Notes on Fourier Transform

Paul Hanakata
Deparment of Physics, Harvard University, USA

(Dated: September 30, 2022)

I. FOURIER TRANSFORM

1. Introduction

In these notes we will learn about: 1. The basic mathematics of Fourier transform 2. How to use Fourier analysis using a scientific package such as scikitlearn 3. Implement simple 1D Fourier transform

Methods based on Fourier transform are applied in many disciplines, from signal-processing to finances. So what is Fourier transform? Before goinng deeper, imagine if you have some functions, say a polynomial $f(x) = ax^3 + x + bx^4$, an exponential e^x, or a sinusoidal function $\cos(x)$.

In class discussion (make a group of two/three) and discuss the following questions: 1. What make these three class functions similar or different? 2. Write e^x and $\cos(x)$ as a series of $x, x^2, x^3, \cdots$.

So what we have done so far? We wrote functions in a polynomial form, i.e., using polynomial functions as the basis. You already can see that it's easy to write any polynamial functions of degree n in terms of x, x^2 and so on because these are the natural basis. On the other hand, we need to write an infinite series for a simple periodic function like $\cos x$. In the similar manner, we can write the functions discussed above in terms $\cos(nx)$ and $\sin(nx)$, with n being interger, and this is known as Fourier series:

$$f(x) = \sum_n A_n \cos(nx) + B_n \sin(nx)$$

Fourier series is a powerful tool to analyze data as now we can write any arbitray (complicated) function in terms of familiar periodic functions. Moreover, as you shall see, the amplitudes tell us about the nature of the functions.

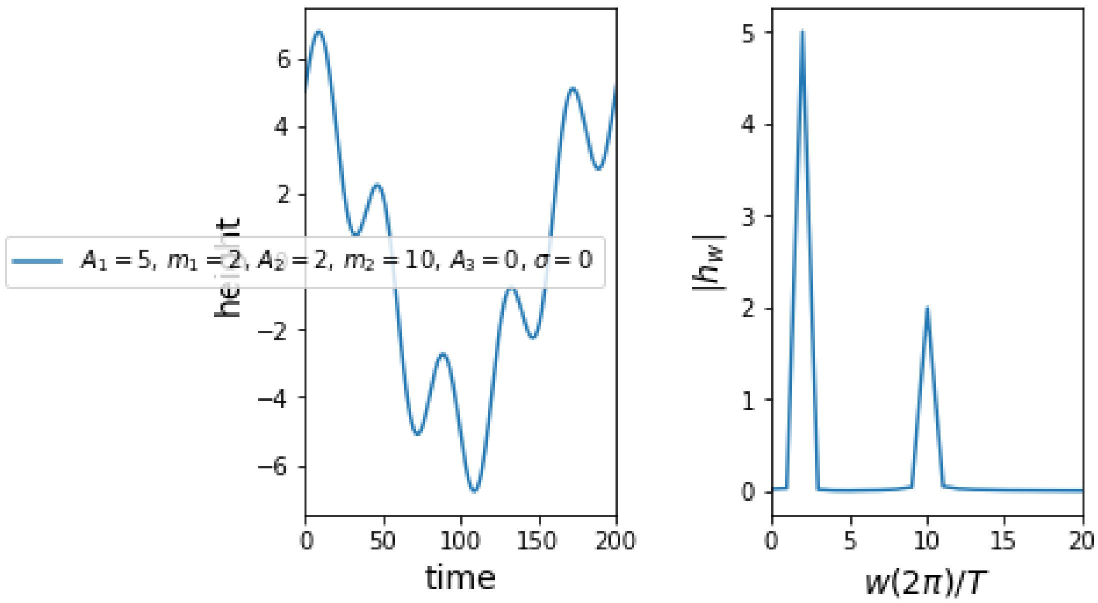
2. Using Fourier transform to analyze data

