

中山大学

计算机学院

《信号与系统》上机实验辅导 (Python 版)

适用专业：

计算机科学与技术、信息与计算科学、软件工程、
网络空间安全、保密管理

《信号与系统》课程组编

2023 年 2 月

说明

本资料主要用于帮助学生熟悉基于 **Python** 的《信号与系统》相关仿真实验中常用的函数与方法，辅导学生自主上机实验。可作为自学材料或基础实验的参考资料。

Python 环境搭建指南

0. 前言

Python 语言虽然简单易用，但是其运行环境如果配置错误或者不小心被污染，使用时就会出现千奇百怪的报错。不仅 Python 语言本身有 2.7, 3.0~3.9 等非常多的版本，而且 Python 社区中也有很多功能强大的包（Package）。Anaconda 能在一台计算机上创建多个 Python 虚拟环境，这些虚拟环境以文件夹的形式隔离开来互不影响。比如我们可以借助 Anaconda 创建一个 Python 2.7 版本的虚拟环境和一个 Python 3.6 版本的虚拟环境。Anaconda 强大的包管理功能很方便，比如安装、更新、卸载包。

1. Anaconda 概述

Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux, Mac, Windows 系统，提供了包管理与环境管理的功能，可以很方便地解决多版本 Python 并存、切换以及各种第三方包安装问题。Anaconda 利用 conda 命令来进行包和环境 (environment) 的管理，并且已经包含了 Python 和相关的配套工具，这意味着我们无需去 Python 官网下载 Python 解释器的安装包。

2. Anaconda 的安装

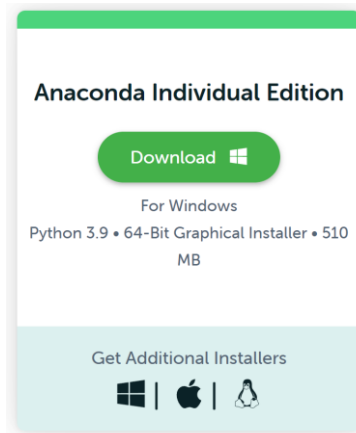
2.1 相关网址

Anaconda 官网: <https://www.anaconda.com/>

Anaconda 个人版安装包: <https://www.anaconda.com/products/individual>

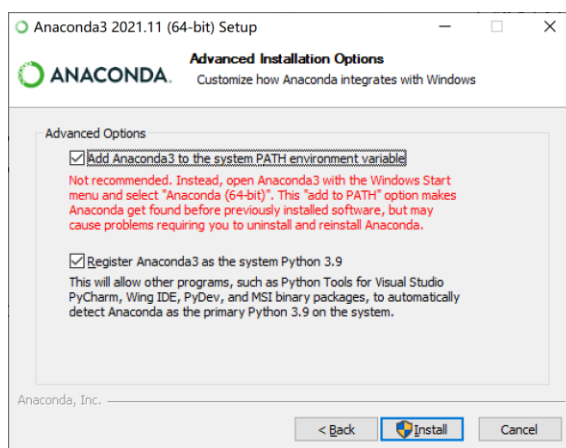
2.2 安装

首先选择自己需要的操作系统对应的安装包，本指南选择的是 Windows 64 位系统，Anaconda 内部支持的 Python 最高版本为 3.9。



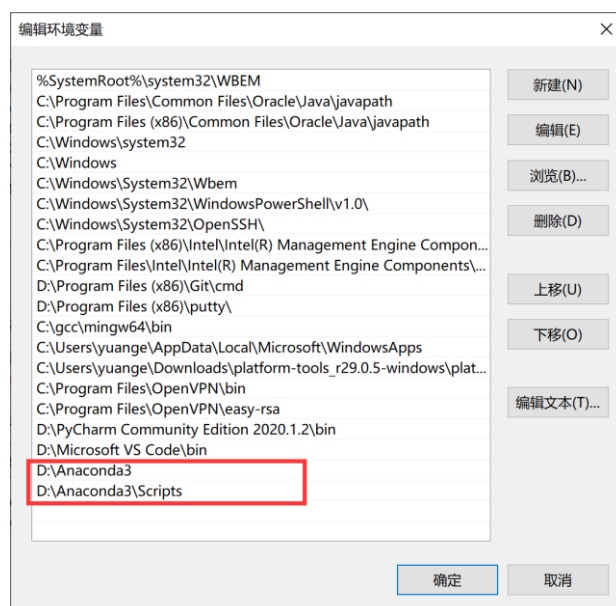
下载完毕安装包后即可开始安装。安装过程中会出现选择是否把 Anaconda 的安装目录加入到 Windows 系统的 PATH 环境变量，这里可以选择打勾。（安装界面提示 **Not recommended**，是因为可能导致 Anaconda 的卸载不干净。如果

不打勾的话，需要在安装完毕后手动添加环境变量。)



如果上一步打勾，一路点下去即可安装完毕，Anaconda 的安装到此结束。

*如果上一步未打勾，还需要手动添加环境变量，以便我们在命令行中调用 conda 命令：



2.3 验证安装

在控制台中输入 `conda -V` 查看 Anaconda 的版本，显示版本号就说明安装成功：

```
C:\Users\yuange>conda -V
conda 4.10.3
```

2.4 配置镜像源

使用 `conda install xxx` 命令可以轻松快捷安装各种 Python 包。但是由于这些包的镜像地址通常都是国外的服务器，下载速度很慢，所以我们可以配置清华的国内镜像源，享受高速的下载速度。清华镜像源的官方配置教程地址是 <https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/>。

Anaconda 镜像使用帮助

Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux, Mac, Windows, 包含了众多流行的科学计算、数据分析的 Python 包。

Anaconda 安装包可以到 <https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/> 下载。

TUNA 还提供了 Anaconda 仓库与第三方源 (conda-forge、msys2、pytorch等, [查看完整列表](#)) 的镜像, 各系统都可以通过修改用户目录下的 `.condarc` 文件。Windows 用户无法直接创建名为 `.condarc` 的文件, 可先执行 `conda config --set show_channel_urls yes` 生成该文件之后再修改。

注: 由于更新过快难以同步, 我们不同步 `pytorch-nightly`, `pytorch-nightly-cpu`, `ignite-nightly` 这三个包。

```
channels:
  - defaults
show_channel_urls: true
default_channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2
custom_channels:
  conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  msys2: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  bioconda: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  menpo: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  pytorch-lts: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  simpleitk: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
```

即可添加 Anaconda Python 免费仓库。

运行 `conda clean -i` 清除索引缓存, 保证用的是镜像站提供的索引。

运行 `conda create -n myenv numpy` 测试一下吧。

Miniconda 镜像使用帮助

Miniconda 是一个 Anaconda 的轻量级替代, 默认只包含了 python 和 conda, 但是可以通过 pip 和 conda 来安装所需要的包。

Miniconda 安装包可以到 <https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/> 下载。

其他三方源

对于conda的其他三方源, 如有需要请修改`anaconda.py`文件, 并提交pull request, 我们会综合考虑多方因素来酌情增减。

注意: 清华源的 `https` 可能过期, 将上图配置文件中的 `https` 都改成 `http` 即可解决。

3. 开始使用 Anaconda

3.1 查看 Python 虚拟环境

打开命令行 (cmd.exe) 后, 使用 `conda info -e` 可显示当前系统上的所有 Python 虚拟环境:

```
C:\Users\yuange>conda info -e
# conda environments:
#
base                                * D:\Anaconda3
```

base 代表默认的基础环境;

星号 (*) 代表当前进入的是哪一个环境。

注意: 第一次使用 `conda` 时还需要调用 `conda init` 初始化环境, 然后重启命令行。

```
C:\Users\yuange>conda init
no change      D:\Anaconda3\Scripts\conda.exe
no change      D:\Anaconda3\Scripts\conda-env.exe
no change      D:\Anaconda3\Scripts\conda-script.py
no change      D:\Anaconda3\Scripts\conda-env-script.py
no change      D:\Anaconda3\condabin\conda.bat
no change      D:\Anaconda3\Library\bin\conda.bat
no change      D:\Anaconda3\condabin\_conda_activate.bat
no change      D:\Anaconda3\condabin\rename_tmp.bat
no change      D:\Anaconda3\condabin\conda_auto_activate.bat
no change      D:\Anaconda3\condabin\conda_hook.bat
no change      D:\Anaconda3\Scripts\activate.bat
no change      D:\Anaconda3\condabin\activate.bat
no change      D:\Anaconda3\condabin\deactivate.bat
no change      D:\Anaconda3\Scripts\activate
no change      D:\Anaconda3\Scripts\deactivate
no change      D:\Anaconda3\etc\profile.d\conda.sh
no change      D:\Anaconda3\etc\fish\conf.d\conda.fish
no change      D:\Anaconda3\shell\condabin\Conda.ps1
no change      D:\Anaconda3\shell\condabin\conda-hook.ps1
no change      D:\Anaconda3\Lib\site-packages\xontrib\conda.xsh
no change      D:\Anaconda3\etc\profile.d\conda.csh
no change      C:\Users\yuange\Documents\WindowsPowerShell\profile.ps1
no change      HKEY_CURRENT_USER\Software\Microsoft\Command Processor\AutoRun
No action taken.
```

