int_{-1}^{1}\cos\left(\frac{\pi k}{2}x\right)\mathrm{d}x \\
A_k &= \frac{1}{2}\left[\frac{\sin\left(\frac{\pi k}{2}x\right)}{\pi k/2}\right]_{-1}^{1} \\
A_k &= \frac{1}{\pi k}\left[\sin\left(\frac{\pi k}{2}\right) - \sin\left(-\frac{\pi k}{2}\right)\right] \\
A_k &= \frac{1}{\pi k}\left[\sin\left(\frac{\pi k}{2}\right) + \sin\left(\frac{\pi k}{2}\right)\right] \\
A_k &= \frac{2}{\pi k}\sin\left(\frac{\pi k}{2}\right)
\end{aligned}$$

and where B_k is given by

$$\begin{aligned}
B_k &= \frac{2}{L}\int_{L/2}^{L/2} f(x)\sin\left(\frac{2\pi k}{T}x\right)\mathrm{d}x \\
B_k &= 0
\end{aligned}$$

since $f(x)\cdot\sin\left(\frac{2\pi k}{T}x\right)$ is an odd function.

Thus, the Fourier series representation of $f(x)$ on the interval $[-2,2]$ is

$$\boxed{f(x) = \frac{1}{2} + \sum_{k=1}^{\infty}\frac{2}{\pi k}\sin\left(\frac{\pi k}{2}\right)\cos\left(\frac{\pi k}{2}x\right)}$$

4.b

```python
# Compute fourier series mode coefficients
k = np.arange(1, 101)
Ak = 2/(np.pi*k)*np.sin(np.pi*k/2)
Bk = k*0

# Plot fourier series mode coefficients
fig, ax = plt.subplots(figsize=(6.5, 3.5), layout='constrained')
ax.plot(k, Ak, '.', label='$A_k$')
ax.plot(k, Bk, '+', label='$B_k$')
ax.set_xlabel('$k$')
ax.set_ylabel('Mode coefficient')
ax.legend()
ax.grid(True)
fig.savefig('4bfig1.png', dpi=300)
plt.show()
```