Imagine you're analyzing a time series of signals of experimental observations of a graphene ribbon moving up and down or price movents of a stock in a day. For simplicity let's create an "artficial" data describing the up/down movements. The height is composed of sin functions and a noise term

$$h(t) = A_1 \sin\left(\frac{m_1 * 2\pi t}{T}\right) + A_2 \sin\left(\frac{m_2 * 2\pi t}{T}\right) + A_3\xi(t),$$

where A_1, A_2 are the amplitudes of sinoduidal functions and A_3 is the amplitude of the noise. Next we generate space data points between $t = [0, T]$ with $T = 400$ and increament time step of $\Delta T = 1$. We first start with zero noise ($A_3 = 0$) and set $A_1 = 1, m_1 = 10, A_2 = 0.2$, and $m_2 = 70$

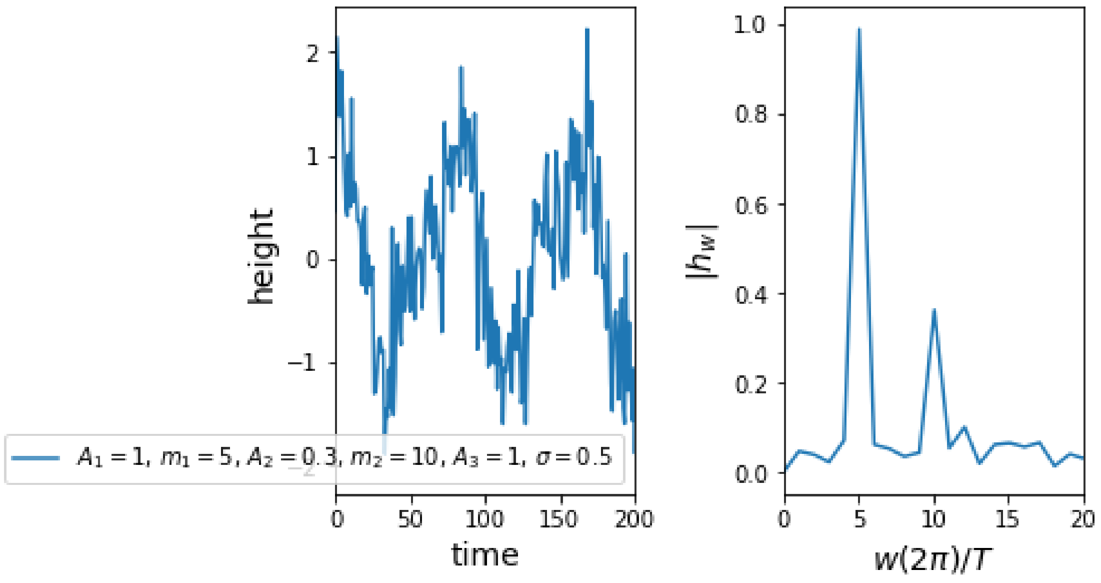
```
In [572]: fft_sin_functions_with_noise(A_1=5, m_1=2, A_2=2, m_2=10, A_3=0, sigma=0)
```



We can see that the absolute amplitude in frequency space peaks at $w = 10$ and $w = 75$. Note that w is in units of $2\pi/T$.

Next we add *weak* (Gaussian) noise with $\sigma = 0.5$ and use smaller m_1, m_2 such that $A_1 = 1, m_1 = 5, A_2 = 0.2, m_2 = 10, A_3 = 1.0, \sigma = 0.5$