3.2 创建一个 Python 虚拟环境

下面创建一个 Python 版本为 3.6.5 的虚拟环境，命名为 py36。conda 命令为：

```
C:\Users\yuange>conda create -n py36 python=3.6.5
```

安装完毕后提示：

```
done
#
# To activate this environment, use
#
#     $ conda activate py36
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

可以使用 `conda activate xxx` 进入到某个 Python 虚拟环境，用 `conda deactivate` 推出当前环境。

进入 py36 环境，命令行开头括号内会显示当前环境的名字：

```
C:\Users\yuange>conda activate py36
(py36) C:\Users\yuange>
```

检查 Python 的版本：

```
(py36) C:\Users\yuange>python -V
Python 3.6.5 :: Anaconda, Inc.
```

至此，Python 3.6.5 虚拟环境创建完毕。

关于更详细的 conda 命令请参考：

<https://blog.csdn.net/menc15/article/details/71477949>

3.3 在虚拟环境中安装包 (Package)

在 Python 虚拟环境中, 我们既可以使用 `conda install xxx` 安装 Python 包也可以直接使用 `pip install xxx` 命令安装, 其中 `pip` 命令实际上调用的是当前虚拟环境内部的 `pip` 工具, 不会影响其他环境。

```
(py36) C:\Users\yuange>conda install numpy
```

```
(py36) C:\Users\yuange>pip install numpy
```

上述两条安装 `numpy` 的命令是等价的。更多关于 `conda` 管理包的命令参考:

<https://blog.csdn.net/menc15/article/details/71477949>

4. 代码编辑器的选择

为了方便地编写 Python 代码, 仅有 Anaconda 是不够的, 我们还需要借助代码编辑器。比较主流的代码编辑器有 VSCode (轻量级代码编辑器) 和 PyCharm (集成式 IDE)。

VSCode 官网: <https://code.visualstudio.com/>

PyCharm 官网: <https://www.jetbrains.com/pycharm/>

VSCode 的特点是软件体积小, 个性化功能多, 许多功能需要用户手动配置。

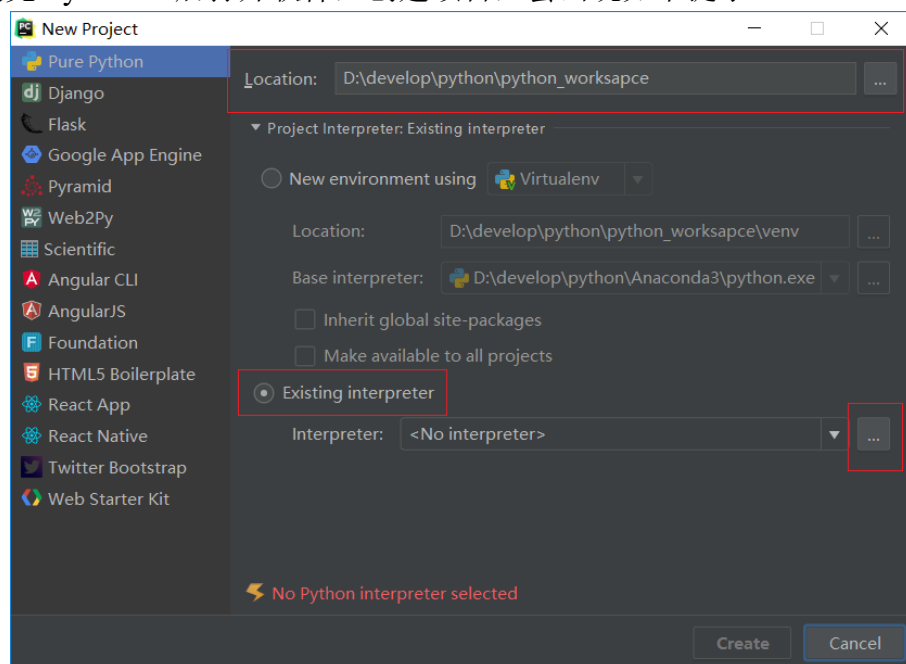
PyCharm 的特点是软件庞大, 包含丰富的功能可直接使用, 比如运行、调试、git 代码管理, 软件内部支持 Anaconda。

4.1 VSCode 的配置

我们可以只把 VSCode 当作代码编辑器而不添加任何个性化功能, 在 Python 代码完成后, 使用命令行运行 Python 即可, 更多复杂的代码高亮个性化、运行、调试等设置请参考: <https://zhuanlan.zhihu.com/p/30324113>

4.2 PyCharm 的配置

安装完 PyCharm 后打开软件, 创建项目, 会出现如下提示:



其中 Existing interpreter 指的是 Python 的解释器对应的路径。因为我们利用 Anaconda 创建了虚拟环境，所有虚拟环境都在 Your_Path_To_Anaconda/envs 目录下，选择该目录下的 python.exe 作为 PyCharm 的解释器即可：



更多关于 PyCharm 的使用请参考：

<https://www.runoob.com/w3cnote/pycharm-windows-install.html>

以及：

https://blog.csdn.net/ITF_001/article/details/114678241

参考资料：

[1] Anaconda 安装教程 (Win10 环境)：

<https://www.cnblogs.com/maxiaodoubao/p/9854595.html>

[2] CondaHTTPError 解决：

<https://www.cnblogs.com/tianlang25/p/12433025.html>

Python 学习网站：

[1] <https://www.liaoxuefeng.com/wiki/1016959663602400>

[2] <https://www.runoob.com/python/python-tutorial.html>

...

实验一 基本信号的产生

1 实验目的

学习使用Python产生基本信号、绘制信号波形、实现信号的基本运算，为信号分析和系统设计奠定基础。

2 实验原理

通过Python的Numpy库和Scipy库可以用于产生基本的信号，如阶跃信号，指数信号，脉冲信号等等。本次实验版本Python=3.6.5, numpy=1.14.5, scipy=1.1.0, matplotlib= 3.1.1。

对于本实验Python库约定：

```
# 导入 需要的 library 库
import numpy as np # 科学计算
import matplotlib.pyplot as plt # 画图
import scipy.signal as sg # 导入 scipy 的 signal 库 命名为 sg

