

Lab4_Frequency analysis (of periodic signals)

频率分析（周期性信号的分析）---傅里叶分析

傅里叶定理指出，任何频率为 f_0 的周期性信号的任何周期性信号都可以通过将频率为 $f_0, 2f_0, 3f_0$ 的 "正弦波"(sine waves)相加来精确构建。等等。将周期性时域信号分割成正弦波称为傅里叶分析。

这个 "傅里叶系列" 中的每个正弦波都有以下特点

- 频率;
- 振幅;
- 相位;
-

f_0 被称为基本频率。

$2f_0, 3f_0, 4f_0, \text{etc.}$ 被称为谐波

The sawtooth signal（锯齿状信号）

锯齿信号的最简单形式定义为

$$X_{\text{saw}}(t) = t - [t],$$

或以0为中心作为

$$x_{\text{saw}}(t) = 2(t - [t]) - 1.$$

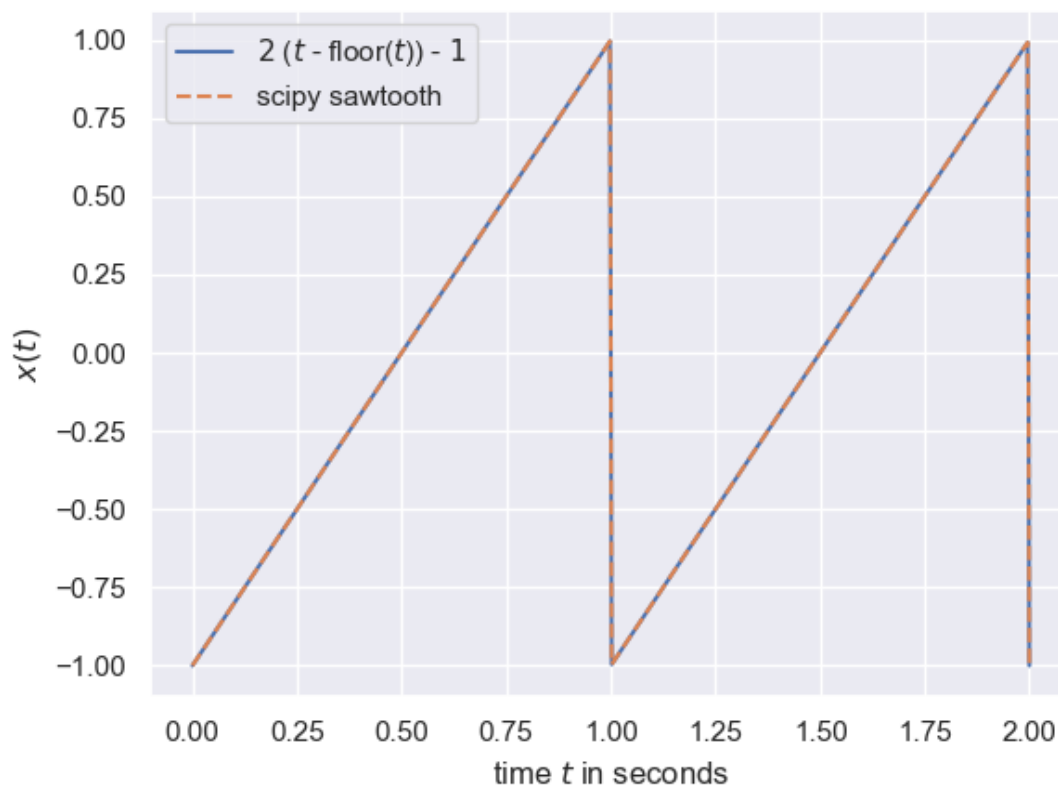
scipy库给了我们更多的灵活性来生成和可视化锯齿信号，因为它们可以像正弦信号一样被生成。

```
from scipy import signal

t = np.linspace(0, 2, 500)
saw_tooth = 2*(t-np.floor(t))-1

f0 = 1 # frequency in Hz for scipy sawtooth
saw_tooth2 = signal.sawtooth(2 * np.pi * f0 * t)

plt.plot(t, saw_tooth, label='$2\$ ($t\$ - \text{floor}(\$t\$)) - \$1\$')
plt.plot(t, saw_tooth2, '--', label='scipy sawtooth');
plt.xlabel('time $t$ in seconds'); plt.ylabel('$x(t)$')
plt.legend();
```



