# 上机实习\_2

## 任务一

1. 对于信号f(t)，画出信号波形

代码：

**def** fun(x,x1,x2):  
 **if** x < x1:  
 **return** x-1  
 **elif** x >= x2:  
 **return** x-2  
 **else**:  
 **return** 1  
X = np.linspace(1, 4, 3000)  
Y = np.array([fun(t, 2, 3) **for** t **in** X])  
plt.title(**u'f(t)'**)  
plt.plot(X,Y,c=**'b'**)  
plt.axvline(x=0, color=**'k'**)  
plt.axhline(y=0, color=**'k'**)  
plt.show()

运行结果：

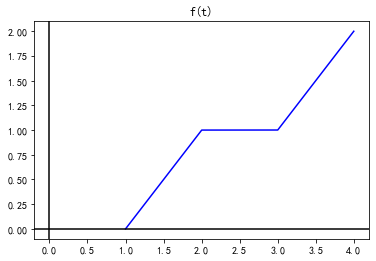


图 1 f(t)原信号与加噪声后信号

1. 给信号加入随机噪声，画出波形

代码：

**def** fun(x,x1,x2):  
 **if** x < x1:  
 **return** x-1  
 **elif** x >= x2:  
 **return** x-2  
 **else**:  
 **return** 1  
X = np.linspace(1, 4, 3000)  
Y = np.array([fun(t, 2, 3)+random.randint(0, 1)\*random.randint(0, 1)\*random.uniform(-0.25, 0.25) **for** t **in** X])  
plt.title(**u'f(t)加噪声'**)  
plt.plot(X,Y,c=**'b'**)  
plt.axvline(x=0, color=**'k'**)  
plt.axhline(y=0, color=**'k'**)  
plt.show()

运行结果：

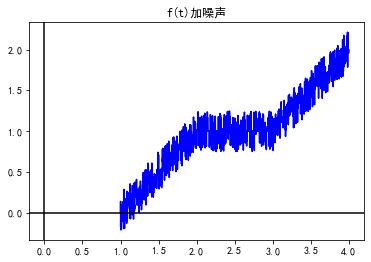


图 2 相加信号

1. 采用三点、五点等不同点的均值滤波方法对信号进行去噪，画出去噪后的波形

代码：

**def** mean\_filter(arr, k):  
 *# applies mean filter to 1-d array with the kernel size 2k+1 . Write your code here* p=len(arr)  
 arr2=np.zeros(2\*k+p, dtype=float)  
 arr3=np.zeros(2\*k+p, dtype=float)  
 arr4=np.zeros(p, dtype=float)  
  
 **for** i **in** range(p):  
 arr2[k+i]=arr[i]  
  
 **for** i **in** range(k,k+p):  
 sum=0  
 **for** j **in** range(-k,k+1):  
 sum+=arr2[i+j]  
 arr3[i]=sum/float(2\*k+1)  
  
 **for** i **in** range(p):  
 arr4[i]=arr3[k+i]  
  
 **return** arr4

Y\_mean\_3 = mean\_filter(Y, 3)  
plt.title(**u'三点均值滤波后 f(t)'**)  
plt.plot(X,Y\_mean\_3,c=**'b'**)  
plt.axvline(x=0, color=**'k'**)  
plt.axhline(y=0, color=**'k'**)  
plt.show()

Y\_mean\_5 = mean\_filter(Y, 5)  
plt.title(**u'五点均值滤波后 f(t)'**)  
plt.plot(X,Y\_mean\_5,c=**'b'**)  
plt.axvline(x=0, color=**'k'**)  
plt.axhline(y=0, color=**'k'**)  
plt.show()

运行结果：

|  |  |
| --- | --- |
|  |  |

图 3 三点、五点均值滤波f(t)

1. 采用三点、五点等不同点的中值滤波方法对信号进行去噪，画出去噪后的波形

代码：

**from** scipy **import** signal  
  
Y\_med\_3 = signal.medfilt(Y, kernel\_size=3)  
plt.title(**u'三点中值滤波后 f(t)'**)  
plt.plot(X,Y\_med\_3,c=**'b'**)  
plt.axvline(x=0, color=**'k'**)  
plt.axhline(y=0, color=**'k'**)  
plt.show()

