r/ThingsYouDidntKnow • 1 mo. ago
TheStocksGuy

Zero Frequency Hero: The Silent Pulse of Sound

Sound Wave Properties

1. **Wave Function**:
   - y(t) equals A times sine of 2 times pi times f times t plus phi
   - Where:
     - y(t): displacement at time t
     - A: amplitude
     - f: frequency
     - phi: phase angle
2. **Frequency (f)**:
   - Measured in Hertz (Hz)
   - Example: f equals 440 Hz (A4 note)
3. **Wavelength (lambda)**:
   - Distance between consecutive peaks
   - Formula: lambda equals v divided by f
   - Example: If v equals 343 meters per second and f equals 440 Hz, then lambda is approximately 0.78 meters
4. **Wave Speed (v)**:
   - Speed at which the wave travels, measured in meters per second (m/s)
   - Example: v approximately equals 343 meters per second (speed of sound in air)
5. **Amplitude (A)**:
   - Maximum displacement from equilibrium
   - Example: A equals 1 unit
6. **Period (T)**:
   - Time for one complete cycle, measured in seconds (s)
   - Formula: T equals 1 divided by f
   - Example: For f equals 440 Hz, T is approximately 0.00227 seconds

Relationship Between Properties

v equals f times lambda

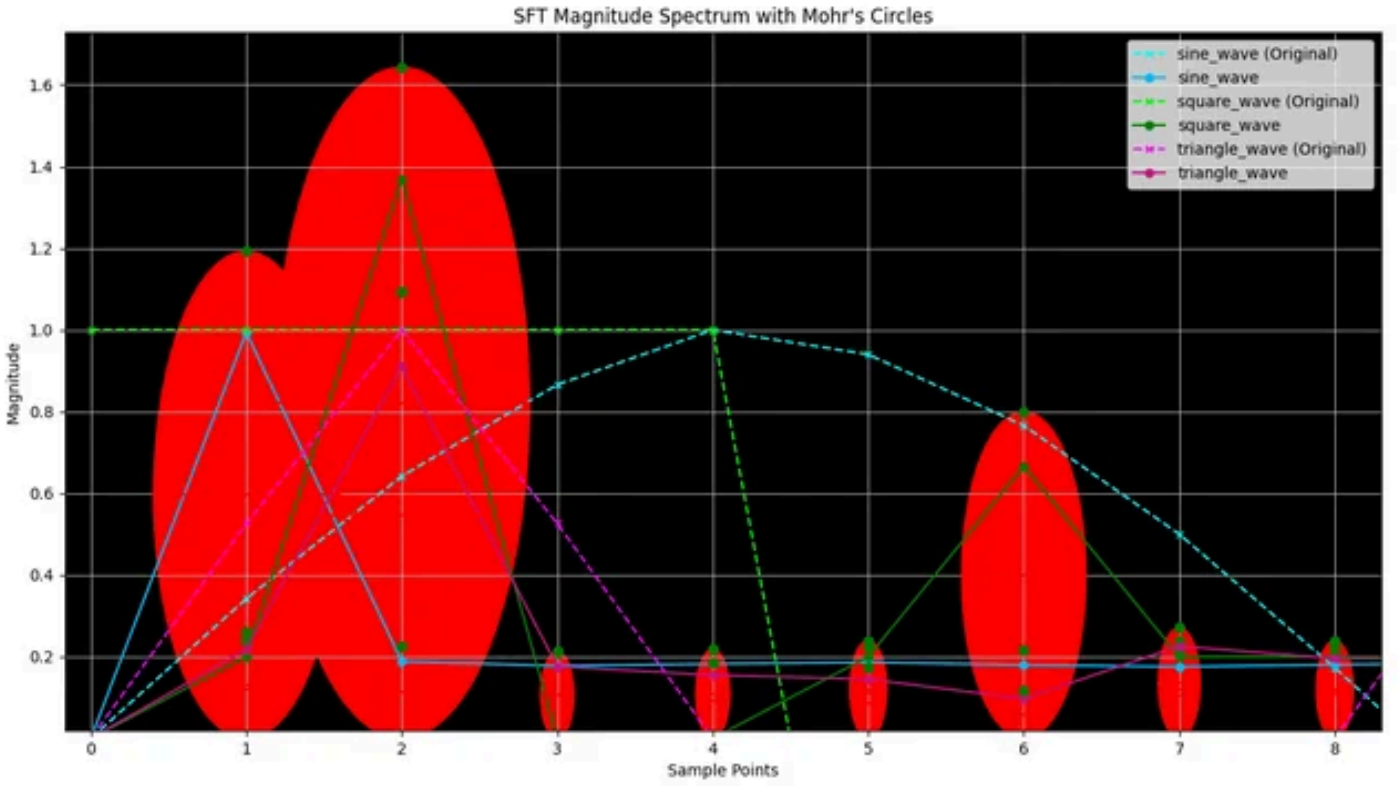
- Wave speed (v) is the product of frequency (f) and wavelength (lambda)

Measuring Sound Wave Properties

2. **Wavelength**:
    - Calculated using the wave speed and frequency
    - lambda equals v divided by f
3. **Wave Speed**:
    - Measured using time-of-flight methods or based on the medium's properties
4. **Amplitude**:
    - Measured using microphones or pressure sensors, often reported in decibels (dB)
5. **Period**:
    - Calculated as the inverse of frequency
    - T equals 1 divided by f

