**Problem Set 5**

**Research method Problem Set 5 due Wed 5th Dec, 23:59**

**LIU BOWEN (1004028)**

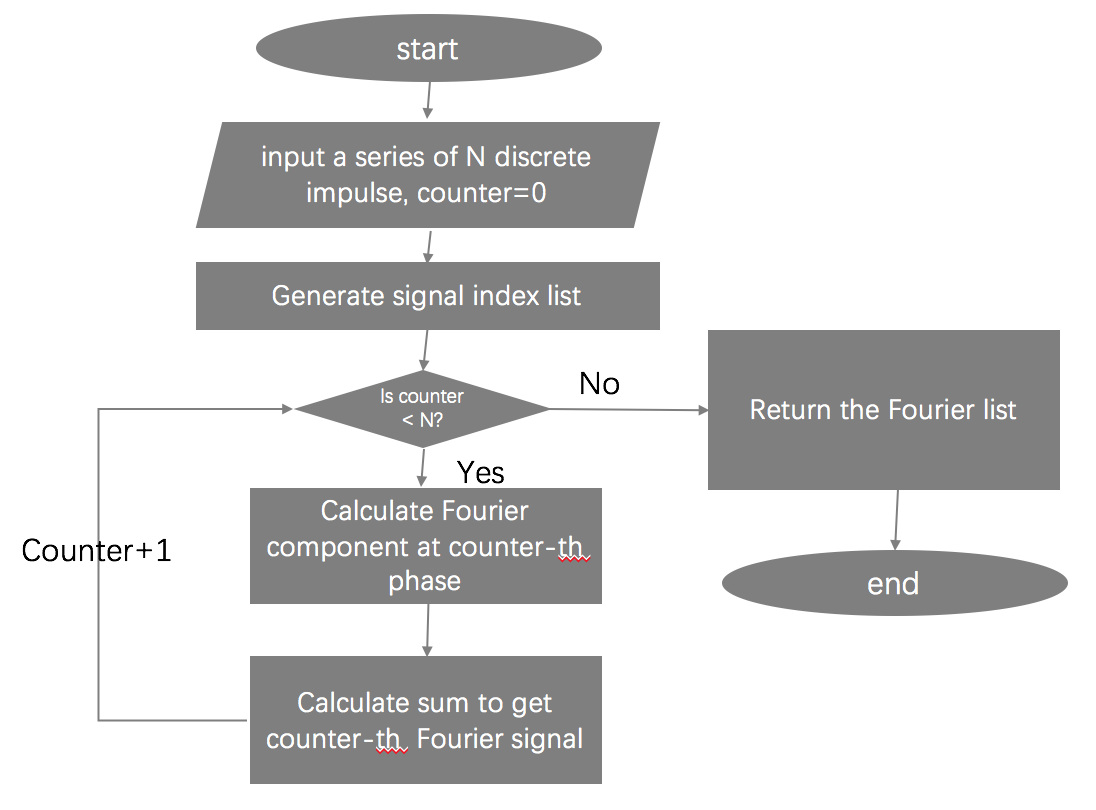
For Problem Set 5, I use the following packages:

|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np  import scipy as sp  import scipy.stats as ss  from scipy.fftpack import fft, ifft |

**Problem 1**

Answer:

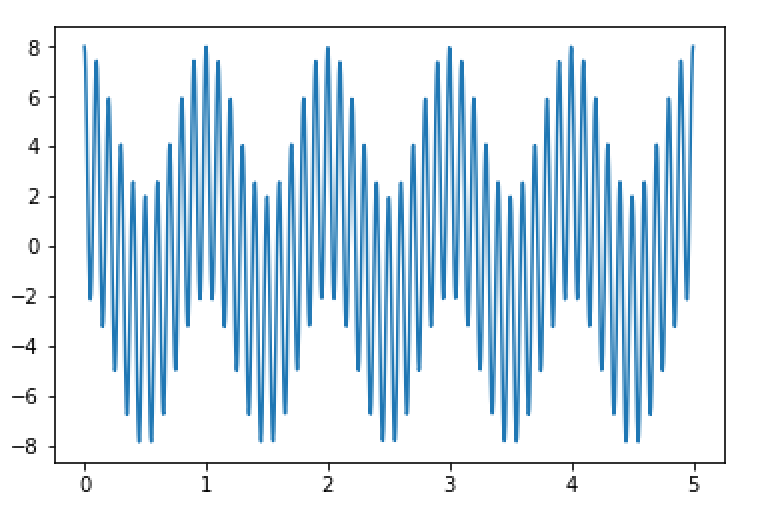
a)