import warnings
warnings.filterwarnings("ignore") # 去掉常规警告
```

2.1 基本信号的产生

2.1.1 连续阶跃信号的产生

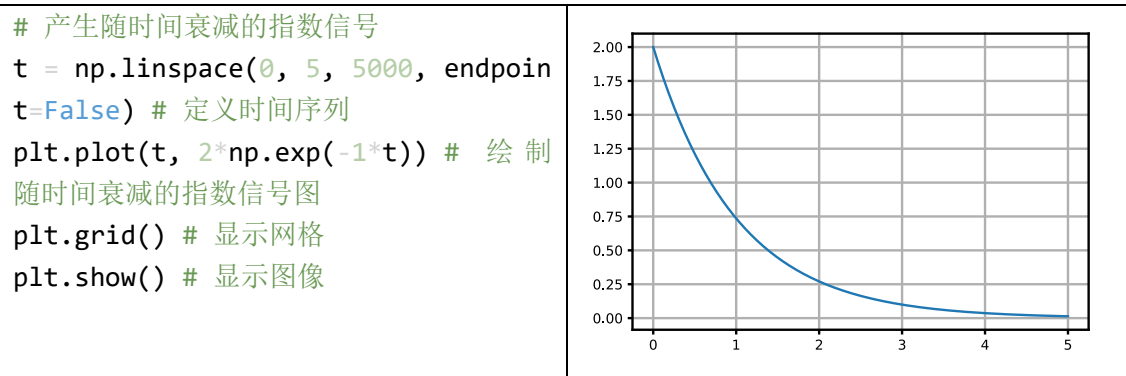
单位阶跃信号生成的方式有很多种，还可以使用 [numpy.heaviside](#) 函数来生成。

Python 程序	图像
<pre># 产生阶跃信号 t = np.linspace(-1, 1, 500, endpoint=False) # 定义时间序列 plt.plot(t, np.array(t > 0, dtype=np.int)) # 绘制阶跃信号图 plt.grid() # 显示网格 _ = plt.ylim(-0.5, 1.5) # 限制 y 轴范围 plt.show() # 显示图像</pre>	

2.1.2 连续指数信号的产生

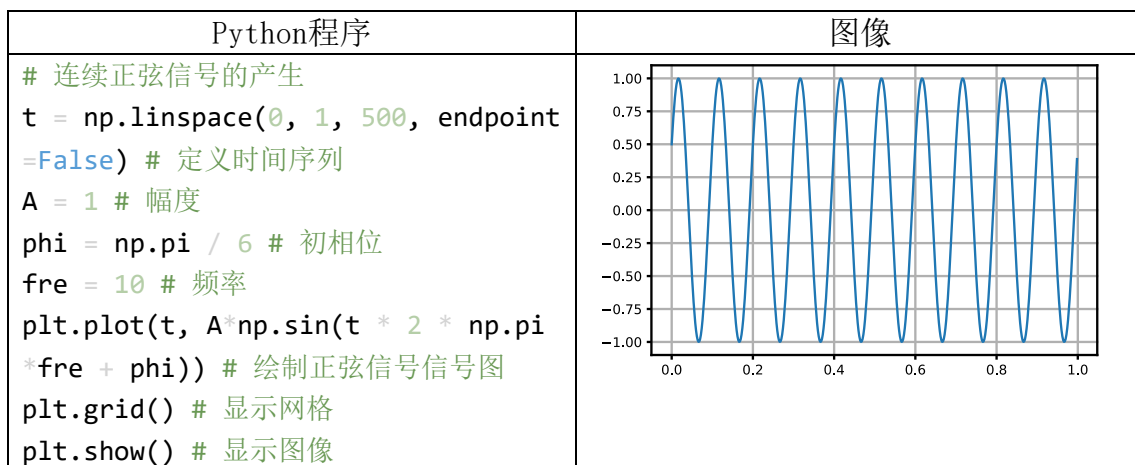
产生随时间衰减的指数信号的 Python 程序：

Python 程序	图像
-----------	----



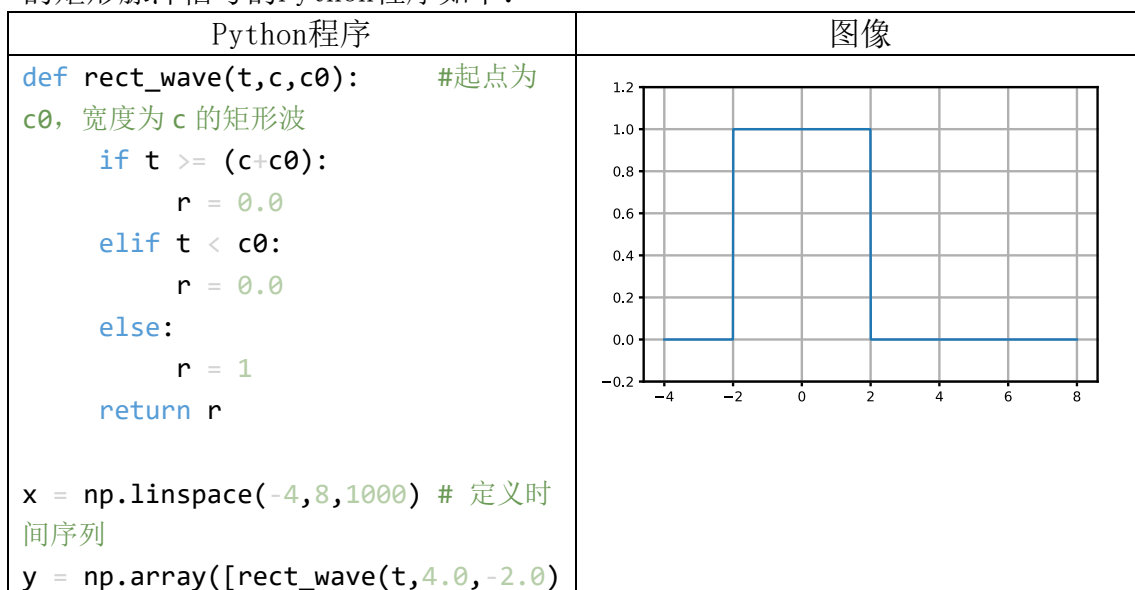
2.1.3 连续正弦信号的产生

利用Numpy提供的函数sin和cos可产生正弦和余弦信号。产生一个幅度为1, 频率为10Hz, 初相位为($\frac{\pi}{6}$)的正弦信号的Python程序如下:



2.1.4 连续矩形脉冲信号

利用Python产生高度为1、宽度为4、延时2秒、关于t=0对称的矩形脉冲信号的Python程序如下:



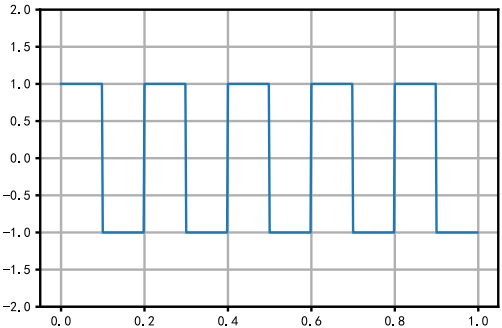
<pre> for t in x]) # 产生矩形波形 _ = plt.ylim(-0.2,1.2) # 限制 y 轴范围 plt.plot(x,y) # 绘制矩形波形 plt.grid() # 显示网格 plt.show() # 显示图像 </pre>	
---	--

2.1.5 连续周期矩形波信号的产生

函数 `scipy.signal.square(w0*t, duty=0.5)` 产生基本频率为 w_0 (周期 $T=2\pi/w_0$) 的周期矩形波信号, 占空比默认为 0.5。占空比 $duty= t/T$, $duty$ 范围 $[0, 1]$ 。

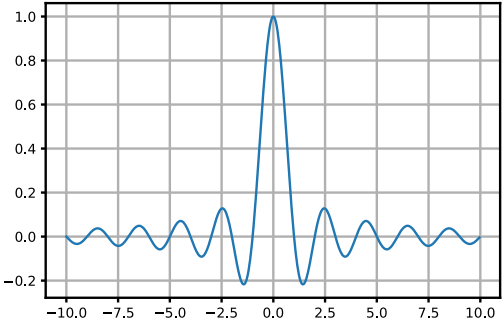
τ 为一个周期中信号为正的时间长度。 $\tau=T/2, duty=0.5, square(w_0*t, 0.5)$ 等同于 `square(w0*t)`。

产生一个幅度为 1, 基频为 5Hz, 占空比为 50% 的周期方波的 Python 程序如下:

Python 程序	图像
<pre> t = np.linspace(0, 1, 500, endpoint=False) # 定义时间序列 plt.plot(t, sg.square(2 * np.pi * 5 * t)) # 绘制周期方波图像 plt.ylim(-2, 2) # 限制 y 轴范围 plt.grid() # 显示网格 plt.show() # 显示图像 </pre>	

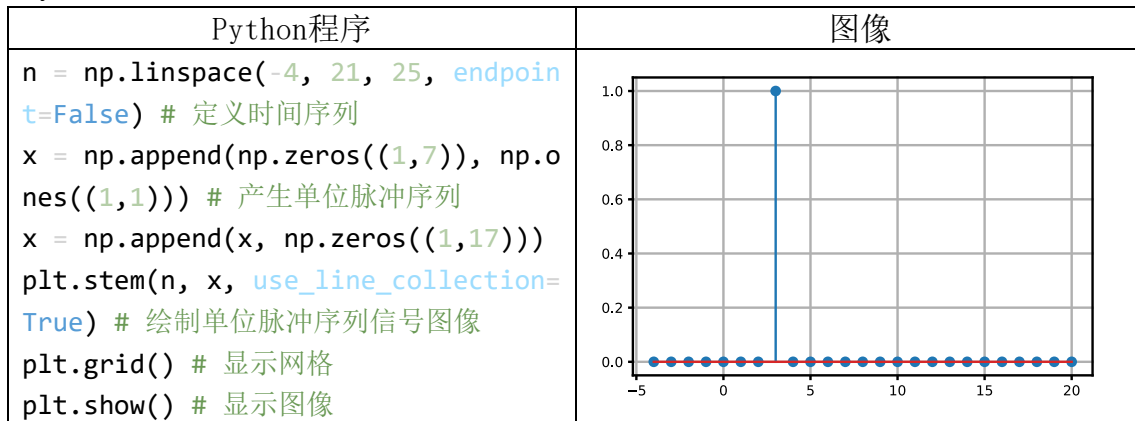
2.1.6 连续抽样信号的产生

可使用函数 `sinc(x)` 计算抽样信号, 函数 `sinc(x)` 的定义为 $\frac{\sin(\pi x)}{\pi x}$ 。产生信号的 Python 程序如下:

Python 程序	图像
<pre> t = np.linspace(-10, 10, 500, endpoint=False) # 定义时间序列 plt.plot(t, np.sinc(t)) # 绘制抽样信号图像 plt.grid() # 显示网格 plt.show() # 显示图像 </pre>	

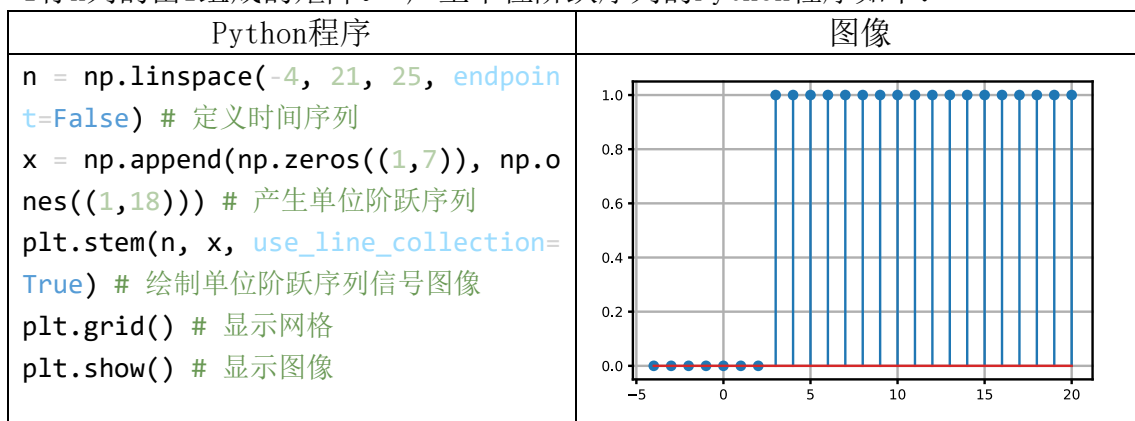
2.1.7 单位脉冲序列的产生

函数 `numpy.zeros((1,n))` 可以生成单位脉冲序列。函数 `numpy.zeros((1,n))` 产生1行n列的由0组成的矩阵。产生单位脉冲序列的Python程序如下：



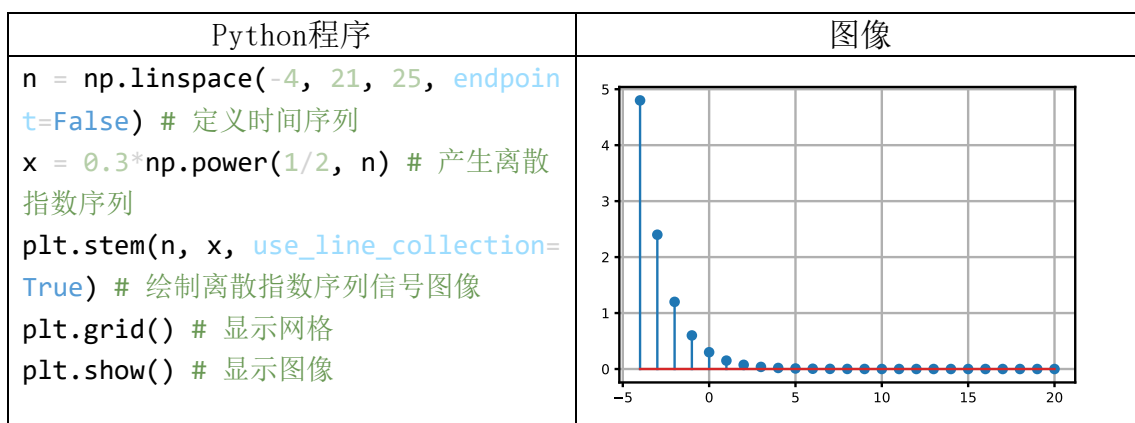
2.1.8 单位阶跃序列的产生

函数 `numpy.ones(1,n)` 可以生成单位阶跃序列。函数 `numpy.ones(1,n)` 产生1行n列的由1组成的矩阵。产生单位阶跃序列的Python程序如下：



2.1.9 指数序列的产生

产生离散指数序列的Python程序如下，离散指数函数为 $x(n) = 0.3 * 0.5^n$ ：



2.1.10 正弦序列的产生

产生正弦序列的 Python 程序如下：

Python程序	图像
<pre>n = np.linspace(-50, 51, 101, endpoint=False) # 定义时间序列 omega = np.pi/10 # 频率 x = 0.3*np.sin(omega*n+ np.pi/5) # 产生正弦序列 plt.stem(n, x, use_line_collection=True) # 绘制正弦序列信号图像 plt.grid() # 显示网格 plt.show() # 显示图像</pre>	

