**使用python进行傅里叶FFT-频谱分析详细教程**

*🔗 原文链接：* [*https://cloud.tencent.com/developer...*](https://cloud.tencent.com/developer/article/1441550)

前言

说明：本文适合信号处理方面有一定的基础的人阅读，能够理解什么时候傅里叶级数和傅里叶变换，能够理解他们的核心思想以及基本原理，能够理解到底什么是“频率域”，能够从频率的角度分析信号。

篇幅较多，阅读本文大约10min。

**一、一些关键概念的引入**

**1、离散傅里叶变换（DFT）**

离散傅里叶变换(discrete Fourier transform) 傅里叶分析方法是信号分析的最基本方法，傅里叶变换是傅里叶分析的核心，通过它把信号从时间域变换到频率域，进而研究信号的频谱结构和变化规律。但是它的致命缺点是：计算量太大，时间复杂度太高，当采样点数太高的时候，计算缓慢，由此出现了DFT的快速实现，即下面的快速傅里叶变换FFT。

**2、快速傅里叶变换（FFT）**

计算量更小的离散傅里叶的一种实现方法。详细细节这里不做描述。

**3、采样频率以及采样定理**

采样频率：采样频率，也称为采样速度或者采样率，定义了每秒从连续信号中提取并组成离散信号的采样个数，它用赫兹（Hz）来表示。采样频率的倒数是采样周期或者叫作采样时间，它是采样之间的时间间隔。通俗的讲采样频率是指计算机每秒钟采集多少个信号样本。

采样定理：所谓采样定理 ，又称香农采样定理，奈奎斯特采样定理，是信息论，特别是通讯与信号处理学科中的一个重要基本结论。采样定理指出，如果信号是带限的，并且采样频率高于信号带宽的两倍，那么，原来的连续信号可以从采样样本中完全重建出来。

定理的具体表述为：在进行模拟/数字信号的转换过程中，当采样频率fs大于信号中最高频率fmax的2倍时,即

fs>2\*fmax

采样之后的数字信号完整地保留了原始信号中的信息，一般实际应用中保证采样频率为信号最高频率的2.56～4倍；采样定理又称奈奎斯特定理。

**4、如何理解采样定理？**

在对连续信号进行离散化的过程中，难免会损失很多信息，就拿一个简单地正弦波而言，如果我1秒内就选择一个点，很显然，损失的信号太多了，光着一个点我根本不知道这个正弦信号到底是什么样子的，自然也没有办法根据这一个采样点进行正弦波的还原，很明显，我采样的点越密集，那越接近原来的正弦波原始的样子，自然损失的信息越少，越方便还原正弦波。故而

采样定理说明采样频率与信号频率之间的关系，是连续信号离散化的基本依据。 它为采样率建立了一个足够的条件，该采样率允许离散采样序列从有限带宽的连续时间信号中捕获所有信息。