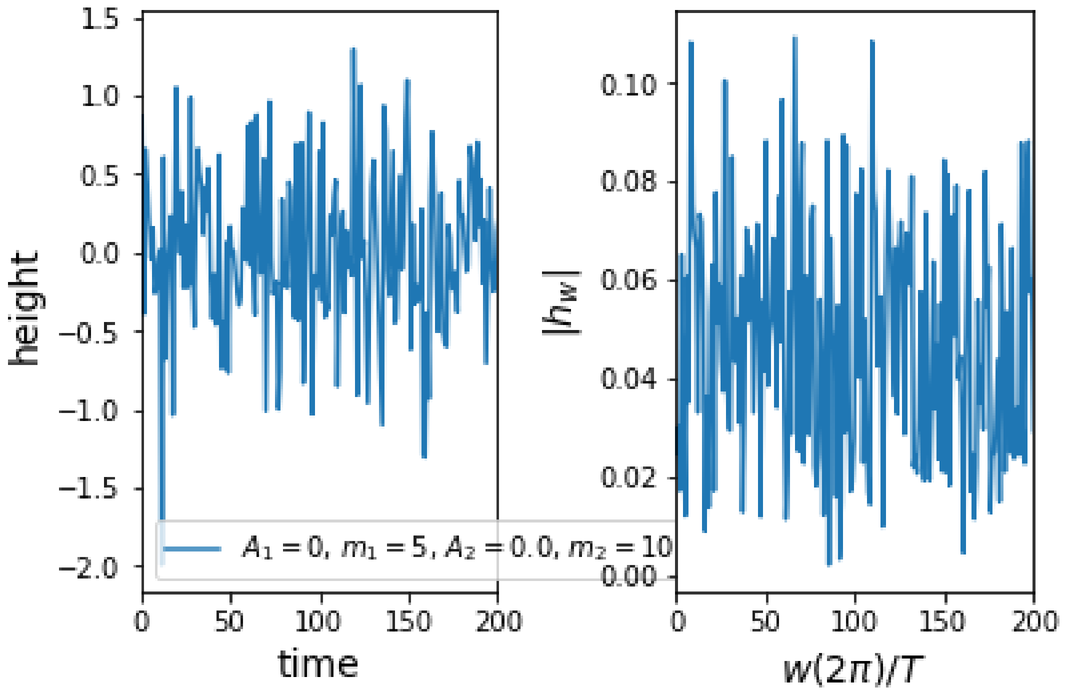
```
In [573]: fft_sin_functions_with_noise(A_1=1, m_1=5, A_2=0.3, m_2=10, A_3=1, sigma=0.5)
```



We can see that the absolute amplitude in frequency space peaks at $w = 5$ and $w = 10$, as expected.

Finally we test Fourier transform a case where we only have noise and no sin functions. We get roughly a flat spectrum as we Fourier transform a white noise.

```
In [335]: fft_sin_functions_with_noise(A_1=0, m_1=5, A_2=0., m_2=100, A_3=1, sigma=0.5)
```



II. HOMEWORK (DUE IN TWO WEEKS)

A. Using fft from scikitlearn to analyze real data

1. Analyze the height movemment of buckled ribbon obtained from molecular dynamics simulation.
2. Download one day stock price movement (take NVDA or APPL) and write what you learn about the stock price movement
3. Use your own code to analyze these data and check the results match

B. Code developements (group project)

1. We often dealing with high-dimensional data. Write discreete 2D Fourier analysis with $N \times N$ square grids.
2. Perform inverse Fourier transform and compare the inverse (reconstructured) results with the real data. How does choice of N affect the reconsrtcution quality?
3. Now peform the same analysis but with different grids.