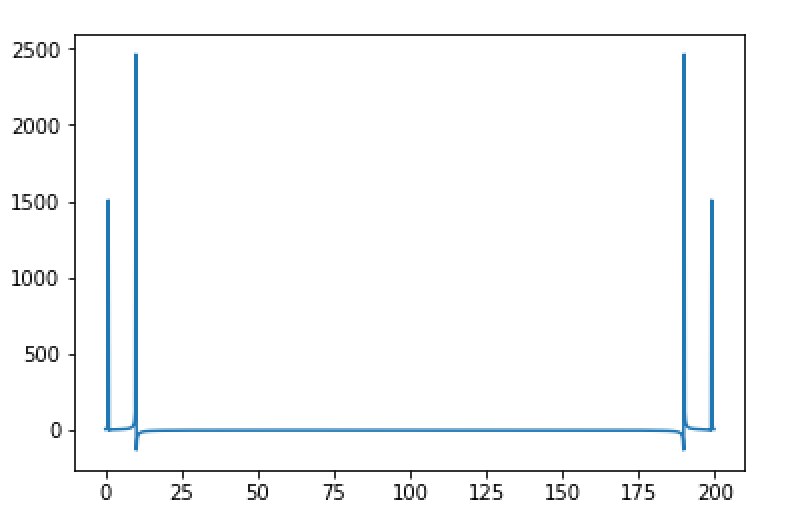
b)

|  |
| --- |
| sample\_num=1000  x=np.linspace(0,5,sample\_num)  y=5\*np.cos((20\*np.pi)\*x) + 3\*np.cos((2\*np.pi)\*x)  Maxtime=5.  timestep=Maxtime/sample\_num  FreqStep =1./( Maxtime )  w=np.fft.fft(y)  plt.figure()  plt.plot(x,y) |

I draw 5 seconds plot and the figure is shown in :



c)