Potential for Further Study

By studying these properties in more detail, especially in different mediums or under varying conditions (such as temperature or humidity), we can gain deeper insights into sound wave behavior. Advanced techniques like Fourier analysis can help break down complex sounds into their constituent frequencies, revealing more about their structure and properties.




```
import cmath
import matplotlib.pyplot as plt

# Example placeholder for well-known test data
```

```
    'triangle_wave': [0.0, 0.526, 1.0, 0.526, 0.0, -0.526, -1.0, -0.526, 0.0, 0.526, 1.0, 0.526, 0
}

def adjust_magnitude(magnitude):
    if magnitude > 3:
        return 3 - (magnitude - 3) * 0.20  # Adjust by 0.20 if over 3
    else:
        return magnitude * 1.20  # Adjust other values directly

def sft(x):
    N = len(x)
    X = [0] * N

    # Separate handling for the zero frequency component (DC component)
    DC_component = sum(x) / N  # Averaging out the input signal

    for k in range(1, N):  # Start from 1 to leave out the zero frequency
        for n in range(N):
            angle = 2j * cmath.pi * k * n / N
            X[k] += x[n] * cmath.exp(-angle)
        # Averaging over N/2 for non-zero frequencies
        X[k] /= N/2

        # Adjust positive and negative values
        if X[k].real < 0 or X[k].imag < 0:
            X[k] = X[k] + 0.20
        else:
            X[k] = X[k] * 1.20

        # Ensure values do not exceed 3
        if X[k].real > 3 or X[k].imag > 3:
            mirrored_value = 3 - ((X[k].real if X[k].real > 3 else X[k].imag) - 3)
            if X[k].real > 3:
                X[k] = cmath.rect(mirrored_value, cmath.phase(X[k]))
            if X[k].imag > 3:
                X[k] = X[k].real + mirrored_value * 1j

    # Set the zero frequency component separately and ensure it's positive
    X[0] = abs(DC_component) * 1.20

    return X

def process_signals(stored_signals):
    measurements = {}

    for name, signal in stored_signals.items():
```

```
        # Store measurements
        measurements[name] = {
            'original_signal': signal,
            'sft_magnitudes': [abs(x) for x in X],
        }

    return measurements

def visualize_sft_measurements(measurements):
    fig, ax = plt.subplots(figsize=(15, 8))
    plt.title("SFT Magnitude Spectrum with Mohr's Circles")
    plt.xlabel("Sample Points")
    plt.ylabel("Magnitude")
    plt.ylim(0, 3)
    plt.grid(True)
    ax.set_facecolor('black')

    # Neon and darker color pairs
    colors = {
        'sine_wave': ('cyan', 'deepskyblue'),
        'square_wave': ('lime', 'green'),
        'triangle_wave': ('magenta', 'mediumvioletred')
    }

    for name, data in measurements.items():
        frequencies = range(len(data['sft_magnitudes']))
        magnitudes = data['sft_magnitudes']
        original_signal = data['original_signal']

        # Plot the original signal (neon colors)
        ax.plot(frequencies, original_signal, color=colors[name][0], linestyle='--', marker='x', m

        # Plot the adjusted magnitudes (darker colors)
        ax.plot(frequencies, magnitudes, color=colors[name][1], marker='o', markersize=5, label=na

        for i, magnitude in enumerate(magnitudes):
            adjusted_magnitude = adjust_magnitude(magnitude)
            plot_mohrs_circle(adjusted_magnitude, ax, frequencies[i], original_signal[i])

            # Add a dot at the adjusted peak height
            plt.plot([frequencies[i]], [adjusted_magnitude], 'go')  # Green dot at the peak height

    plt.legend()
    plt.show()

def plot_mohrs_circle(magnitude, ax, position, original_value):
```

```
    # Define the circle
    circle = plt.Circle((position, sigma_avg), R, color='r', fill=True)

    ax.add_artist(circle)
    ax.plot([position - R, position + R], [sigma_avg, sigma_avg], 'r-')  # Red line at the middle

    # Draw lines within the circle to represent intersections
    if original_value != magnitude:
        intersection_left = (position - R, sigma_avg)
        intersection_right = (position + R, sigma_avg)
        ax.plot([position, intersection_left[0]], [sigma_avg, intersection_left[1]], 'r--')  # Lin
        ax.plot([position, intersection_right[0]], [sigma_avg, intersection_right[1]], 'r--')  # L
    else:
        ax.plot([position], [sigma_avg], 'ro')  # Single dot if there's only one point

# Process the stored signals
measurements = process_signals(stored_signals)

# Visualize the SFT measurements
visualize_sft_measurements(measurements)
```

1 1 Share Approved 1 month ago

Post Insights

Only the post author and moderators can see this

143	100%	1	0
Total Views	Upvote Rate	Comments	Total Shares

Hourly views for first 48 hours

TheStocksGuy OP • 1mo ago

Subject: Improved Methodology for Measuring Sound Waves

I have implemented several significant changes to our methodology, resulting in a cleaner and more visually appealing approach. Unfortunately, Reddit's restrictions on commenting forced me to create new blog entries to share these updates. Editing old posts was not a viable option, as it involved replacing untested updates with similarly untested ones.

Here is the improved version, which has undergone initial testing using graphical representations and specific wave patterns such as square waves, as shown in the examples provided. This method represents one of the most advanced techniques for measuring sound currently available. I strongly encourage its testing, use, and further improvement to ensure it meets the needs of law enforcement and other agencies. Our goal is to help these organizations solve cases more effectively and address instances where scientific understanding of pattern mapping may be lacking.

The green lines in the visualization are crucial for understanding the overall pattern construct. They should be closely examined to comprehend why the behavior occurs in comparison to the adjusted and unadjusted values. This approach provides a detailed mapping of the patterns and their relationships within the same wavelengths and frequencies.

Your feedback and contributions to refining this method would be highly appreciated.

Github: https://github.com/BadNintendo/SFT/tree/main

1 ...

Approved 1 month ago