text hahalalalalalallalalal

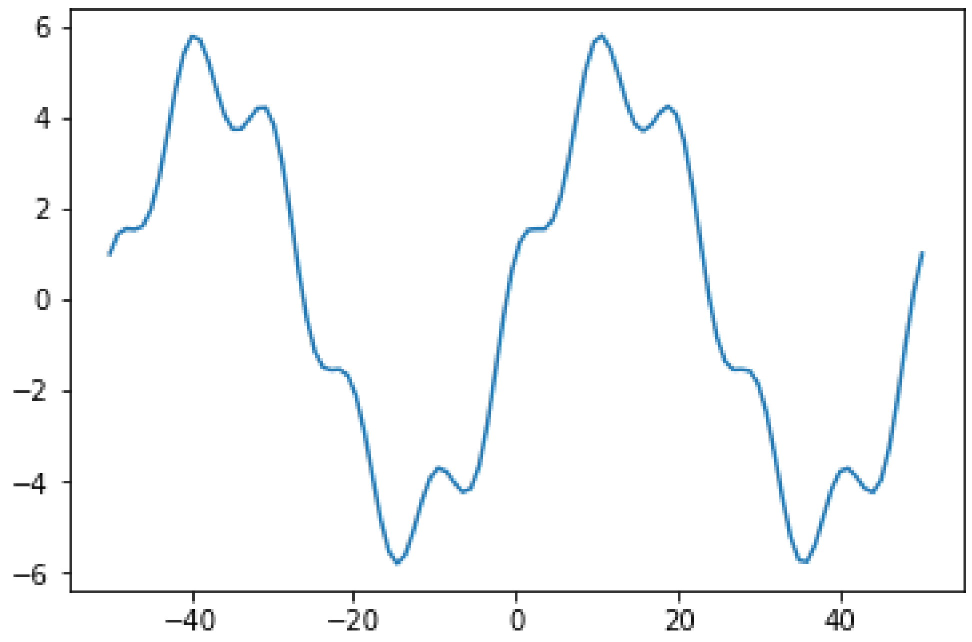
```
In [ ]:
```

```
In [273]:  #!/usr/bin/env python -W ignore::DeprecationWarning
          import numpy as np
          import matplotlib.pyplot as plt
          import scipy.fftpack
```

```
fontsize=14
def fft_sin_functions_with_noise(A_1, m_1, A_2, m_2, A_3, sigma):
    #A_1=1
    #m_1=10
    #A_2=.2
    #m_2=75
    #A_3=0
    fs=14
    # Number of samplepoints
    N = 400 #this is T
    # sample spacing
    dT = 1.0/N
    x = np.linspace(0.0, int(N*dT), N) #rescaling x from 0 to 1
    y =  A_1*np.cos(m_1* 2.0*np.pi*x) +A_2*np.sin(m_2 * 2.0*np.pi*x)
+A_3*np.random.normal(0, sigma, len(x))

    #plt.figure(1)
    plt.subplot(121)
    #plt.subplot(211)


    #plt.plot(t, 2*s1)

    plt.plot(x*N, y)
    plt.legend([r'$A_1=$'+str(A_1)+r', $m_1=$'+str(m_1)+r', $A_2=$'+str(A_2)+r',
$m_2=$'+str(m_2)+r', $A_3=$'+str(A_3)+r', $\sigma=$'+str(sigma)])
    plt.ylabel('height', fontsize=fs)
    plt.xlabel('time', fontsize=fs)
    #plt.ylim([-2, 2])
    plt.xlim([0, 200])

    #plt.show()
    #y = 0.2*np.sin(50.0 * 2.0*np.pi*x) + 1*np.sin(100 * 2.0*np.pi*x)
    #print (x.shape, y.shape)

    #y = np.exp(-x*100)
    yf = scipy.fftpack.fft(y)
    xf = np.linspace(0.0, 1.0/(2.0*dT), int(N/2))
    #plt.plot(t, s1)
    plt.subplot(122)
    #plt.subplot(212)

    #fig, ax = plt.subplots()
    plt.plot(xf, 2.0/N * np.abs(yf[:N//2]))
    plt.ylabel(r'$|h_w|$', fontsize=fs)
    plt.xlabel(r'$w (2\pi)/T$', fontsize=fs)
    plt.xlim([0, 2*m_2])
    plt.subplots_adjust(wspace=.5)
    plt.show()
```