Listing 3: Python code to generate Figure 6

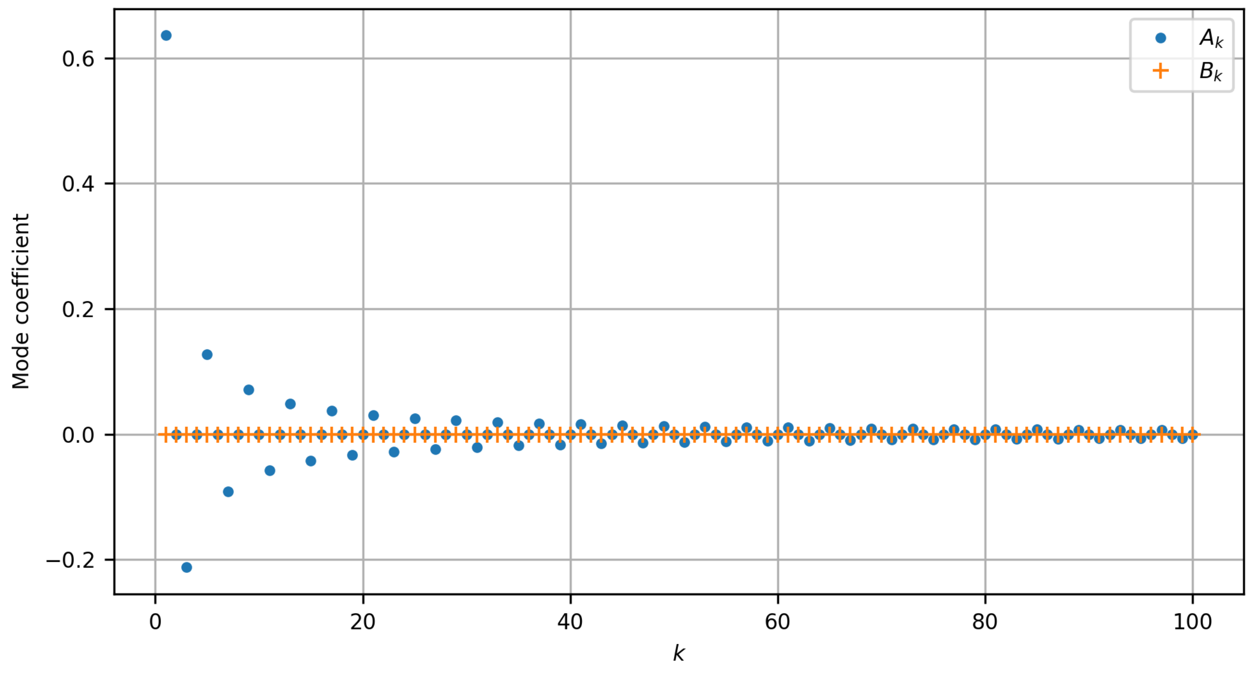

Figure 6: Fourier series mode coefficients of $f(x)$

4.c

```python
# Define square wave function
    def square(x):
        return np.mod(x-1, 4) >= 2

    # Define fourier series square wave approximation function
    def square_approx(x, n):
        sum = 0.5 + np.zeros(x.shape)
        for k in range(1, n+1):
            sum += 2/(np.pi*k) * np.sin(np.pi*k/2) * np.cos(np.pi*k/2*x)
        return sum

    # Plot square wave function
    dx = 0.01
    x_vec = np.arange(-2, 2+dx, dx)

    fig, ax = plt.subplots(figsize=(5, 3), layout='constrained')
    ax.plot(x_vec, square(x_vec), 'k--', label='truth')
    ax.plot(x_vec, square_approx(x_vec, 10), label='Fourier series approx.')
    ax.set_xlabel('$x$')
    ax.set_ylabel('$f(x)$')
    ax.legend()
    fig.savefig('4cfig1.png', dpi=300)
    plt.show()
```

Listing 4: Python code to generate Figure 7

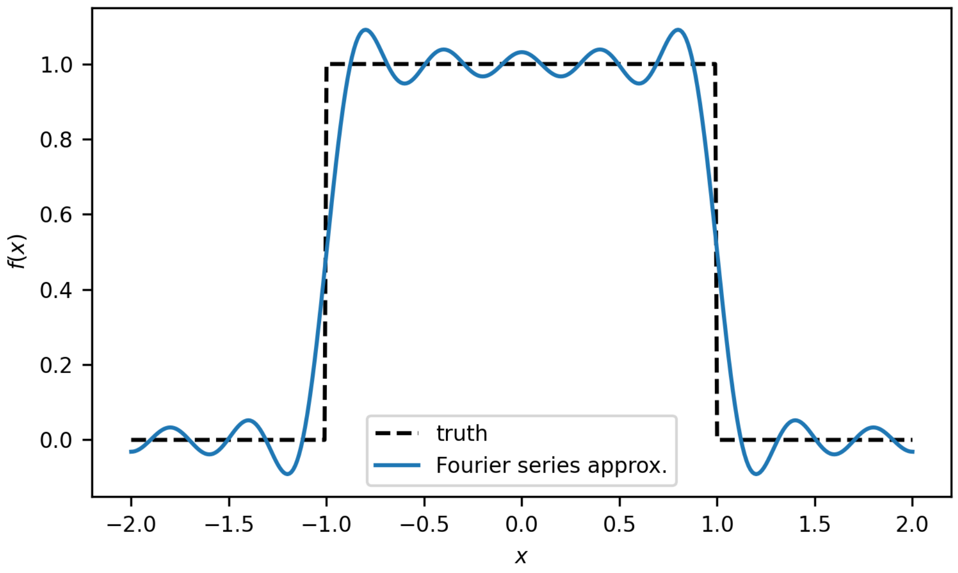

Figure 7: $k = 10$ Fourier series approximation of $f(x)$

4.d

Assuming that $f(x)$ is zero for all $x < -1$ and $x > 1$,

$$\mathcal{F}(f(x)) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} \mathrm{d}x$$