2.1.11 离散周期矩形波序列的产生

产生方波幅度为 1、基频为 rad，占空比为 50%的周期方波的 Python 程序为：

Python程序	图像
<pre>n = np.linspace(-10, 11, 21, endpoint=False) # 定义离散时间序列 rad = np.pi / 4 # 基波周期 N=8 plt.stem(n, sg.square(rad * n, 0.5)) # 绘制离散周期方波图像 plt.xticks(np.arange(-10, 10, step=5.0)) # 横坐标间隔 plt.grid() # 显示网格 plt.show() # 显示图像</pre>	

2.2 序列的基本运算

这里假设基本信号为 $x = \text{np.array}([1,2,5,4,6,9])$ ，离散时间序列为 $n = \text{np.arange}(\text{len}(x))$ ，相关 Python 代码参考 `SignalOP.ipynb`

运算名称	数学表达式	Python 实现
信号幅度变化	$y[n] = Ax[n]$	<code>Y=A*x</code>
信号时移	$y[n] = x[n - k]$	<code>Y=np.append(np.zeros(k),x)</code>
信号翻转	$y[n] = x[-n]$	<code>Y = x[::-1]</code>
信号累积	$y[n] = \sum_{n=-\infty}^{\infty} x[n]$	<code>Y= np.sum(x)</code>
信号差分（或者近似微分）	$y[n] = x[n + 1] - x[n]$	<code>Y= np.diff(x)</code>

信号求和	$y = \sum_{n=n_1}^{n_2} x[n]$	$Y = \text{np.sum}(x[n1:n2+1])$
信号能量	$E_x = \sum_{n=-\infty}^{\infty} x[n] ^2$	$E = \text{np.sum}(\text{np.abs}(x)**2)$
信号功率	$P_x = \frac{1}{N} \sum_{n=0}^{N-1} x[n] ^2$	$E = \text{np.sum}(\text{np.abs}(x**2)/N)$
信号相加	$y[n] = x_1[n] + x_2[n]$	$Y = x1 + x2$
信号相乘	$y[n] = x_1[n]x_2[n]$	$Y = x1 * x2$

2.3 相关测试

进行平移、翻转和尺度变换

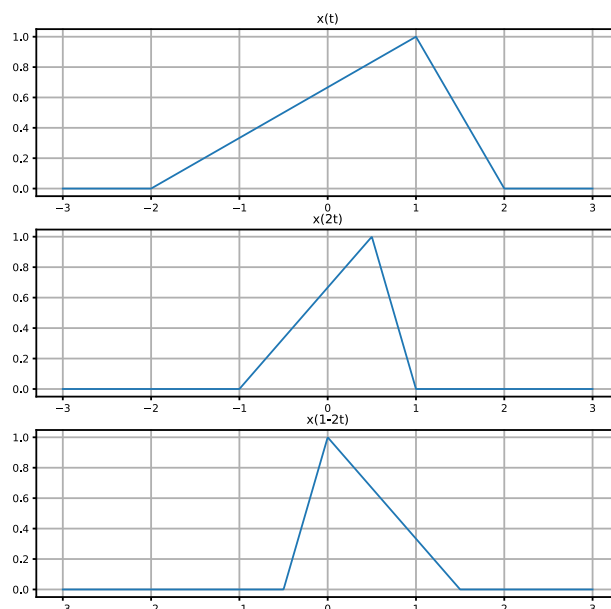
- A. 对于如 MATLAB 中 $x(t)=\text{tripuls}(t,w,s)$ 的三角波， t 为时间序列， w 为三角波信号的宽度， s 为三角波信号的斜度。 `scipy.signal.sawtooth(t, width=1)` 用于产生周期三角波信号，为此需要通过 Python 编写用于产生非周期三角波信号的函数 `triangle_wave(x,width,skew)`

x: 时间

width: 三角波信号的宽度

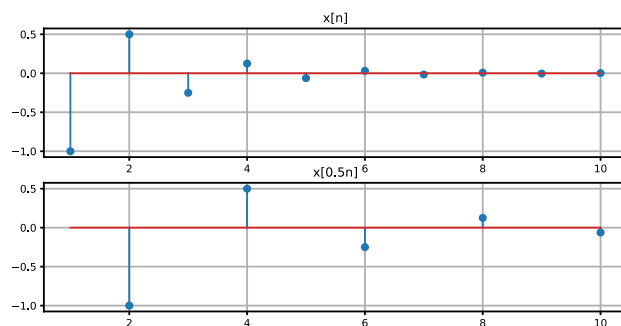
skew: 三角波信号的斜度，当 `skew` 为 0 时，函数产生一个对称的三角形脉冲，`skew` 范围 $[-1, 1]$ 。

则 $x(t)$, $x(2t)$, $x(1-2t)$ 的 Python 程序为 **experiment_1_1.py**:



图一、三角波的平移、翻转和尺度变换

- B. 对离散指数序列 $x[n] = Aa^n$ 以及 $x[0.5n]$ ，程序见 **experiment_1_2.py**。



图二、离散指数序列及其尺度变换

3 实验内容与步骤

- 1) 验证程序实例中的相关程序。
- 2) 利用 $x(t) = u(t) - u(t - 2) + u(t - 0.5) - u(t - 1.5)$, $(-3 \leq t \leq 3)$, 编写相关程序, 绘制出 $x(-2t)$, $x(t/2 + 1)$ 和 $5x(t)$ 的波形。

- 3) 设 $x[n] = \begin{cases} -1 & n < -2 \\ n & -2 \leq n \leq 1 \\ \frac{1}{n} & n > 1 \end{cases}$, 编写程序, 绘制 $x[-n]$, $x[2n + 2]$, $x[n/2]$, $(-20 \leq n \leq 20)$ 。

4 实验报告要求

整理并给出实验内容与步骤中的程序代码与产生的波形。

实验二 信号的卷积

1 实验目的

深刻理解卷积运算，掌握离散时间信号卷积和、连续时间信号卷积积分的编程计算方法。

2 实验原理

在Python语言中，Scipy库中的[scipy.signal](#)可用于计算信号的卷积。
本次实验版本Python=3.6.5，numpy=1.14.5，scipy=1.1.0，matplotlib= 3.1.1。

本实验Python库导入：