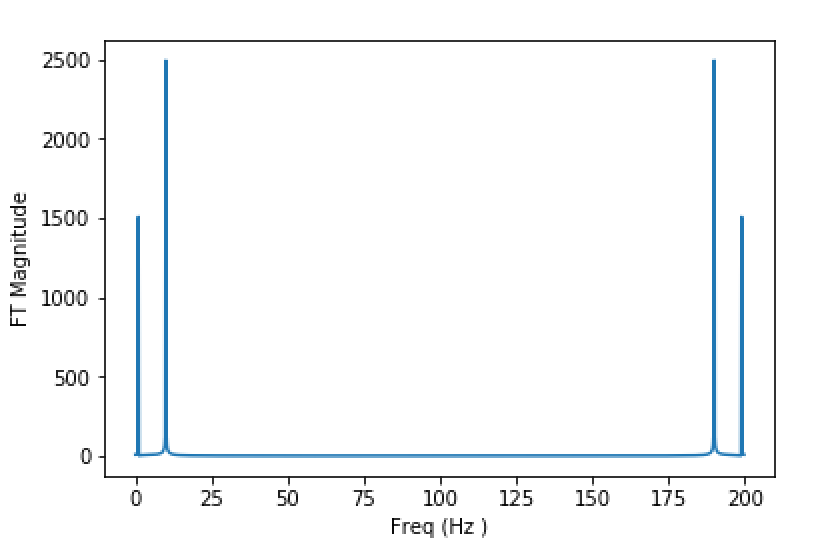


|  |
| --- |
| sample\_num=1000  x=np.linspace(0,5,sample\_num)  y=5\*np.cos((20\*np.pi)\*x) + 3\*np.cos((2\*np.pi)\*x)  Maxtime=5.  timestep=Maxtime/sample\_num  FreqStep =1./( Maxtime )  freq =[]  for i in range(sample\_num):  freq.append(i\*FreqStep)  def nativeFFT(x):  size = len(x)  samples = np.arange(0, size)  res = []  for i in samples:  res.append(sum(np.exp(-samples \* i \* 2.0 \* np.pi \* 1j/size) \* y))  return res  ab = (nativeFFT(y))  plt.plot(freq, ab) |

d)

|  |
| --- |
| sample\_num=1000  x=np.linspace(0,5,sample\_num)  y=5\*np.cos((20\*np.pi)\*x) + 3\*np.cos((2\*np.pi)\*x)  Maxtime=5.  timestep=Maxtime/sample\_num  FreqStep =1./( Maxtime )  freq =[]  for i in range(sample\_num):  freq.append(i\*FreqStep)  w= np.fft.fft(y)  mag=[]  for line in w:  mag.append(np.linalg.norm(line))  plt.figure()  plt.plot(freq , mag)  plt.xlabel('Freq (Hz )')  plt.ylabel('FT Magnitude') |

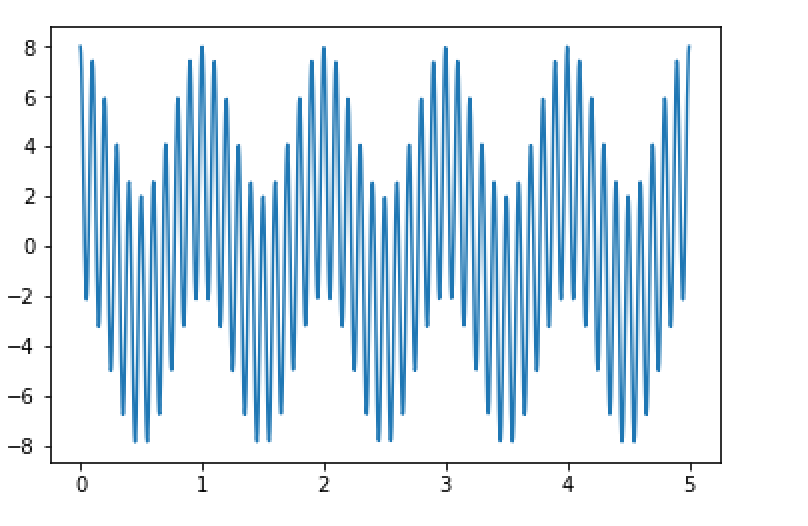
The result is :



e)

|  |
| --- |
| from scipy.fftpack import fft, ifft  plt.figure()  plt.plot(x, ifft(w)) |

The result is:

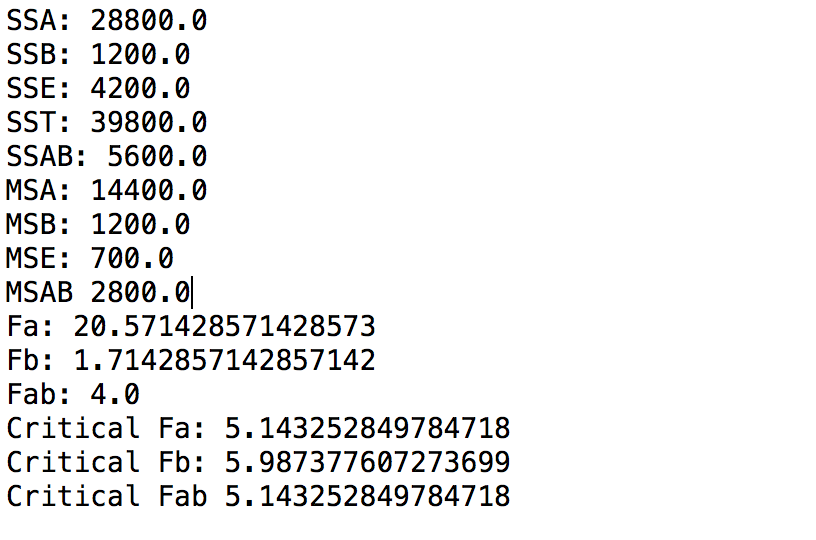


**Problem 2**

Answer:

|  |
| --- |
| def twoFactor():  data = [[[100,140],[180,140]],[[230,210],[160,200]],[[310,270],[210,250]]]  size\_data = len(data)  size\_data\_row = len(data[0])  size\_data\_unit = len(data[0][0])  DoF1 = size\_data-1  DoF2 = size\_data\_row-1  DoF3 = (size\_data-1)\*(size\_data\_row-1)  DoF4 = size\_data \* size\_data\_row \* (size\_data\_unit-1)    mean1 = []  for i in range(size\_data):  mean1.append(np.mean(data[i]))    mean2 = []  for i in range(size\_data\_row):  tmp = []  for j in range(size\_data):  tmp.append(data[j][i])  mean2.append(np.mean(tmp))    SSA = size\_data\_row \* size\_data\_unit \* np.var(mean1) \* len(mean1)  SSB = size\_data \* size\_data\_unit \* np.var(mean2) \* len(mean2)  sse\_list = []  for j in range(size\_data\_row):  for i in range(size\_data):  sse\_list.append(len(data[i][j])\*np.var(data[i][j]))  SSE = sum(sse\_list)  SST = np.var(data) \* size\_data \* size\_data\_row \* size\_data\_unit  SSAB = SST – SSA – SSB - SSE    print 'SSA:',SSA  print 'SSB:',SSB  print 'SSE:',SSE  print 'SST:',SST  print 'SSAB:',SSAB    MSA = SSA/(size\_data-1.0)  MSB = SSB/(size\_data\_row-1.0)  MSAB = SSAB/((size\_data-1.0) \* (size\_data\_row-1.0))  MSE = SSE/(size\_data \* size\_data\_row \* (size\_data\_unit-1))  print 'MSA:',MSA  print 'MSB:',MSB  print 'MSE:',MSE  print 'MSAB',MSAB  Fa = MSA/MSE  Fb = MSB/MSE  Fab = MSAB/MSE  Fac = ss.f.ppf(0.95, DoF1, DoF4)  Fbc = ss.f.ppf(0.95, DoF2, DoF4)  Fabc = ss.f.ppf(0.95, DoF3, DoF4)  print 'Fa:', Fa  print 'Fb:', Fb  print 'Fab:', Fab  print 'Critical Fa:', Fac  print 'Critical Fb:', Fbc  print 'Critical Fab', Fabc  twoFactor() |

The result is :



From the results, only Fa > critical Fa so factor A would be significant.

**Problem 3**

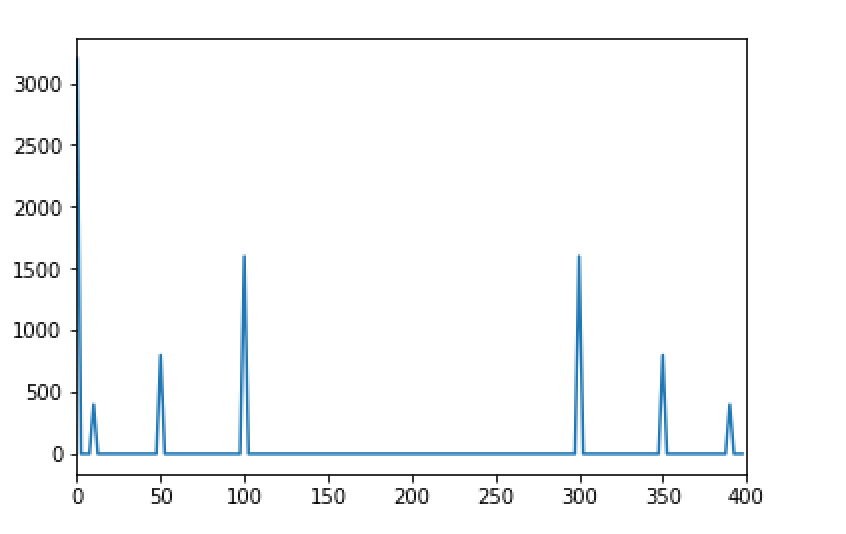
Answer:

The max frequency is 100 and the question need to plot 0.4s so the twice minimum sample point is 100\*0.4\*2\*2 = 160

a)

|  |
| --- |
| Ns= 160 #number of samples  time = np.linspace(0, 0.4, Ns, endpoint = False)  y=20 + 10\*np.cos((2\*np.pi)\*50\*time) + 5\*np.cos((2\*np.pi)\*10\*time) + 20\*np.cos((2\*np.pi)\*100\*time)  Maxtime = 0.4  timestep = Maxtime/Ns  FreqStep =1./( Maxtime )  freq =[]  for i in range(Ns):  freq.append (i\*FreqStep)  w = np.fft.fft(y)  plt.figure()  plt.plot(freq ,w)  plt.xlim (0, 400) |

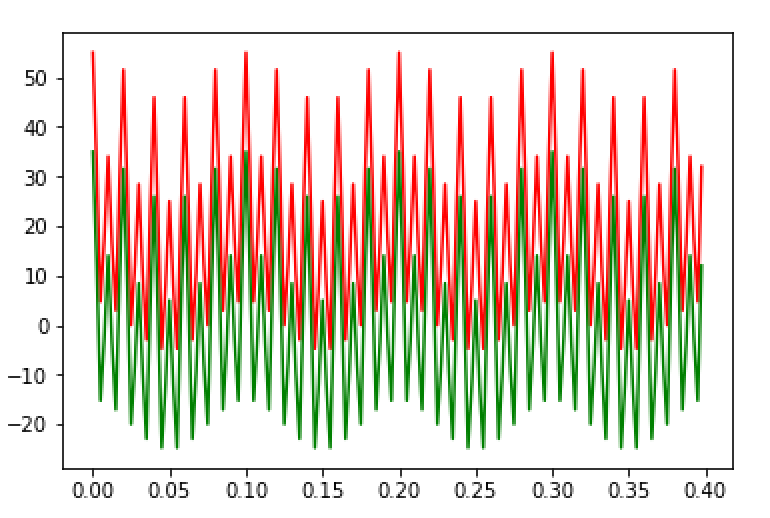
The result is below and there are 7 Fourier components (including 4 components presented at lower Hz and other higher 3 components are symmetry components).



b)

|  |
| --- |
| def window(freq, spec, f):  F\_spec=spec  F\_spec[f]=0  return F\_spec  filtered0 = window(freq, w, 0)  plt.figure()  plt.plot(freq, filtered0)  plt.xlim (0 ,400)  plt.figure()  plt.plot(time,y, 'r')  F\_spec= np.fft.ifft(filtered0)  plt.plot(time, F\_spec, 'g') |

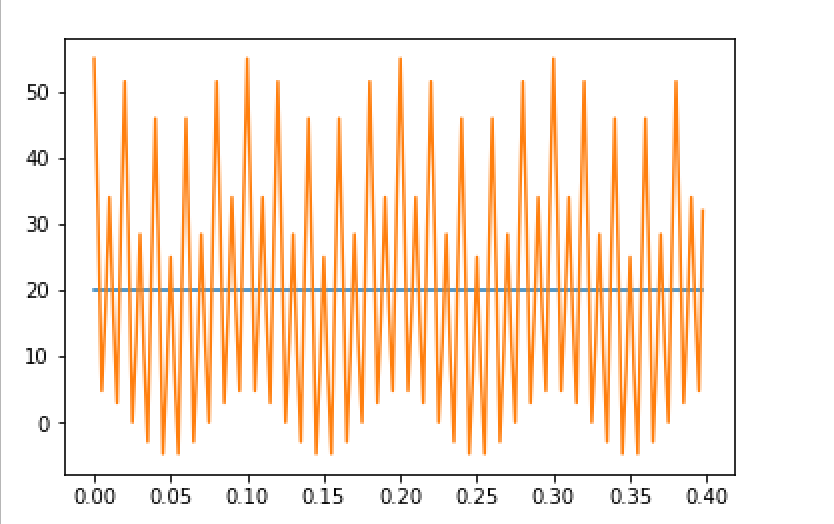
window() function can remove the 0 hz from the frequency spectrum. Then, I use ifft to replot against f() and the red one is before and green one is after.