In [509]:
```
N=100
dT = 1.0/N
sigma=.1
x = (np.linspace(0, int(N*dT), N)-0.5)*N #rescaling x from -N/2 to N/2
A_1=5
m_1=2
A_2=1
```

```
m_2=10
A_3=0

#y =  A_1*(1+np.sin(2.0*np.pi*x/N))/2+A_2*np.cos(m_2 * 2.0*np.pi*x/N)
+A_3*np.random.normal(0, sigma, len(x))
y =  A_1*np.sin(m_1*2.0*np.pi*x/N)+A_2*np.cos(m_2 * 2.0*np.pi*x/N)

plt.plot(x,y)
plt.show()
```



```
In [163]: y[1]

Out[163]: 0.0010066617640577813

In [564]: def fourierDiscrete(x, y):
              QTOT=N+1
              fr =  np.zeros(QTOT)
              fi = np.zeros(QTOT)
              f2 = np.zeros(QTOT)
              qq = np.zeros(QTOT)
              delta_q =2*np.pi/N
              countq=0
              shiftL=N/2.
              #for i in range(0, int(N/2)):
              for i in range(0, QTOT):
                  #print(i)
                  q = i*delta_q
                  qq[countq] = q
                  for j in range (len(y)):
                      fr[countq]+=y[j]*np.cos(q*(x[j]+shiftL))
                      fi[countq]+=y[j]*np.sin(q*(x[j]+shiftL))
```

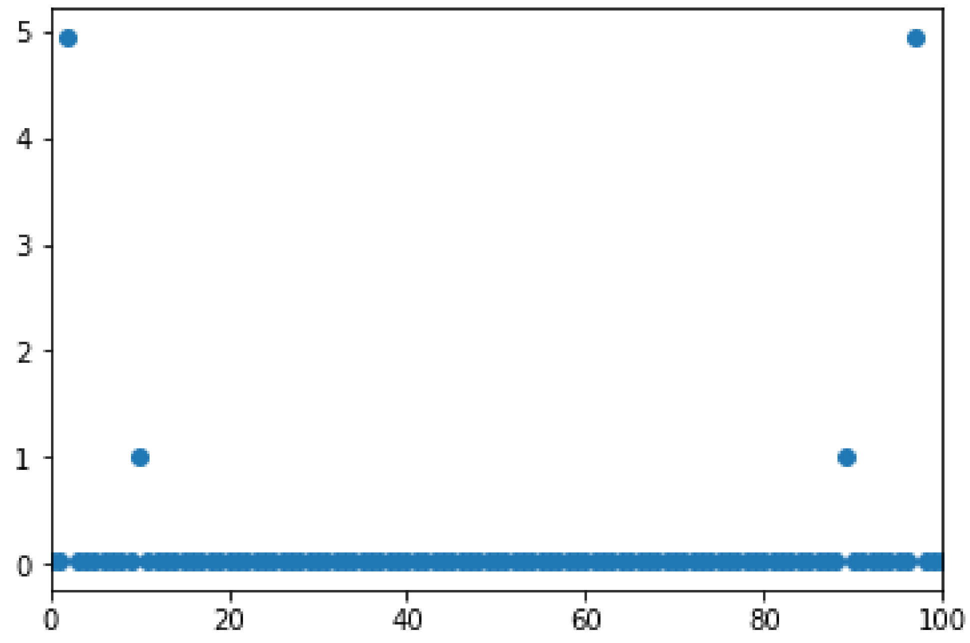
```
            #rescale fr and fi by number of sample points
            #fr=fr
            #fi=fi

            f2[countq]=(fr[countq]*fr[countq]+fi[countq]*fi[countq])
            countq+=1
        #print(countq)
        return fr/len(y), fi/len(y), f2/(len(y))**2, qq
```

```
In [565]: fr, fi, f2, qq = fourierDiscrete(x, y)
```