Y\_med\_5 = signal.medfilt(Y, kernel\_size=5)  
plt.title(**u'五点中值滤波后 f(t)'**)  
plt.plot(X,Y\_med\_5,c=**'b'**)  
plt.axvline(x=0, color=**'k'**)  
plt.axhline(y=0, color=**'k'**)  
plt.show()

运行结果：

|  |  |
| --- | --- |
|  |  |

图 4 三点、五点中值滤波

1. 分析去噪效果

均值滤波在最后的时刻会出现在均值后信号出现变形的情况，综合以上图片，五点中值滤波效果最好。

## 二、 任务二

1. 设计一个偶对称的周期方波信号，设计一个等于单周期的非周期方波信号，画出信号波形

代码：

**from** scipy **import** signal  
N1 = 30000  
t = np.linspace(-1.5, 1.5, N1, endpoint=**False**)  
sq = signal.square(2 \* np.pi \* (t+0.25)) *#square函数生成方波  
# 方波信号*plt.plot(t, (sq+1)/2, c=**'b'**)  
plt.xlim(-1.5, 1.5)  
plt.title(**u'偶对称周期方波信号-1Hz 采样10000Hz'**)  
plt.axvline(x=0, color=**'k'**)  
plt.axhline(y=0, color=**'k'**)  
plt.ylabel(**'Amplitude'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.xlabel(**'Time'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.show()

N1 = 30000  
t = np.linspace(-0.5, 0.5, N1, endpoint=**False**)  
sq = signal.square(2 \* np.pi \* (t+0.25)) *#square函数生成方波  
# 方波信号*plt.plot(t, (sq+1)/2, c=**'b'**)  
plt.xlim(-0.75, 0.75)  
plt.title(**u'非周期方波信号'**)  
plt.axvline(x=0, color=**'k'**)  
plt.axhline(y=0, color=**'k'**)  
plt.ylabel(**'Amplitude'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.xlabel(**'Time'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.show()

运行结果：

|  |  |
| --- | --- |
|  |  |

图 5 偶对称的周期方波信号和等于单周期的非周期方波信号

1. 画出非周期方波信号的频谱

代码：

f = []  
A = []  
**for** i **in** range(-20000, 20000):  
 f.append(i/2000)  
 **if** i!=0:  
 An = abs(math.sin(0.5\*math.pi\*i/2000)/(math.pi\*i/2000))  
 **else**:  
 An = 0.5  
 A.append(An)  
  
N1 = 150000  
t1 = np.linspace(0, 15, N1, endpoint=**False**)  
phi1 = signal.square(1/2 \* np.pi \* (t1+2))  
  
N1 = 150000  
t2 = np.linspace(-15, 0, N1, endpoint=**False**)  
phi2 = signal.square(1/2 \* np.pi \* (t2))  
  
*# 振幅频谱图*plt.plot(f, A, label=**"all"**, c=**'b'**)  
plt.xlim(-10, 10)  
plt.axhline(y=0, ls=**"-"**, c=**"k"**)  
plt.axvline(x=0, color=**'k'**)  
plt.title(**u'非周期方波信号-1Hz 振幅频谱图'**)  
plt.ylabel(**u'|x(f)|'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.xlabel(**u'f'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.show()  
  
*# 复数频谱图*plt.plot(t1, math.pi/4\*(phi1+1), label=**"all"**, c=**"b"**)  
plt.plot(t2, -math.pi/4\*(phi2+1), label=**"all"**, c=**"b"**)  
plt.xlim(-10, 10)  
plt.axhline(y=0, ls=**"-"**, c=**"k"**)  
plt.axvline(x=0, color=**'k'**)  
plt.title(**u'非周期方波信号-1Hz 相位频谱图'**)  
plt.ylabel(**u'φ'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.xlabel(**u'f'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.ylim(-3, 3)  
plt.show()