c)

|  |
| --- |
| Ns= 160 #number of samples  time = np.linspace(0, 0.4, 160, endpoint = False)  y=20 + 10\*np.cos((2\*np.pi)\*50\*time) + 5\*np.cos((2\*np.pi)\*10\*time) + 20\*np.cos((2\*np.pi)\*100\*time)  FreqStep = 1./0.4  freq = np.arange(0,len(time)\*FreqStep,FreqStep)  f = [sum(np.exp(-time\*i\*2\*np.pi\*1j)\*y) for i in freq]  y\_removeTime = np.fft.ifft(f \* (freq==0))  plt.figure()  plt.plot(time, y\_removeTime)  plt.plot(time, y) |

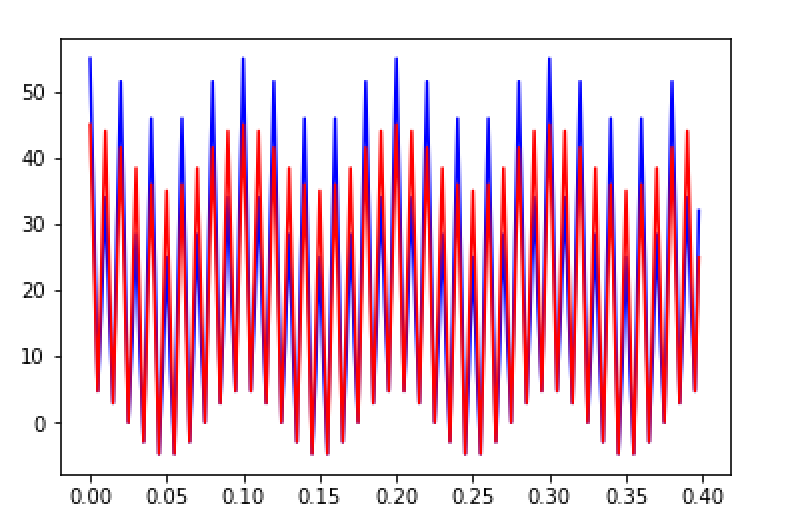
The result is shown below, the orange one is for previous f(t) and the blue straight line is for removing time varying signal.



d)

|  |
| --- |
| def window(freq, spec, f0):  indx0 = freq.index(f0)  F\_spec=spec  F\_spec[indx0]=0  indx0 = freq.index(f0)  indx02 = len(freq) - indx0  F\_spec[indx02] = 0  return F\_spec  filtered50 = window (freq, w, 50)  plt.figure()  plt.plot(time, y, 'b')  F\_spec=np.fft.ifft(filtered50)  plt.plot(time, F\_spec, 'r') |

The result is shown below, the blue one is for previous figure and the read one is after removing 50 Hz, and if I need to remove 50 Hz and 350 Hz also needs to be removed.



e) If removing the below 100 Hz, the range above 300 Hz also needs to be removed.

|  |
| --- |
| def window(freq, spec, f0):  indx0 = freq.index(f0)  indx02 = len(freq) - indx0  F\_spec=spec  F\_spec[0: int( indx0 )]=0  F\_spec[indx02+1: len(freq)]=0  return F\_spec  filtered50 = window (freq, w, 100)  plt.figure()  plt.plot(time,y)  F\_spec=np.fft.ifft(filtered50)  plt.plot(time, (F\_spec)) |

The result is below and the blue one is before and orange one is after.