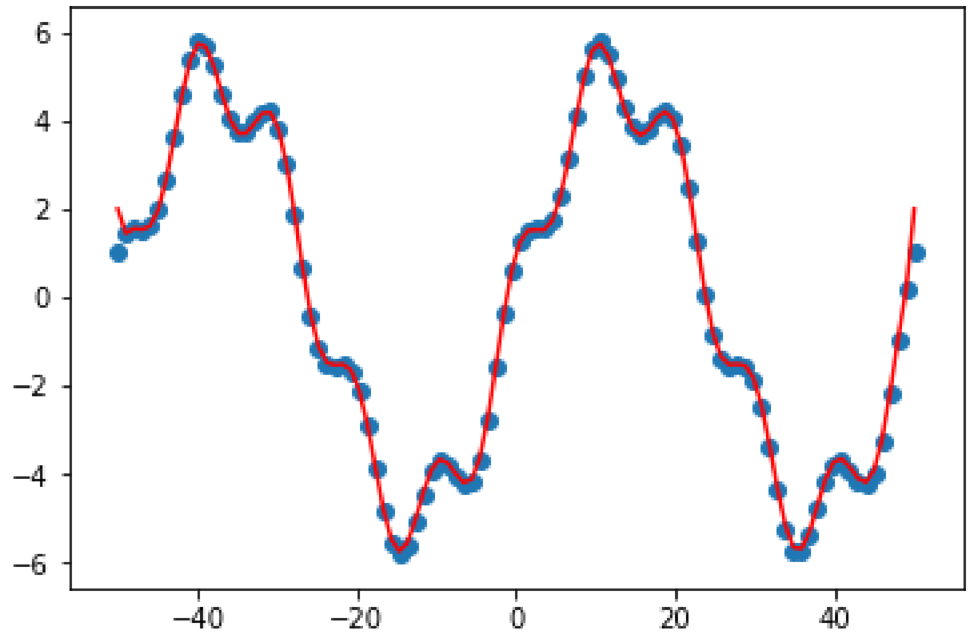
```
In [566]: qq
```

```
Out[566]: array([0.        , 0.06283185, 0.12566371, 0.18849556, 0.25132741,
                 0.31415927, 0.37699112, 0.43982297, 0.50265482, 0.56548668,
                 0.62831853, 0.69115038, 0.75398224, 0.81681409, 0.87964594,
                 0.9424778 , 1.00530965, 1.0681415 , 1.13097336, 1.19380521,
                 1.25663706, 1.31946891, 1.38230077, 1.44513262, 1.50796447,
                 1.57079633, 1.63362818, 1.69646003, 1.75929189, 1.82212374,
                 1.88495559, 1.94778745, 2.0106193 , 2.07345115, 2.136283  ,
                 2.19911486, 2.26194671, 2.32477856, 2.38761042, 2.45044227,
                 2.51327412, 2.57610598, 2.63893783, 2.70176968, 2.76460154,
                 2.82743339, 2.89026524, 2.95309709, 3.01592895, 3.0787608 ,
                 3.14159265, 3.20442451, 3.26725636, 3.33008821, 3.39292007,
                 3.45575192, 3.51858377, 3.58141563, 3.64424748, 3.70707933,
                 3.76991118, 3.83274304, 3.89557489, 3.95840674, 4.0212386 ,
                 4.08407045, 4.1469023 , 4.20973416, 4.27256601, 4.33539786,
                 4.39822972, 4.46106157, 4.52389342, 4.58672527, 4.64955713,
                 4.71238898, 4.77522083, 4.83805269, 4.90088454, 4.96371639,
                 5.02654825, 5.0893801 , 5.15221195, 5.2150438 , 5.27787566,
                 5.34070751, 5.40353936, 5.46637122, 5.52920307, 5.59203492,
                 5.65486678, 5.71769863, 5.78053048, 5.84336234, 5.90619419,
                 5.96902604, 6.03185789, 6.09468975, 6.1575216 , 6.22035345,
                 6.28318531])
```