```
# 导入 需要的 library 库
import numpy as np # 科学计算
import matplotlib.pyplot as plt # 画图
import scipy.signal as sg # 导入 scipy 的 signal 库 命名为 sg
```

2.1 相关测试

编程计算卷积

- A. 连续时间信号的卷积: 求信号 $x_1(t)=u(t)$ 与 $x_2(t)=e^{-3t}u(t)$ 的卷积积分。注意： $x_2(t)$ 是个持续时间无限长的连续时间信号，而计算机数值计算只能计算有限时长的离散信号，因此需要对 $x_2(t)$ 进行抽样离散化，并且要使得 $x_2(t)$ 衰减到足够小。样例程序可见 **experiment_2_1.py**。

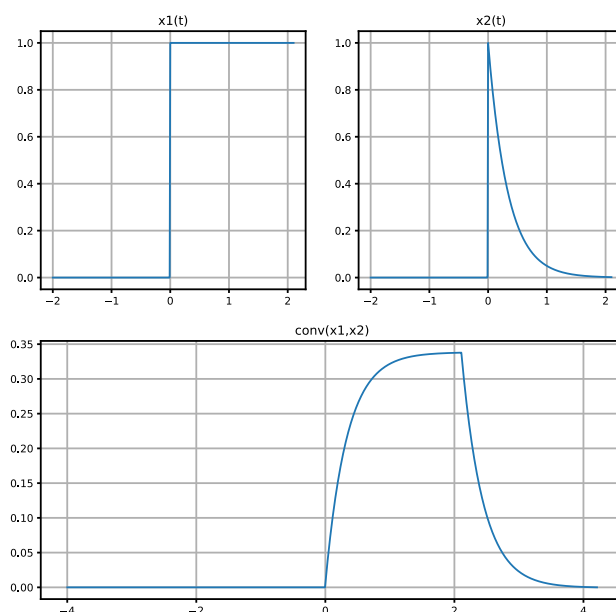


图 1 连续时间信号卷积图示

B. 离散时间信号的卷积：计算 $x[k]=[1,2,1,1,0,-3;k=0,1,2,3,4,5]$ 与 $h[k]=[1,-1,1;k=0,1,2]$ 的卷积和。样例程序见 **experiment_2_2.py**。

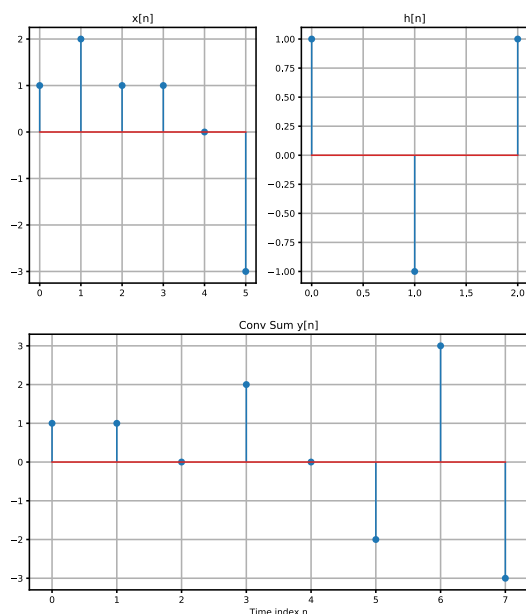


图 2 离散时间信号卷积图示

3 实验内容与步骤

- 4) 验证程序实例中的相关程序；
- 5) 编写程序，绘制下列信号的卷积波形。

A. 已知 $x_1(t) = tu(t)$, $x_2(t) = e^{-t}u(t)$, 求 $x_1(t) * x_2(t)$; (要求：抽样频率 $f_s=1000$; 时间 $t = -1.1 \sim 2.1$)

B. 已知 $x[n] = [3, 2, 1, -2, 1, 0, 4, 0, 3; n = 0:8]$; $h[n] = [1, -2, 3, -4, 3, 2, 1; n = 0:6]$; 求 $x[n] * h[n]$ 。

4 实验报告要求

整理并给出实验内容与步骤中的程序代码与产生的波形。

5 分析题

- A. 连续时间与离散时间信号的卷积定义分别是什么？卷积的作用是什么？
conv 函数只输出了卷积的结果，没有输出对于的时间向量，如何使得时间向量和卷积的结果对应起来？
- B. 两个离散时间信号进行卷积和所得新序列的时域区间与原来的两个序列具有什么关系？

实验三 周期信号的傅里叶级数

1 实验目的

- A. 理解周期信号的傅里叶级数分解，掌握傅里叶级数的计算方法；
- B. 利用傅里叶级数理解并计算 LTI 系统对周期信号的响应。

2 实验原理

本次实验版本Python=3.6.5, numpy=1.14.5, scipy=1.1.0, sympy=1.1.1, matplotlib= 3.1.1。

对于本实验Python库约定：

导入 需要的库

import numpy as np # 科学计算

import matplotlib.pyplot as plt # 画图

import scipy.signal as sg # 导入 scipy 的 signal 库 命名为 sg

from scipy.integrate import quad # 用于求解定积分

import sympy

import warnings

warnings.filterwarnings("ignore") # 去掉常规警告

2.1 周期信号的傅里叶级数

设 $x(t)$ 为连续时间周期信号，其基波周期为 T ，基波频率为 $\omega_0 = \frac{2\pi}{T}$ 。该信号满足 Dirichlet 条件，可以展开成傅里叶级数。傅里叶级数有复指数形式和三角函数形式两种，其中复指数形式为：

$$\begin{cases} x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t} \\ a_k = \frac{1}{T} \int_T x(t) e^{-jk\omega_0 t} dt \end{cases} \quad (1)$$

三角函数形式为：

$$\begin{cases} x(t) = \frac{c_0}{2} + \sum_{k=1}^{\infty} c_k \cos(k\omega_0 t) + \sum_{k=1}^{\infty} d_k \sin(k\omega_0 t) \\ c_0 = \frac{2}{T} \int_T x(t) dt \\ c_k = \frac{2}{T} \int_T x(t) \cos(k\omega_0 t) dt \\ d_k = \frac{2}{T} \int_T x(t) \sin(k\omega_0 t) dt \end{cases} \quad (2)$$