我们在上面的图中看到，使用方程（2）和使用scipy库实现锯齿的结果是一样的（尽管scipy的方法证明了更多的灵活性。

- **任务1:** 生成一个长度为2秒的锯齿信号，基频为200Hz的锯齿信号，并播放它。为了实现这一目标，请替换下面代码中由# ...'表示的注释。该代码将显示时域信号的初始20毫秒，以及使用函数`np.fft.rfft()`计算实值时间序列的（复数）离散傅里叶变换（DFT）的频谱（频域）。

```
from scipy import signal          # for easy sawtooth signal generation
from IPython import display as ipd # to playback audio signals

fs= 8000                          # sampling frequency
t = np.arange(0,2,1/fs)          # time vector

f0 = 200                          # frequency in 200 Hz for scipy sawtooth
saw_tooth = signal.sawtooth(2*np.pi*f0*t)

# plot first 20 ms (=160 samples at sampling frequency of 8000 Hz)
plt.subplot(1,2,1)
plt.plot(t[0:160], saw_tooth[0:160], '--', label='scipy sawtooth');
plt.xlabel('time $t$ in seconds'); plt.ylabel('$x(t)$')
plt.legend();

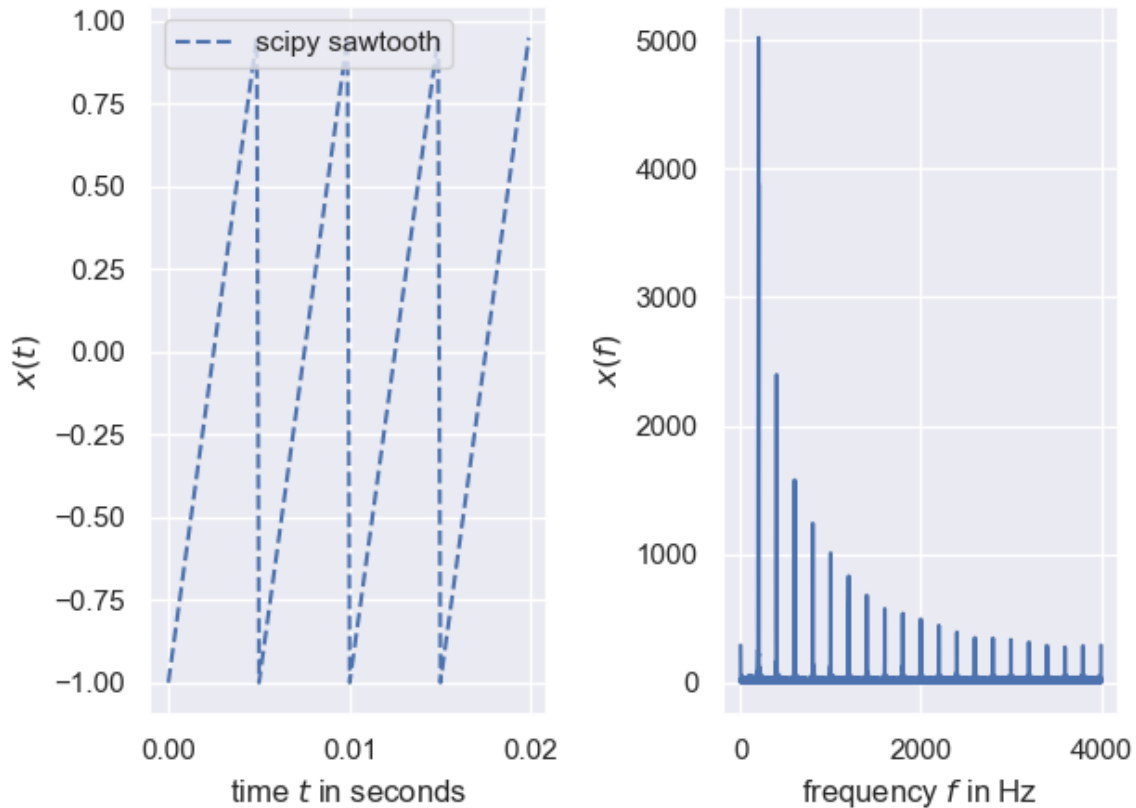
# calculate the spectrum (frequency domain representation)
FFT_length = 2**15 # take a power of two which is larger than the signal length
f = np.linspace(0, fs/2, num=int(FFT_length/2+1))
spectrum = np.abs(np.fft.rfft(saw_tooth,n=FFT_length))

# plot the spectrum
plt.subplot(1,2,2)
```

```
plt.plot(f,spectrum)
plt.xlabel('frequency $f$ in Hz');plt.ylabel('$x(f)$')

plt.tight_layout() # this allows for some space for the title text.

# playback sound file (if you want)
ipd.Audio(saw_tooth, rate=fs)
```