**二、使用scipy包实现快速傅里叶变换**

本节不会说明FFT的底层实现，只介绍scipy中fft的函数接口以及使用的一些细节。

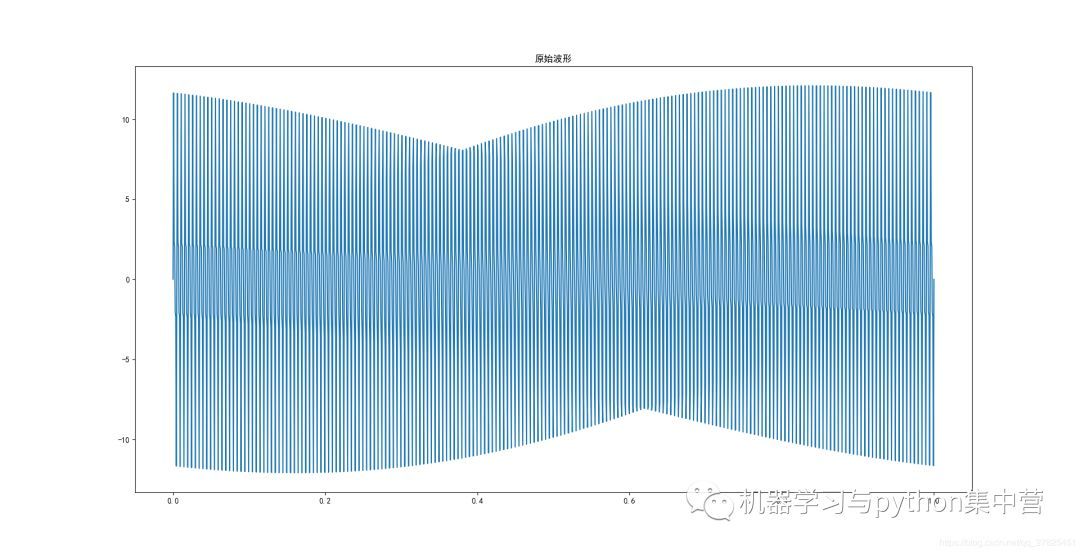
**1、产生原始信号——原始信号是三个正弦波的叠加**

|  |
| --- |
| JavaScript import numpy as np from scipy.fftpack import fft,ifft import matplotlib.pyplot as plt from matplotlib.pylab import mpl   mpl.rcParams['font.sans-serif'] = ['SimHei'] #显示中文 mpl.rcParams['axes.unicode\_minus']=False #显示负号     #采样点选择1400个，因为设置的信号频率分量最高为600赫兹，根据采样定理知采样频率要大于信号频率2倍，所以这里设置采样频率为1400赫兹（即一秒内有1400个采样点，一样意思的） x=np.linspace(0,1,1400)    #设置需要采样的信号，频率分量有200，400和600 y=7\*np.sin(2\*np.pi\*200\*x) + 5\*np.sin(2\*np.pi\*400\*x)+3\*np.sin(2\*np.pi\*600\*x) |

这里原始信号的三个正弦波的频率分别为，200Hz、400Hz、600Hz,最大频率为600赫兹。根据采样定理，fs至少是600赫兹的2倍，这里选择1400赫兹，即在一秒内选择1400个点。

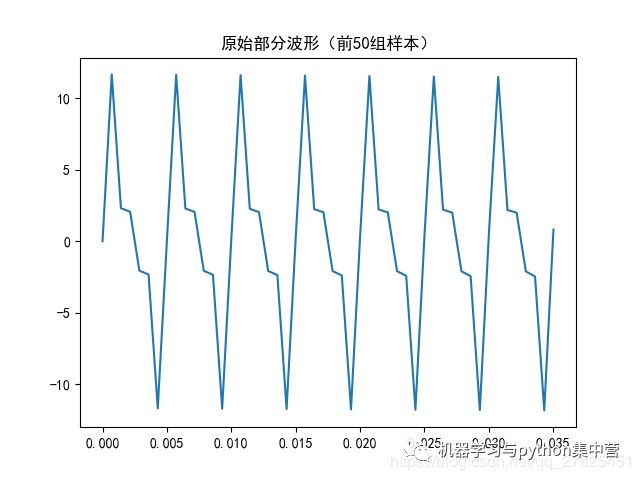
原始的函数图像如下：

|  |
| --- |
| JavaScript plt.figure() plt.plot(x,y)  plt.title('原始波形')   plt.figure() plt.plot(x[0:50],y[0:50])  plt.title('原始部分波形（前50组样本）') plt.show() |



;

由图可见，由于采样点太过密集，看起来不好看，我们只显示前面的50组数据，如下：



;

**2、快速傅里叶变换**

其实scipy和numpy一样，实现FFT非常简单，仅仅是一句话而已，函数接口如下：

from scipy.fftpack import fft,ifft

from numpy import fft,ifft

其中fft表示快速傅里叶变换，ifft表示其逆变换。具体实现如下：

|  |
| --- |
| JavaScript fft\_y=fft(y) #快速傅里叶变换 print(len(fft\_y)) print(fft\_y[0:5]) '''运行结果如下： 1400 [-4.18864943e-12+0.j 9.66210986e-05-0.04305756j 3.86508070e-04-0.08611996j   8.69732036e-04-0.12919206j 1.54641157e-03-0.17227871j] ''' |