在用 Python 计算傅里叶级数的系数时，本质上是对信号进行积分运算，可采用以下两种方式进行：

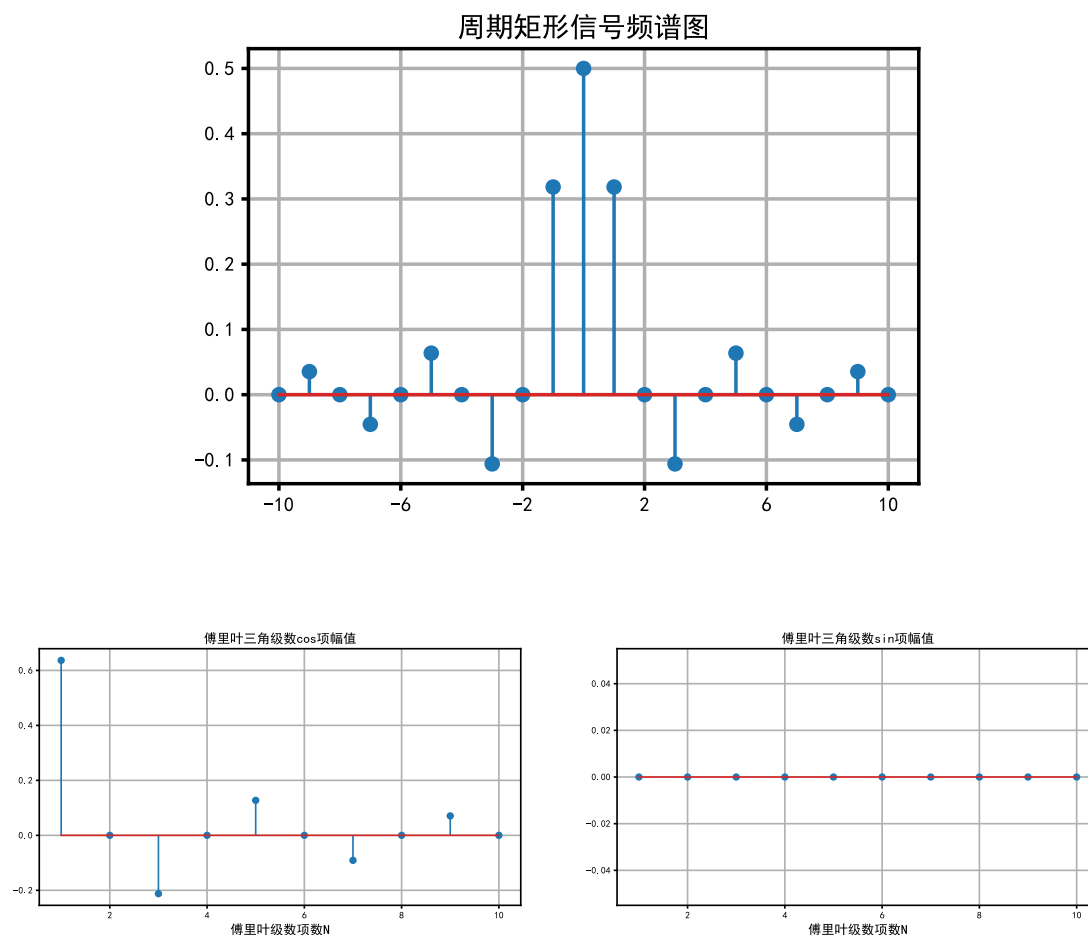
- 1) 符号积分: `scipy.integrate(f, (x, a, b))`, `f` 为符号表达式, `x` 为积分变量, `a` 为积分下限, `b` 为积分上限;
- 2) 数值积分: `scipy.integrate.quad(func, a, b)`, `func` 是被积分的函数名, `a` 为积分下限, `b` 为积分上限。

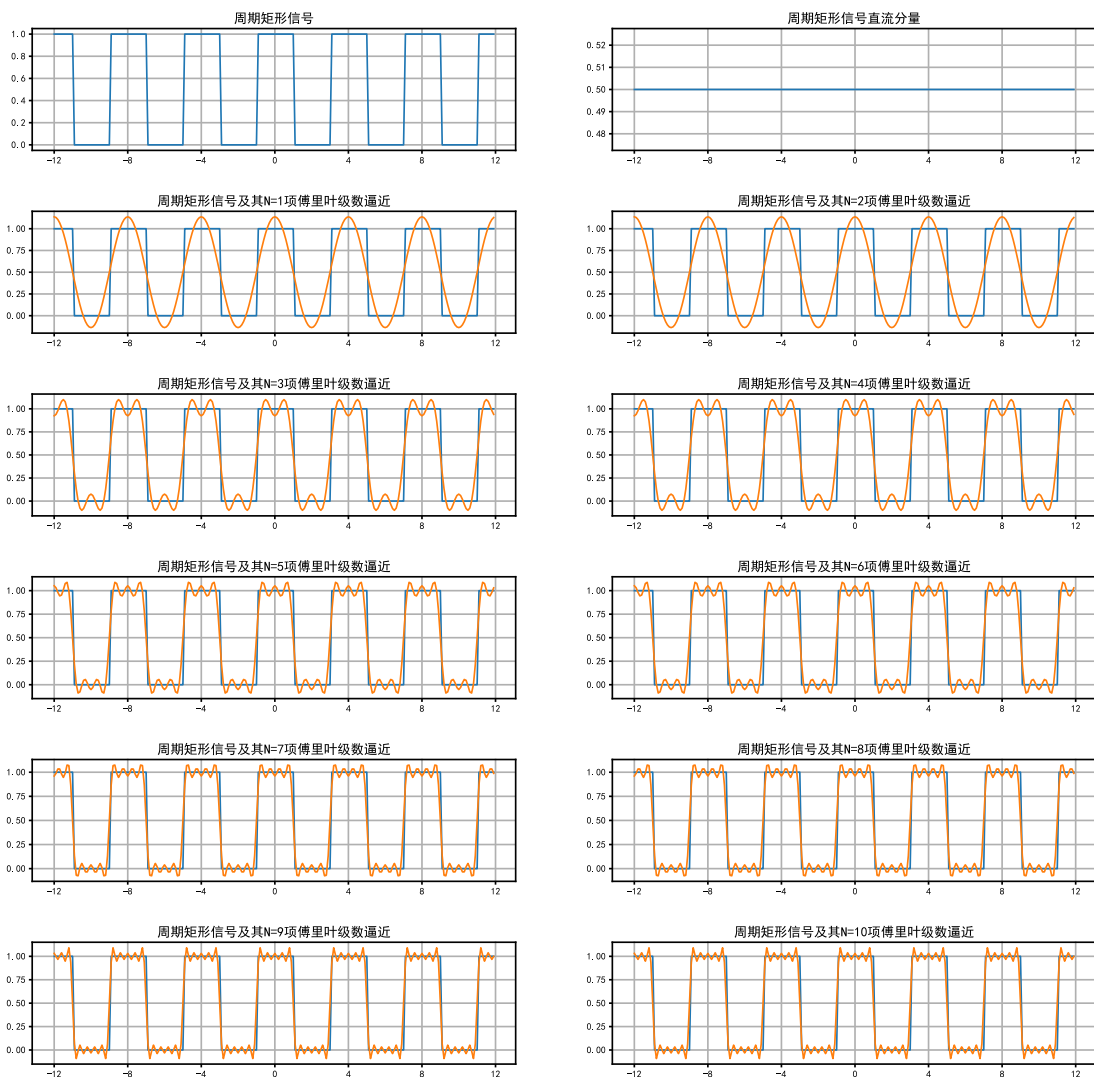
2.2 程序实例

给定基波周期为 4, 矩形脉冲带宽为 2, 幅度为 1 的连续时间周期信号 $x(t)$, 用 Python 分别计算并绘制其复指数形式 (`experiment_3_1.py`) 和三角函数形式的傅里叶级数系数, 并用有限项级数

$$x_N(t) = \sum_{k=-N}^N a_k e^{jk\omega_0 t} = \frac{c_0}{2} + \sum_{k=1}^N c_k \cos(k\omega_0 t) + \sum_{k=1}^N d_k \sin(k\omega_0 t)$$

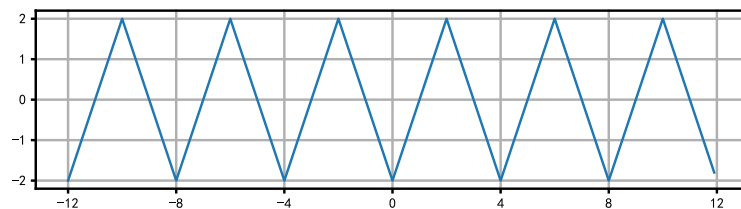
$N = 1, \dots, 10$, 逼近 $x(t)$ (`experiment_3_2.py`)。





3 实验内容与步骤

- A. 给定下图的周期三角信号 $x(t)$ ，用 Python 分别计算并绘制其复指数形式和三角函数形式的傅里叶级数系数，并用有限项级数 $x_N(t)$ ， $N = 1, \dots, 10$ ，逼近 $x(t)$ 。



- B. 对 2.2 中的周期方波和上面 3-A 中的周期三角波，分别绘图演示其有限项级数 $x_N(t)$ 当项数 $N = 10, 100, 1000$ 时对 $x(t)$ 的逼近效果。对比分析两者逼近过程中是否出现吉布斯（Gibbs）现象并解释原因。

C. 已知微分方程

$$\frac{dy(t)}{dt} + 2y(t) = x(t)$$

满足初始松弛条件。计算该方程所对应系统的频率响应。将 3-A 中的周期三角波输入该系统，用 Python 求解并绘制其输出响应。

4 实验要求

完成上面 3 中的实验内容与步骤，整理并给出对应的程序代码与波形，完成其中的问题。

5 实验思考题

请由式（1）中复指数形式的傅里叶级数推导式（2）中三角函数形式的傅里叶级数。

实验四 离散时间非周期信号的傅里叶变换

1 实验目的

- (1) 深入理解和掌握离散时间非周期信号的傅里叶变换及计算方法；
- (2) 熟悉离散时间傅里叶变换的性质；
- (3) 理解离散时间 LTI 系统的频域分析原理和方法，掌握离散时间 LTI 系统的频率响应求解方法，并能编程绘制相应的幅频、相频响应曲线。

2 实验原理

若 $x[n]$ 是离散时间非周期信号，则其离散时间傅里叶变换为

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x[n]e^{-j\omega n} \quad (1)$$

对应的离散时间傅里叶反变换为

$$x[n] = \frac{1}{2\pi} \int_{2\pi} X(e^{j\omega}) e^{j\omega n} d\omega \quad (2)$$

这里的积分区间可以是任意的 2π 周期。

Python 提供了 `scipy.signal.freqz()` 函数可以近似求解离散时间傅里叶变换。对于(1)式的无穷级数求和还可用 SymPy 符号函数求和的方法，如 `sum()`、`summation()`、`gosper_sum()` 等函数都可用于级数求和操作。对于(2)式可用 SymPy 符号函数积分的方法进行，如 `integrate()` 函数。

此外，也可以根据定义，使用求和函数 `numpy.sum()` 和积分函数 `numpy.integrate.quad()` 对信号的(1)、(2)式进行数值计算。

通过 Python 的 NumPy 库和 SciPy 库可以产生基本的信号，如阶跃信号、指数信号、脉冲信号等等，其中 [scipy.signal](#) 可用于计算信号的卷积。

本次实验中程序库版本 Python=3.6.5，numpy=1.14.5，scipy=1.1.0，matplotlib= 3.1.1。导入方式如：