当我们看上图右边的光谱时，我们看到它是由等距离的谱线组成的，在较高的频率下振幅逐渐减小。

Fourier series of Saw-tooth signal (Task 2)

锯齿信号的傅里叶级数由以下公式给出

$$x'_{\text{saw}}(t) = \frac{2}{\pi} \left[\sin(\omega_0 t) + \frac{\sin(2\omega_0 t)}{2} + \frac{\sin(3\omega_0 t)}{3} + \dots \right],$$

$$= \frac{2}{\pi} \sum_{i=1}^{\infty} \frac{\sin(i\omega_0 t)}{i},$$

with $\omega = 2\pi f$ 是角频率。ie.

$$\omega_0 = 2\pi f_0$$

下面的代码从（左图）开始生成前四个正弦信号，以及这些信号的叠加，从而一步步形成锯齿信号（右图）。

```

fs = 8000 # Sampling frequency

t = np.arange(0,2,1/fs) # time vector 2

f0 = 2          # fundamental frequency in Hz


sin1 = np.sin(2*np.pi*f0*t)

sin2 = np.sin(2*np.pi*2*f0*t)/2

sin3 = np.sin(2*np.pi*3*f0*t)/3

sin4 = np.sin(2*np.pi*4*f0*t)/4


plt.figure(figsize=(8,6))

plt.subplot(4,2,1)

plt.plot(t,sin1,label='$\mathrm{sin}(\omega_0 t)$')

plt.ylabel('$x_1(t)$')

plt.legend()


plt.subplot(4,2,3)

plt.plot(t,sin2,label='$\mathrm{sin}(2 \omega_0 t)/2$')

plt.ylabel('$x_2(t)$')

plt.legend()


plt.subplot(4,2,4)

plt.plot(t,sin1+sin2,label='$x_1(t)+x_2(t)$')

plt.legend()


plt.subplot(4,2,5)

plt.plot(t,sin3,label='$\mathrm{sin}(3 \omega_0 t)/3$')

plt.ylabel('$x_3(t)$')

plt.legend()