运行结果：

|  |  |
| --- | --- |
|  |  |

图 6 非周期方波信号的频谱

1. 采用不同的频域抽样间隔f0对非周期信号的频谱进行抽样，并用抽样后的离散频谱恢复连续谱（给出3个以上的频率点）

代码：

*# 取[100, 100], f0 = 1***def** x\_new(f0, t):  
 res = 0  
 **for** i **in** range(-20, 20):  
 **if** i != 0:  
 res = res + math.sin(0.5\*math.pi\*i\*f0)/(math.pi\*i\*f0) \* math.cos(2\*math.pi\*i\*f0\*t)  
 **else**:  
 res = res + 0.5  
 **return** res\*f0  
  
X = []  
t = [i/1000 **for** i **in** range(-2000,2000)]  
f0 = 1  
**for** i **in** t:  
 X.append(x\_new(f0, i))  
  
plt.plot(t, X, label=**"all"**, c=**"b"**)  
plt.xlim(-1.5, 1.5)  
plt.axhline(y=0, ls=**"-"**, c=**"k"**)  
plt.axvline(x=0, color=**'k'**)  
plt.title(**u'非周期方波信号 恢复信号图'**)  
plt.ylabel(**u'x'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.xlabel(**u't'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.ylim(-0.25, 1.5)  
plt.show()

*# 取[100, 100], f0 = 0.5***def** x\_new(f0, t):  
 res = 0  
 **for** i **in** range(-200, 200):  
 **if** i != 0:  
 res = res + math.sin(0.5\*math.pi\*i\*f0)/(math.pi\*i\*f0) \* math.cos(2\*math.pi\*i\*f0\*t)  
 **else**:  
 res = res + 0.5  
 **return** res\*f0  
  
X = []  
t = [i/1000 **for** i **in** range(-5000,5000)]  
f0 = 0.5  
**for** i **in** t:  
 X.append(x\_new(f0, i))  
  
plt.plot(t, X, label=**"all"**, c=**"b"**)  
plt.xlim(-3, 3)  
plt.axhline(y=0, ls=**"-"**, c=**"k"**)  
plt.axvline(x=0, color=**'k'**)  
plt.title(**u'非周期方波信号 恢复信号图'**)  
plt.ylabel(**u'x'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.xlabel(**u't'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.ylim(-0.5, 1.5)  
plt.show()

*# 取[100, 100], f0 = 2.5***def** x\_new(f0, t):  
 res = 0  
 **for** i **in** range(-40, 40):  
 **if** i != 0:  
 res = res + math.sin(0.5\*math.pi\*i\*f0)/(math.pi\*i\*f0) \* math.cos(2\*math.pi\*i\*f0\*t)  
 **else**:  
 res = res + 0.5  
 **return** res\*f0  
  
X = []  
t = [i/1000 **for** i **in** range(-2000,2000)]  
f0 = 1.25  
**for** i **in** t:  
 X.append(x\_new(f0, i))  
  
plt.plot(t, X, label=**"all"**, c=**"b"**)  
plt.xlim(-1.5, 1.5)  
plt.axhline(y=0, ls=**"-"**, c=**"k"**)  
plt.axvline(x=0, color=**'k'**)  
plt.title(**u'非周期方波信号 恢复信号图'**)  
plt.ylabel(**u'x'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.xlabel(**u't'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.ylim(-0.5, 1.5)  
plt.show()

运行结果：

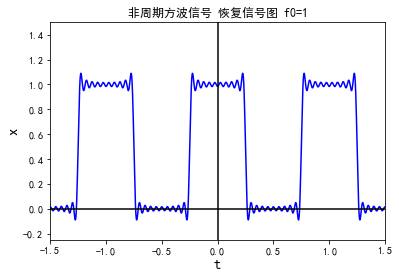


图 7 恢复连续谱 f0=1

|  |  |
| --- | --- |
|  |  |

图 8 恢复连续谱（分别是f0 = 0.5, 2.5）

f = []  
A = []  
**for** i **in** range(-1000, 1000):  
 f.append(i)  
 **if** i!=0:  
 An = math.sin(0.5\*math.pi\*i)/(math.pi\*i)  
 **else**:  
 An = 0.5  
 A.append(An)