```
import numpy as np # 科学计算
import matplotlib.pyplot as plt # 画图
import scipy.signal as sg # 导入 scipy 的 signal 库
from scipy.integrate import quad, simps # 用于求积分

import warnings
warnings.filterwarnings("ignore") # 去掉常规警告
```

2.1 离散时间傅里叶变换的性质

(1) 线性：若 $x_1[n] \leftrightarrow X_1(e^{j\omega})$ ， $x_2[n] \leftrightarrow X_2(e^{j\omega})$ ，则 $a_1x_1[n] + a_2x_2[n] \leftrightarrow a_1X_1(e^{j\omega}) + a_2X_2(e^{j\omega})$ ；

(2) 时移：若 $x[n] \leftrightarrow X(e^{j\omega})$ ，则 $x[n - n_0] \leftrightarrow e^{-j\omega n_0}X(e^{j\omega})$ ；

(3) 频移：若 $x[n] \leftrightarrow X(e^{j\omega})$ ，则 $e^{j\omega n_0}x[n] \leftrightarrow X(e^{j(\omega - \omega_0)})$ ；

(4) 共轭与共轭对称性：若 $x[n] \leftrightarrow X(e^{j\omega})$ ，则 $x^*[n] \leftrightarrow X^*(e^{-j\omega})$ ；

(5) 差分与累加：若 $x[n] \leftrightarrow X(e^{j\omega})$ ，则 $x[n] - x[n - 1] \leftrightarrow (1 - e^{-j\omega})X(e^{j\omega})$ ，
 $\sum_{k=-\infty}^n x[k] \leftrightarrow \frac{X(e^{j\omega})}{1 - e^{-j\omega}} + \pi X(e^{j0}) \sum_{k=-\infty}^{\infty} \delta(\omega - 2\pi k)$ ；

(6) 时间反转：若 $x[n] \leftrightarrow X(e^{j\omega})$ ，则 $x[-n] \leftrightarrow X(e^{-j\omega})$ ；

(7) 时域内插：若 $x[n] \leftrightarrow X(e^{j\omega})$ ， $x_k[n] = \begin{cases} x[n/k], & n \text{ 为 } k \text{ 的整数倍} \\ 0, & \text{其他 } n \end{cases}$ ，则 $x_k[n] \leftrightarrow X(e^{jk\omega})$ 。

(8) 卷积性质：若 $x_1[n] \leftrightarrow X_1(e^{j\omega})$ ， $x_2[n] \leftrightarrow X_2(e^{j\omega})$ ，则 $x_1[n] * x_2[n] \leftrightarrow X_1(e^{j\omega})X_2(e^{j\omega})$ ；

(9) 相乘性质：若 $x_1[n] \leftrightarrow X_1(e^{j\omega})$ ， $x_2[n] \leftrightarrow X_2(e^{j\omega})$ ，则 $x_1[n] * x_2[n] \leftrightarrow \frac{1}{2\pi} X_1(e^{j\omega}) \otimes X_2(e^{j\omega})$ 。

2.2 离散时间 LTI 系统的频率响应和频域分析

设离散时间 LTI 系统的单位脉冲响应为 $h[n]$ ，输入信号为 $x[n]$ ，输出响应为 $y[n]$ ，则 $y[n] = x[n] * h[n]$ 。对应的傅里叶变换表达为 $Y(e^{j\omega}) = X(e^{j\omega}) \cdot H(e^{j\omega})$ ，其中 $H(e^{j\omega})$ 为系统的频率响应，反映了系统的频域特性，模-相位形式为 $H(e^{j\omega}) = |H(e^{j\omega})|e^{j\angle H(e^{j\omega})}$ 。

对于由线性常系数差分方程

$$\sum_{k=0}^N a_k y[n - k] = \sum_{k=0}^M b_k x[n - k]$$

描述的离散时间 LTI 系统，频率响应为：