```

```
plt.subplot(4,2,6)

plt.plot(t,sin1+sin2+sin3,label='$x_1(t)+x_2(t)+x_3(t)$')

plt.legend()


plt.subplot(4,2,7)

plt.plot(t,sin4,label='$\mathrm{sin}(4 \omega_0 t)/4$')

plt.ylabel('$x_4(t)$')

plt.xlabel('time $t$ in seconds')

plt.legend()


plt.subplot(4,2,8)

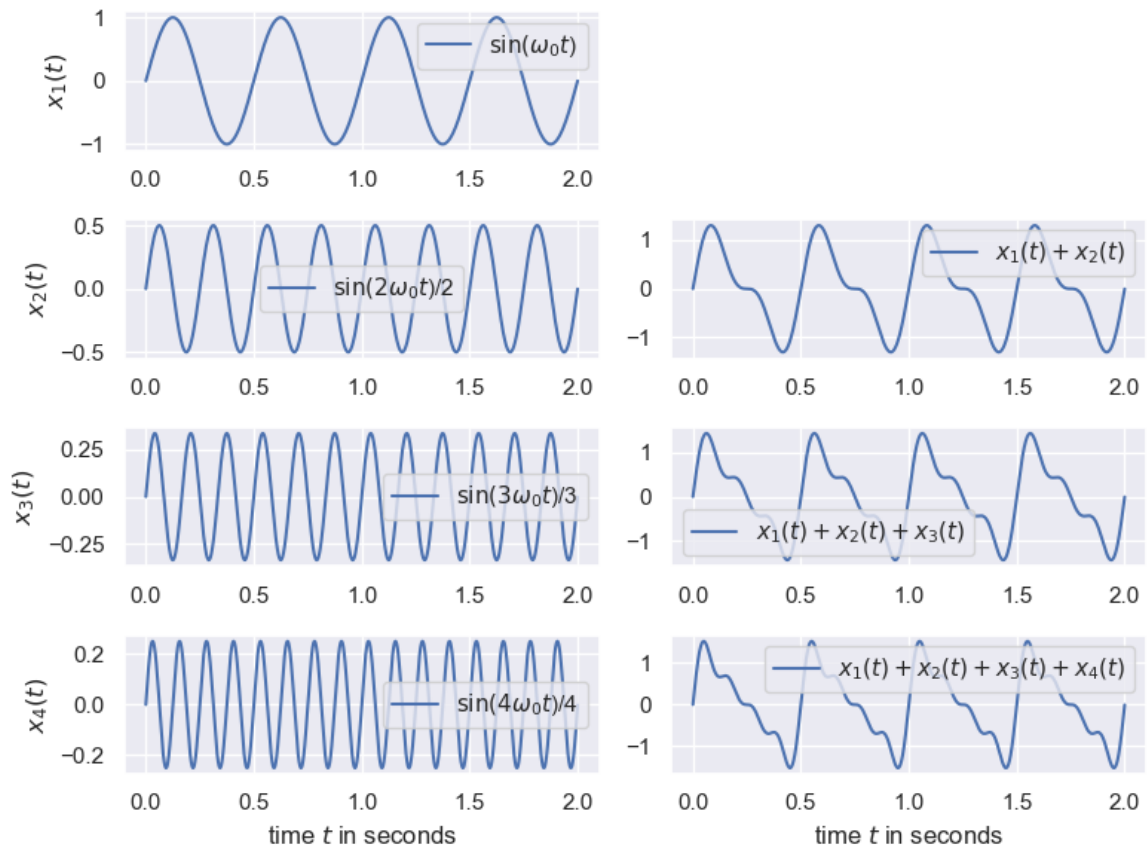
plt.plot(t,sin1+sin2+sin3+sin4,label='$x_1(t)+x_2(t)+x_3(t)+x_4(t)$')

plt.xlabel('time $t$ in seconds')

plt.legend()


plt.tight_layout()

plt.show()
```



请注意，左侧面板中的正弦波的振幅在下降（见y轴标签）。

作为一个函数的实现（下面的generateSawTooth()）当然在代码的可重用性方面更有效率。

```
def generateSawTooth(f0=2, length = 2, fs=8000, order=10, height=1):

    """

    Return a saw-tooth signal with given parameters.

    Parameters
    \-----

    f0 : float, optional

    • fundamental frequency $f_0$ of the signal to be generated,

    • default: 1 Hz

    length : float, optional

    • length of the signal to be generated, default: 2 sec.

    fs : float, optional

    • sampling frequency $f_s$, default: 8000 Hz

    order : int, optional
```

- number of sinusoids to approximate saw-tooth, default: 10

height : float, optional

- height of saw-tooth, default: 1

Returns

\-----

sawTooth

- generated sawtooth signal

t

- matching time vector

"""

-

```
t=np.arange(0,length,1/fs) # time vector
```

```
sum = np.zeros(len(t))
```

```
for ii in range(order):
```

- jj=ii+1

- sum += np.sin(2*np.pi*jj*f0*t)/jj

```
return 2*height*sum/np.pi, t
```

```
saw,t = generateSawTooth(order=100)
```

```
plt.plot(t,saw)
```

```
plt.xlabel('time $t$ in seconds')
```

```
plt.show()
```