|  |  |
| --- | --- |
|  |  |

图 9 f0=1 抽样恢复

f = []  
A = []  
**for** i **in** range(-1000, 1000):  
 f.append(i/2)  
 **if** i!=0:  
 An = math.sin(0.5\*math.pi\*i/2)/(math.pi\*i/2)  
 **else**:  
 An = 0.5  
 A.append(An)

|  |  |
| --- | --- |
|  |  |

图 10 f0=0.5 抽样恢复

f = []  
A = []  
**for** i **in** range(-1000, 1000):  
 f.append(i\*2.5)  
 **if** i!=0:  
 An = math.sin(0.5\*math.pi\*i\*2.5)/(math.pi\*i\*2.5)  
 **else**:  
 An = 0.5  
 A.append(An)

|  |  |
| --- | --- |
|  |  |

图 11 f0=2.5 抽样恢复

1. 比较不同f0的方波信号的频谱所恢复的连续频谱与标准方波表达式频谱，并加以说明

当f0<时，此时出现混叠现象；当f0=时，此时复原的信号和原信号，除了间断点出现了吉布斯现象之外，其他基本一致；当f0>时，此时信号的有效部分仍在区间[-0.25,0.25]内，可恢复出原有信号。

## 三、任务三

1. 利用任务二中的周期方波信号并画出

运行结果：

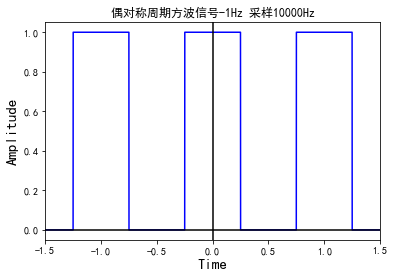


图 12周期方波信号

1. 取当n∈[-5, 5]时频谱的傅里叶级数反变换求取信号，画出结果，显示至少二个周期

*# 取n在[-5, 5]***def** x\_new(f0, t):  
 res = 0  
 **for** i **in** range(-5, 5):  
 **if** i != 0:  
 res = res + math.sin(0.5\*math.pi\*i\*f0)/(math.pi\*i\*f0) \* math.cos(2\*math.pi\*i\*f0\*t)  
 **else**:  
 res = res + 0.5  
 **return** res\*f0  
  
X = []  
t = [i/1000 **for** i **in** range(-2000,2000)]  
f0 = 1  
**for** i **in** t:  
 X.append(x\_new(f0, i))  
  
plt.plot(t, X, label=**"all"**, c=**"b"**)  
plt.xlim(-1.5, 1.5)  
plt.axhline(y=0, ls=**"-"**, c=**"k"**)  
plt.axvline(x=0, color=**'k'**)  
plt.title(**u'非周期方波信号 恢复信号图 [-5,5]'**)  
plt.ylabel(**u'x'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.xlabel(**u't'**, fontdict={**'weight'**:**'normal'**,**'size'**: 15})  
plt.ylim(-0.5, 1.5)  
plt.show()

运行结果：

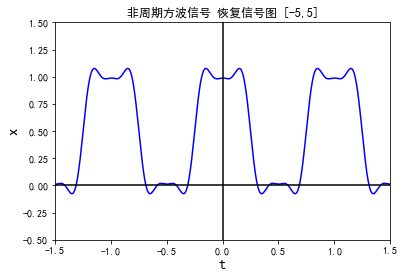


图 13 n=5 的反变换

1. 再分别取不同n=20、100时的频谱利用傅里叶级数反变换求取相应的信号，并画出

|  |  |
| --- | --- |
|  |  |

图 14 n=20、100 的反变换

1. 比较求得信号与原信号，分析说明结果

吉布斯现象（又叫吉布斯效应）：将具有不连续点的周期函数（如矩形脉冲)进行傅立叶级数展开后，选取有限项进行合成。当选取的项数越多，在所合成的波形中出现的峰起越靠近原信号的不连续点。