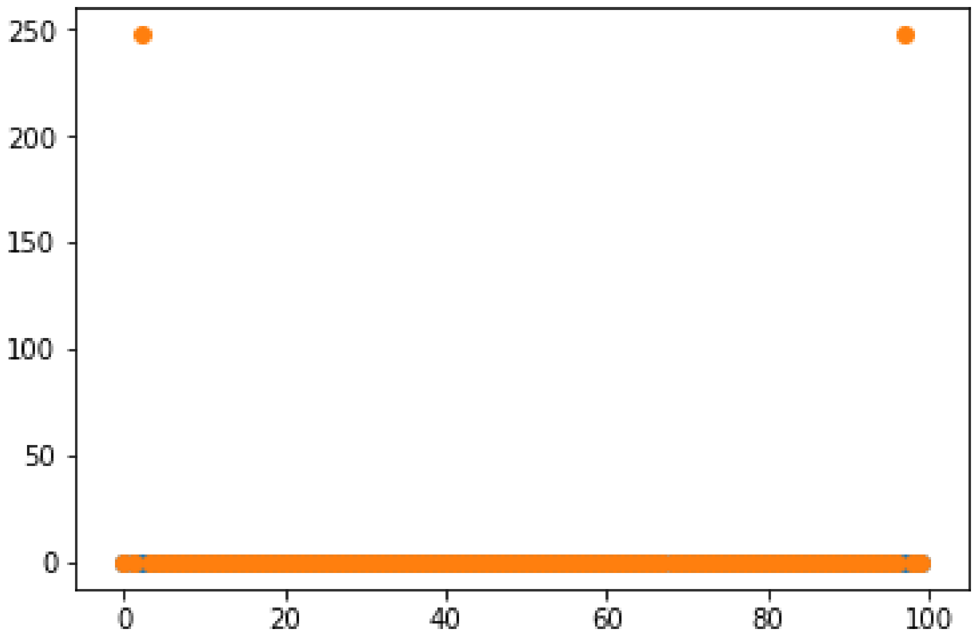
```
In [569]: #since we only use 1/2 q-space in this plot we multiply abs|fq| by 2 due to symmtey !
          plt.scatter(qq/2/np.pi*100, 2*np.sqrt(f2))
          plt.xlim([0, 100])
          plt.show()
```



```
In [556]: f_in_posSpace = inverseFourier(fr, fi)
          plt.plot(x, f_in_posSpace, color='r')
          plt.scatter(x, y)
          plt.show()
```



```
In [382]: plt.scatter(qq/2/np.pi*100, fr)
          plt.scatter(qq/2/np.pi*100, fi)

          plt.show()
```



```
In [560]: #fx = np.zeros(N)
          def inverseFourier(fr, fi):
              f_in_posSpace= np.zeros(N)
              for i in range (len(fx)):
                  for j in range(len(fr)):
                      #fx[i] += fr[j]*np.cos(qq[j]*x[i])/N #+ fi[i]*np.cos(qq[i]*x[i])
                      #since we only use 1/2 q-space (symmetry) we need to multiply by 2!
                      f_in_posSpace[i] += fr[j]*np.cos(qq[j]*x[i])+fi[j]*np.sin(qq[j]*x[i])
              return f_in_posSpace
```