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = \frac{\sum_{k=0}^M b_k e^{-jk\omega}}{\sum_{k=0}^N a_k e^{-jk\omega}}$$

对于上述频率响应，Python 工具箱中提供的 `scipy.signal.freqz()` 可以直接计算，调用形式为：

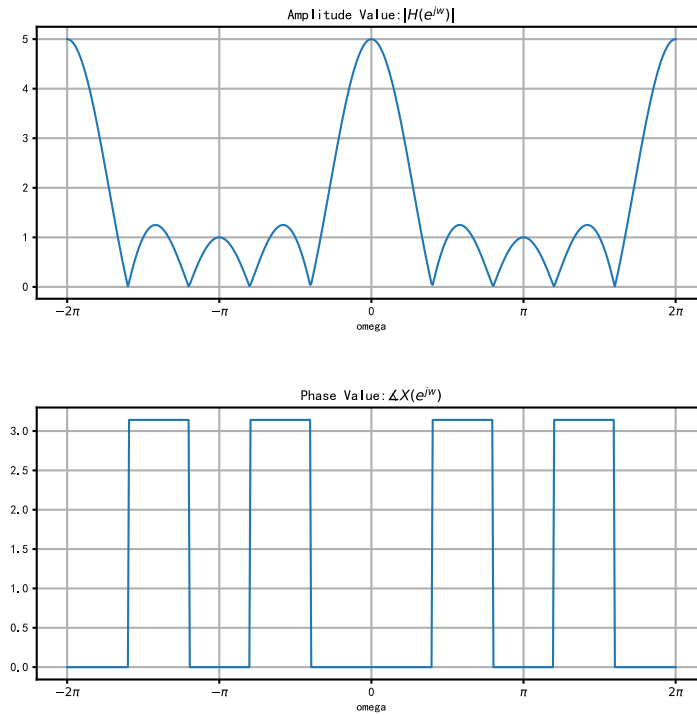
$$[w, h] = \text{freqz}(b, a, n)$$

其中 **b** 为频率响应中分子多项式的系数向量，**a** 为分母多项式的系数向量，**n** 为返回的频率响应对应的频率点数，**w** 为角频率向量，向量 **h** 则是返回在 **w** 所定义的频率点上频率响应值。

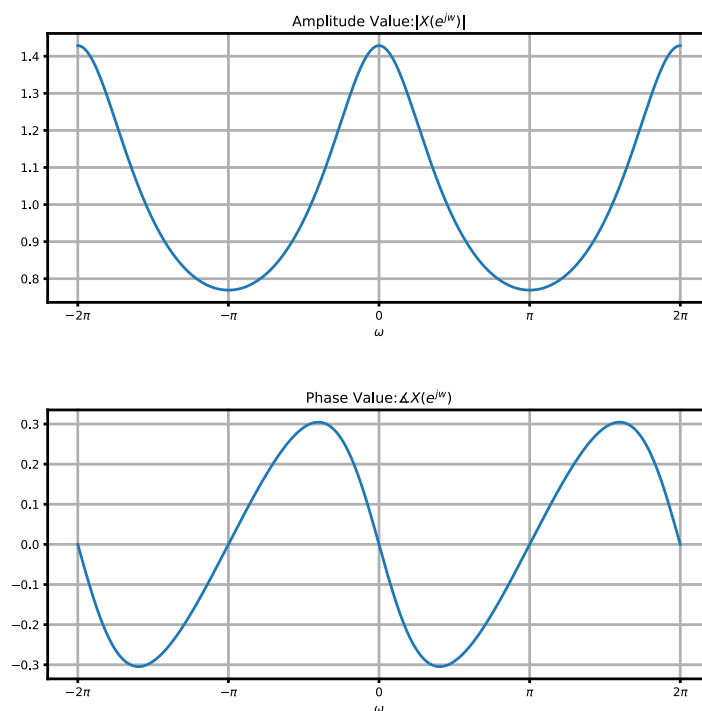
提示：当 $b_k = x[k]$ ， $a_k = \delta[k]$ ，且当 $k < 0$ ，有 $x[k] = 0$ 时， $H(e^{j\omega})$ 即为离散时间非周期信号 $x[n]$ 的离散时间傅里叶变换近似解。

3 程序实例

(1) 画出频率响应 $H(e^{j\omega}) = \frac{\sin(\omega(N_1 + \frac{1}{2}))}{\sin(\frac{\omega}{2})}$ 的幅频、相频曲线，取 $N_1 = 2$ 。原信号为 $h[n] = \begin{cases} 1, & |n| \leq N_1 \\ 0, & |n| > N_1 \end{cases}$ 。（见 `experiment_4_1.py`）

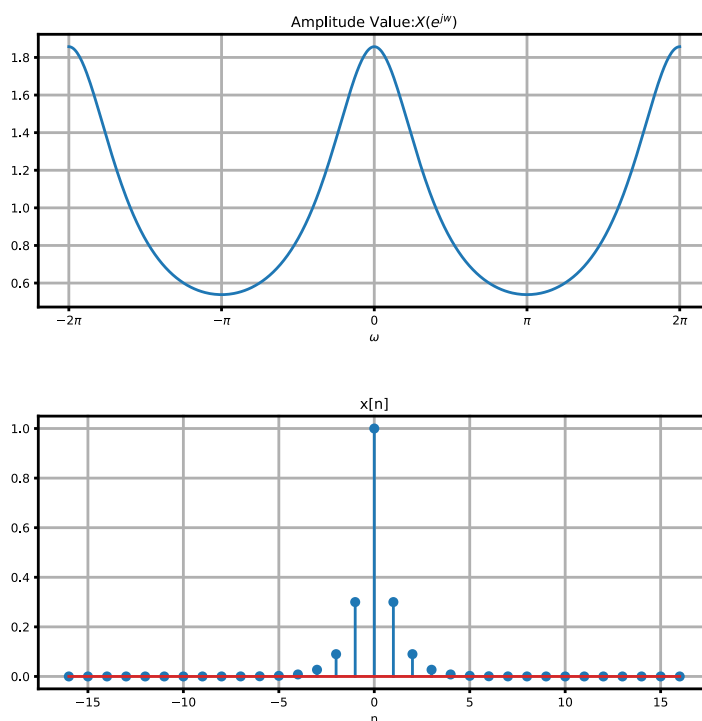


(2) 求离散时间信号 $x[n] = 0.3^n u[n]$ 的傅里叶变换（见 `experiment_4_2.py`）。

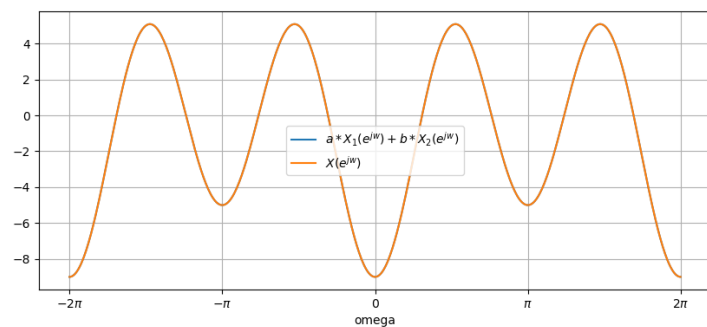
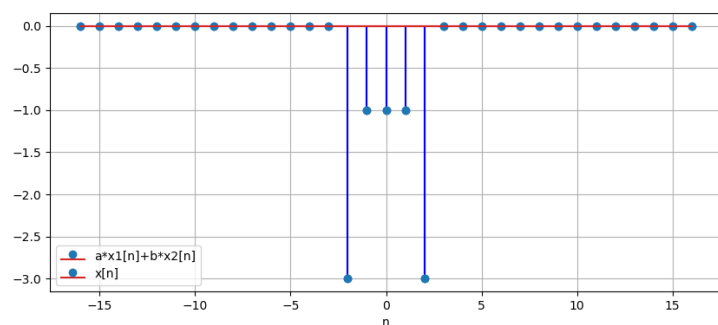


(3) 求 $X(e^{j\omega}) = \frac{1-a^2}{1-2a\cos\omega+a^2}$, $|a| < 1$ 的离散时间傅里叶逆变换, 这里取 $a = 0.3$ 。

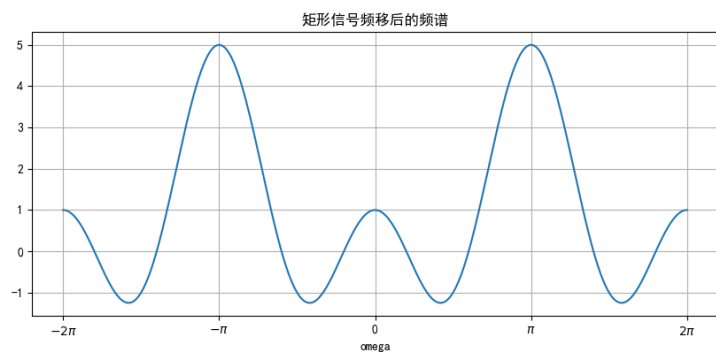
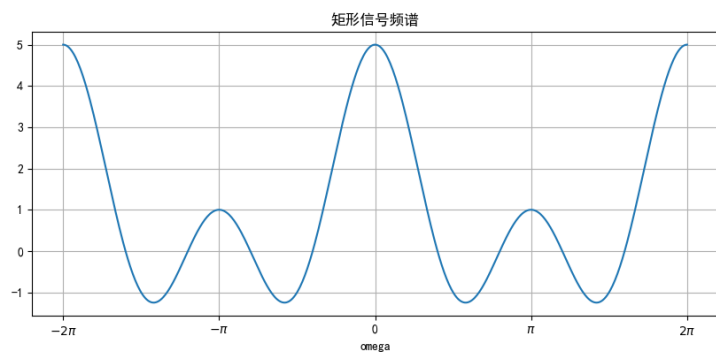
绘制 $|X(e^{j\omega})|$ 和原信号 $x[n]$ 的图像 (见 experiment_4_3.py)。



(4) 信号 $x_1[n] = u[n+1] - u[n-2]$, $x_2[n] = u[n+2] - u[n-3]$, $x[n] = ax_1[n] + bx_2[n]$ 。取 $a = 2, b = -3$ 。请验证离散时间傅里叶变换的线性特性。(见 experiment_4_4.py)



(5) 考虑频移信号 $x[n] = (u[n+2] - u[n-3])\cos(\omega_0 n)$, 其中 $\omega_0 = \pi \text{ rad/s}$ 。请绘制原矩形脉冲信号 $u[n+2] - u[n-3]$ 的频谱及频移后信号 $x[n]$ 的频谱。(见 experiment_4_5.py)



4 实验内容和步骤

(1) 编程计算双边指数衰减信号 $x[n] = e^{-2|n|}$ 的离散时间傅里叶变换, 并验证其时域内插, 即

$$x_k[n] = \begin{cases} x[n/k], & n \text{ 为 } k \text{ 的整数倍} \\ 0, & \text{其他 } n \end{cases}$$

的离散时间傅里叶变换，取 $k = 3$ 。请分别绘制 $x[n]$ 和 $x_k[n]$ 的幅频曲线和相频曲线。

(2) 考虑差分方程 $y[n] - ay[n-1] = x[n]$ ，其中 $|a| < 1$ 。取 $a = 0.2$ ，编程求解该方程所描述系统的频率响应，并：

- (a) 画出系统的幅频和相频特性曲线；
- (b) 求解系统的单位脉冲响应并绘制出图形。

(3) 设离散时间双边指数衰减信号 $x[n] = e^{-2|n|}$, $y[n] = x[n] * x[n]$ ，请

- (a) 编程用卷积性质求解 $Y(e^{j\omega})$ 与 $y[n]$ ，并绘制 $x[n]$ 、 $|Y(e^{j\omega})|$ 和 $y[n]$ 的图像；
- (b) 编程用时域卷积求解 $y[n]$ ，绘制 $y[n]$ 的图像，并与 (a) 的结果比较。

5 实验预习

离散时间非周期信号的傅里叶变换有哪些性质？

6 实验报告要求

- (1) 完成上面 4 中的实验内容与步骤，整理并给出对应的程序代码与波形；
- (2) 结合实验，分析结果的意义。