$$\mathcal{F}(f(x)) = \int_{-\infty}^{-1} 0 \cdot e^{i\omega x} \mathrm{d}x + \int_{-1}^{1} 1 \cdot e^{i\omega x} \mathrm{d}x + \int_{1}^{\infty} 0 \cdot e^{i\omega x} \mathrm{d}x$$

$$\mathcal{F}(f(x)) = \int_{-1}^{1} e^{i\omega x} \mathrm{d}x$$

$$\mathcal{F}(f(x)) = \left[\frac{e^{i\omega x}}{i\omega} \right]_{-1}^{1}$$

$$\boxed{\mathcal{F}(f(x)) = \frac{e^{i\omega} - e^{-i\omega}}{i\omega}}$$

4.e

The Fourier series is a representation of a periodic function f with period T in a new discrete basis. The basis is discrete because only functions which have periods T_k which are integer fractions of T are valid basis functions. The Fourier transform extends this to non-periodic (or infinite-period) functions f. A consequence of this is that all frequencies of basis function are valid, so the basis is now continuous.

5 Using Spectrograms for Species ID

5.a

Sound features in the time-domain signal are spread across thousands of pressure samples, obscuring them from discovery and interpretation. In contrast, the spectrogram is a natural representation of the coherent features which differentiate sounds (including bird vocalizations). Spectrograms also naturally lend themselves as inputs to machine powerful learning architectures built for image classification.

5.b

The horizontal axis of a spectrogram is time (seconds) and the vertical axis of a spectrogram is frequency (Hz).

5.c

There is a direct trade-off between temporal resolution and frequency resolution in a spectrogram which is set by the window length. If a bird vocalization contained short-time features and broad frequencies, increasing the temporal resolution to capture the short-time features would degrade the ability to capture the extreme frequencies and vice versa. In this case, higher resolution source (time-domain) data must be obtained to capture the full range of features.

5.d

The fast Fourier transform's bisection algorithm requires data which has length equal to a power of 2.

5.e

The short-time Fourier transform (STFT) is a way to get a time-varying Fourier transform of a signal by breaking the signal into short "windows," performing the Fourier transform on each window, and then joining the results to form a two-dimensional time-frequency output. The temporal resolution is determined by the width of the windows, so shorter windows are required to attain fine temporal resolution. However, the discrete Fourier transform's frequency resolution is determined by the inverse of the width of the window, so wider windows are required to attain fine frequcy resolution.

5.f

The data varies in amplitude. Amplitude is mostly dependent on factors other than the features of the bird vocalization (e.g. distance from microphone, microphone sensitivity). Thus, normalization helps eliminate these unwanted variables.

5.g

Logarithmic scaling makes low-amplitude and high-amplitude features simultaneously interpretable.

5.h

They normalized the data with three unique methods and encoded each into a color channel. This allowed the machine learning model to utilize features that are made apparent using any of the three normalization methods.

5.i

Deviating from logarithmic scaling can have potential for minor improvements in model accuracy but come with a large increase in computational cost.

6 Connections to Your Own Work

6.a

High-frequency surface-mounted pressure sensors are used in wind tunnel tests to detect buffet onset and other aerodynamic features. Classification models could be trained to recognize buffet onset, transition to turbulence, and flow separation based on real-time surface pressure data in the frequency domain.

6.b

I would use a spectrogram. The system is time-varying as the model pitches/yaws in the wind tunnel, so the FFT is inappropriate. Since the data is extremely noisy and high-frequency, a frequency-domain approach would be required to isolate the coherent features.