Fourier Transform

Overview

The FFT (Fast Fourier Transform) module can perform Fourier transform on the input data and return the corresponding frequency amplitude. Through FFT operation, the time domain signal can be converted into the frequency domain signal, which helps to analyze the frequency components of the signal.

What is Fourier Transform?

Fourier transform is a mathematical tool that converts time domain signals into frequency domain signals. In simple terms, it can decompose any periodic signal into a superposition of several sine waves of different frequencies.

Common application areas:

1. Signal Processing

- Audio compression (MP3 and other formats)
- Image compression (JPEG, etc.)
- Noise filtering

1. Communication System

- Modem
- Spectrum Analysis
- Filter Design

1. Physics and Engineering

- Vibration Analysis
- Optical imaging
- Quantum mechanical calculations

1. Medical Imaging

- CT scan
- Magnetic resonance imaging (MRI)
- Ultrasound imaging

1. other

- Seismic wave analysis
- Astronomical data processing
- Financial Market Analysis

Code Sample

```
from machine import FFT
import array
import math
from ulab import numpy as np

"""
傅里叶变换(Fourier Transform)是一种将时域信号分解为不同频率正弦波的叠加的数学方法。
它可以帮助我们分析信号中包含的频率成分。FFT(快速傅里叶变换)是一种高效计算傅里叶变换的算法。

The Fourier Transform is a mathematical method that decomposes a time-domain
signal into
the sum of sinusoidal waves of different frequencies. It helps us analyze the
frequency
components contained in a signal. FFT (Fast Fourier Transform) is an efficient
algorithm
for computing the Fourier transform.
"""

# 定义圆周率常量
# Define PI constant
PI = 3.14159265358979323846264338327950288419716939937510

class SignalProcessor:
    def __init__(self, num_points=64):
        """
        初始化信号处理器
        Initialize signal processor
        """
        self.num_points = num_points
        self.data = []

    def generate_test_signal(self):
        """
        生成测试信号 - 包含5个不同频率的余弦波叠加
        Generate test signal - sum of 5 cosine waves with different frequencies
        """
        for i in range(self.num_points):
            # 生成5个不同频率、不同幅值的余弦波
            # Generate 5 cosine waves with different frequencies and amplitudes
            data0 = 10 * math.cos(2 * PI * i / self.num_points)      # 1倍频
            data1 = 20 * math.cos(2 * 2 * PI * i / self.num_points)  # 2倍频
            data2 = 30 * math.cos(3 * 2 * PI * i / self.num_points)  # 3倍频
            data3 = 0.2 * math.cos(4 * 2 * PI * i / self.num_points) # 4倍频
            data4 = 1000 * math.cos(5 * 2 * PI * i / self.num_points)# 5倍频

            # 将所有波形叠加
            # Sum all waves together
            self.data.append(int(data0 + data1 + data2 + data3 + data4))

        return self.data

    def perform_fft(self, sampling_rate=38400):
        """
        执行FFT变换
        Perform FFT analysis
```

```
        Args:
            sampling_rate: 采样率(Hz) / Sampling rate in Hz
        """
        try:
            # 将列表转换为numpy数组
            # Convert list to numpy array
            data_array = np.array(self.data, dtype=np.uint16)

            # 创建FFT对象并执行变换
            # Create FFT object and perform transform
            fft_obj = FFT(data_array, self.num_points, 0x555)

            # 获取FFT结果
            # Get FFT results
            fft_result = fft_obj.run()

            # 计算幅值谱
            # Calculate amplitude spectrum
            amplitude_spectrum = fft_obj.amplitude(fft_result)

            # 计算频率点
            # Calculate frequency points
            frequency_points = fft_obj.freq(self.num_points, sampling_rate)

            return {
                'fft_result': fft_result,
                'amplitude': amplitude_spectrum,
                'frequencies': frequency_points
            }

        except Exception as e:
            print(f"FFT计算错误 / FFT calculation error: {e}")
            return None

def main():
    # 创建信号处理器实例
    # Create signal processor instance
    processor = SignalProcessor(64)

    # 生成测试信号
    # Generate test signal
    signal = processor.generate_test_signal()
    print("原始信号 / Original signal:", signal)

    # 执行FFT分析
    # Perform FFT analysis
    results = processor.perform_fft()

    if results:
        print("\nFFT结果 / FFT results:", results['fft_result'])
        print("\n幅值谱 / Amplitude spectrum:", results['amplitude'])
        print("\n频率点 / Frequency points:", results['frequencies'])

if __name__ == "__main__":
    main()
```