我们发 **现以下几个特点：**

（1）变换之后的结果数据长度和原始采样信号是一样的

（2）每一个变换之后的值是一个复数，为a+bj的形式，那这个复数是什么意思呢？

我们知道，复数a+bj在坐标系中表示为（a,b），故而复数具有模和角度，我们都知道快速傅里叶变换具有

“振幅谱”“相位谱”，它其实就是通过对快速傅里叶变换得到的复数结果进一步求出来的，

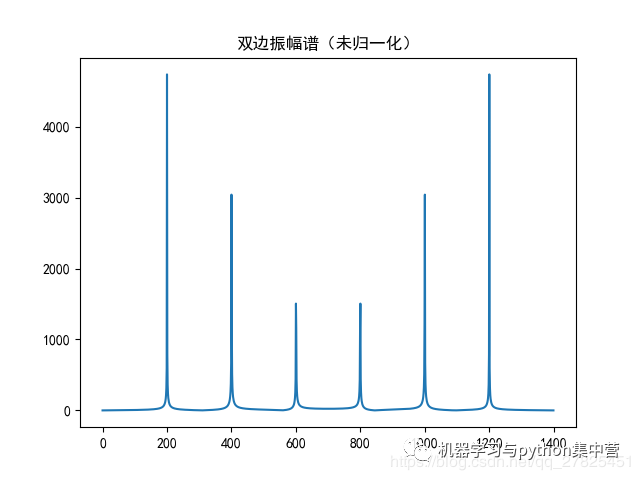
那这个直接变换后的结果是不是就是我需要的，当然是需要的，在FFT中，得到的结果是复数，

（3）FFT得到的复数的模（即绝对值）就是对应的“振幅谱”，复数所对应的角度，就是所对应的“相位谱”，现在可以画图了。

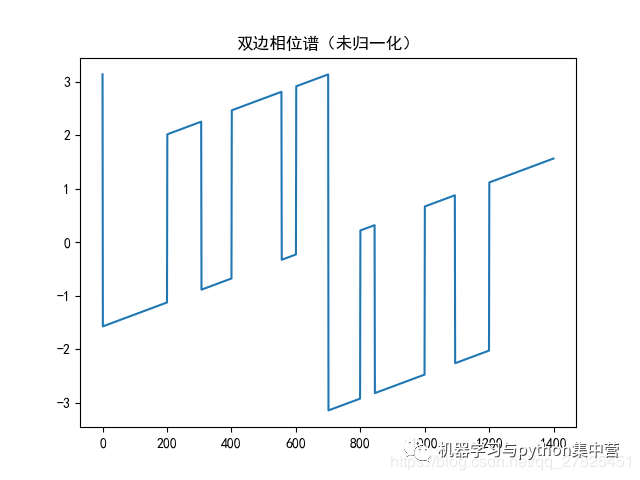
**3、FFT的原始频谱**

|  |
| --- |
| JavaScript N=1400 x = np.arange(N) # 频率个数   abs\_y=np.abs(fft\_y) # 取复数的绝对值，即复数的模(双边频谱) angle\_y=np.angle(fft\_y) #取复数的角度   plt.figure() plt.plot(x,abs\_y)  plt.title('双边振幅谱（未归一化）')   plt.figure() plt.plot(x,angle\_y)  plt.title('双边相位谱（未归一化）') plt.show() |

显示结果如下：



;



;

**注意：** 我们在此处仅仅考虑“振幅谱”，不再考虑相位谱。

我们发现，振幅谱的纵坐标很大，而且具有对称性，这是怎么一回事呢？

**关键：** 关于振幅值很大的解释以及解决办法——归一化和取一半处理

比如有一个信号如下：

Y=A1+A2\*cos(2πω2+φ2）+A3\*cos(2πω3+φ3）+A4\*cos(2πω4+φ4）

经过FFT之后，得到的“振幅图”中，

第一个峰值（频率位置）的模是A1的N倍，N为采样点，本例中为N=1400，此例中没有，因为信号没有常数项A1

第二个峰值（频率位置）的模是A2的N/2倍，N为采样点，

第三个峰值（频率位置）的模是A3的N/2倍，N为采样点，

第四个峰值（频率位置）的模是A4的N/2倍，N为采样点，

依次下去......

考虑到数量级较大，一般进行归一化处理，既然第一个峰值是A1的N倍，那么将每一个振幅值都除以N即可

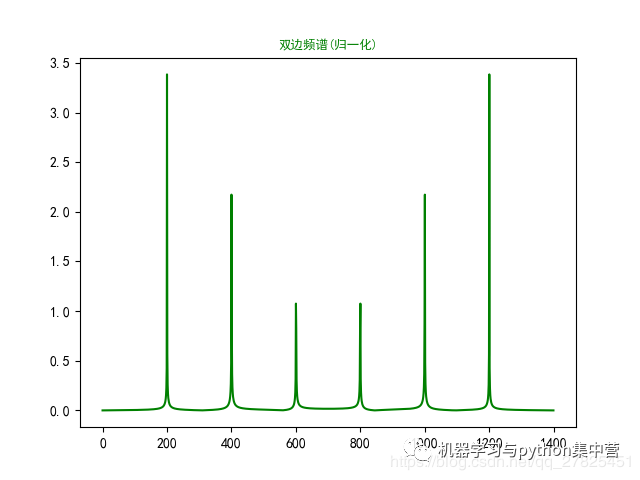
FFT具有对称性，一般只需要用N的一半，前半部分即可。

**4、将振幅谱进行归一化和取半处理**

先进行归一化

|  |
| --- |
| JavaScript normalization\_y=abs\_y/N #归一化处理（双边频谱） plt.figure() plt.plot(x,normalization\_y,'g') plt.title('双边频谱(归一化)',fontsize=9,color='green') plt.show() |