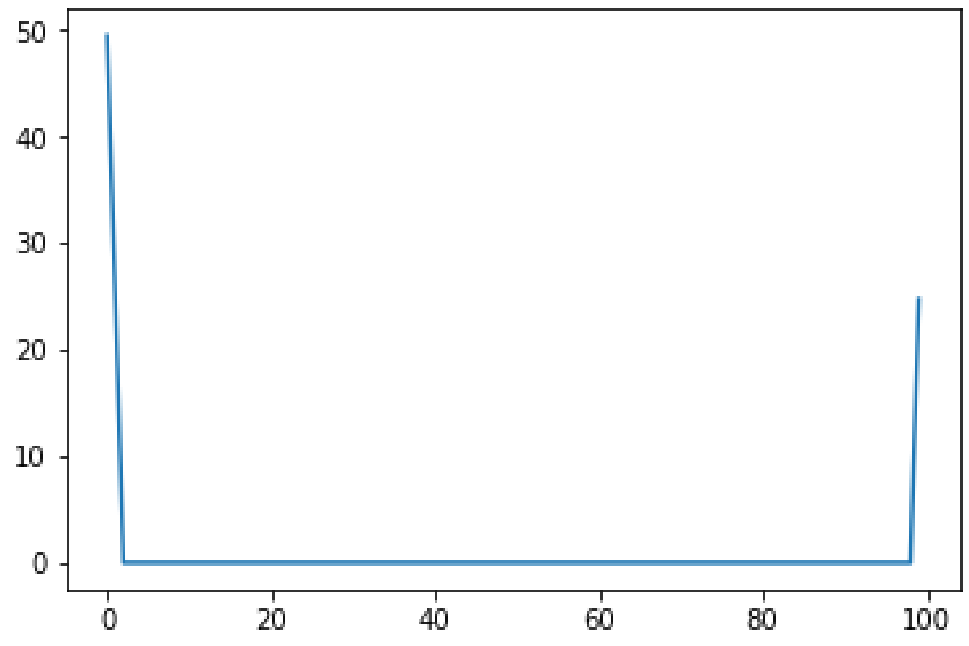
```
In [146]: fr*np.cos(qq*x)
```

```
Out[146]: array([ 7.06209326e-15, -1.04125024e-03,  2.61654779e-14,  1.15282851e-03,
                  9.03304347e-15, -1.02608469e-03,  1.14500253e-14,  1.97761591e-04,
                 -1.04708701e-14,  1.36656349e-03, -2.02241444e-14, -1.82475801e-03,
                 -8.24223105e-15, -1.06321586e-03,  3.10004415e-14,  2.38588934e-03,
                 -3.10433132e-16,  3.03219487e-03, -3.13831266e-14,  6.68504550e-04,
                 -2.19701971e-16, -2.37201338e-03,  1.66043226e-14, -4.85502499e-03,
                  3.02577840e-15, -6.74814844e-03,  2.26908717e-15, -7.87959483e-03,
                  1.80596754e-15, -6.28866117e-03, -6.54262909e-15,  2.93454565e-03,
                 -1.14660235e-14,  2.25230617e-02,  8.01741343e-17,  3.09670928e-02,
                 -9.05300328e-15, -2.52538391e-02,  5.63389751e-16, -8.48147063e-02,
                 -2.92010840e-15,  1.37460674e-01, -7.43836430e-16,  5.06568889e-02,
                 -2.27510683e-15, -9.07262499e-01, -5.02760556e-16,  8.16650131e+00,
                  2.46379301e+01,  4.20116167e+01,  4.95000000e+01,  4.19693146e+01,
                  2.44391125e+01,  7.94120041e+00, -4.68420271e-16, -7.71653882e-01,
                 -1.48822912e-15, -3.80373655e-02, -1.25975629e-15,  1.65449815e-01,
                 -2.89059360e-15, -6.29517819e-02, -8.19620590e-17, -4.44062450e-02,
                 -9.31667844e-15,  1.74200370e-02,  2.52660775e-15,  2.60041179e-02,
                 -5.66197717e-15,  1.28167751e-02, -6.01623463e-15,  2.59726772e-03,
                  2.35752319e-15, -1.31987801e-03,  3.84114346e-15, -1.53391992e-03,
                  5.17438099e-15,  5.51106991e-18,  2.12837010e-14,  2.22483692e-03,
                 -1.61932119e-16,  3.84865668e-03,  5.91882865e-16,  3.03219487e-03,
                 -1.73909367e-14, -4.62785386e-04,  2.11580204e-14, -2.47097295e-03,
                  1.93045972e-14,  3.98761657e-04, -8.46955233e-15,  1.50547166e-03,
                 -3.18693923e-14, -1.47627177e-03, -1.75424565e-14,  7.34030534e-04,
                 -3.27204817e-15, -1.87431926e-04, -1.25066668e-15,  6.12477959e-18])
```

```
In [70]: np.sum(fr), np.sum(fi)
```

```
Out[70]: (98.99999999999997, 3.3662309051329954e-14)
```

```
In [68]: plt.plot(fr)
         plt.show()
```



```
In [147]: https://github.com/markjay4k/fourier-transform/blob/master/fourier%20transform.ipynb
```

```
      File "<ipython-input-147-cb6e3490169d>", line 1
    https://github.com/markjay4k/fourier-transform/blob/master/fourier%20transform.ipynb
         ^
SyntaxError: invalid syntax
```

```
In [ ]:
```