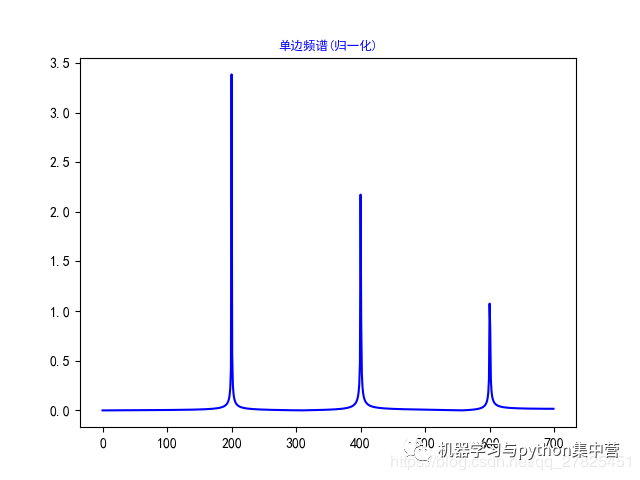
结果为：



;

现在我们发现，振幅谱的数量级不大了，变得合理了，接下来进行取半处理：

|  |
| --- |
| JavaScript half\_x = x[range(int(N/2))] #取一半区间 normalization\_half\_y = normalization\_y[range(int(N/2))] #由于对称性，只取一半区间（单边频谱） plt.figure() plt.plot(half\_x,normalization\_half\_y,'b') plt.title('单边频谱(归一化)',fontsize=9,color='blue') plt.show() |



;

这就是我们最终的结果，需要的“振幅谱”。

**三、完整代码**

|  |
| --- |
| JavaScript import numpy as np from scipy.fftpack import fft,ifft import matplotlib.pyplot as plt from matplotlib.pylab import mpl   mpl.rcParams['font.sans-serif'] = ['SimHei'] #显示中文 mpl.rcParams['axes.unicode\_minus']=False #显示负号   #采样点选择1400个，因为设置的信号频率分量最高为600赫兹，根据采样定理知采样频率要大于信号频率2倍，所以这里设置采样频率为1400赫兹（即一秒内有1400个采样点，一样意思的） x=np.linspace(0,1,1400)    #设置需要采样的信号，频率分量有200，400和600 y=7\*np.sin(2\*np.pi\*200\*x) + 5\*np.sin(2\*np.pi\*400\*x)+3\*np.sin(2\*np.pi\*600\*x)   fft\_y=fft(y) #快速傅里叶变换   N=1400 x = np.arange(N) # 频率个数 half\_x = x[range(int(N/2))] #取一半区间   abs\_y=np.abs(fft\_y) # 取复数的绝对值，即复数的模(双边频谱) angle\_y=np.angle(fft\_y) #取复数的角度 normalization\_y=abs\_y/N #归一化处理（双边频谱）  normalization\_half\_y = normalization\_y[range(int(N/2))] #由于对称性，只取一半区间（单边频谱）   plt.subplot(231) plt.plot(x,y)  plt.title('原始波形')   plt.subplot(232) plt.plot(x,fft\_y,'black') plt.title('双边振幅谱(未求振幅绝对值)',fontsize=9,color='black')    plt.subplot(233) plt.plot(x,abs\_y,'r') plt.title('双边振幅谱(未归一化)',fontsize=9,color='red')    plt.subplot(234) plt.plot(x,angle\_y,'violet') plt.title('双边相位谱(未归一化)',fontsize=9,color='violet')   plt.subplot(235) plt.plot(x,normalization\_y,'g') plt.title('双边振幅谱(归一化)',fontsize=9,color='green')   plt.subplot(236) plt.plot(half\_x,normalization\_half\_y,'blue') plt.title('单边振幅谱(归一化)',fontsize=9,color='blue')   plt.show() |

