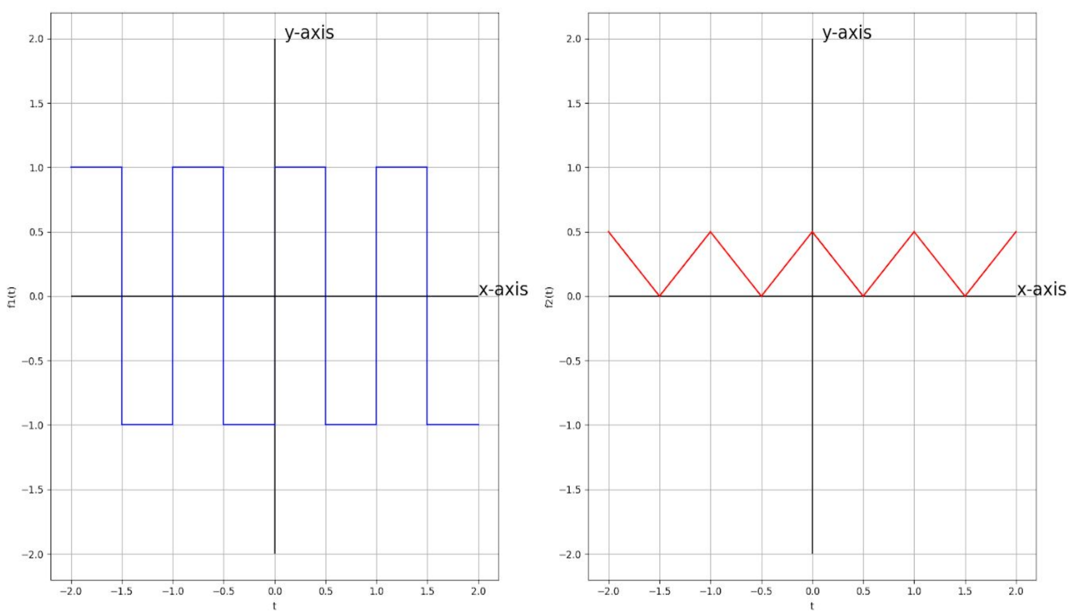
Solution for Q.2:

```
# Q.2_(a).py  (Part (a) in Question.2) :

import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import cmath

w, h = 10, 8;
Matrix = [[0 for x in range(w)] for y in range(h)]
output1 = [[0 for x in range(w)] for y in range(h)]
output2 = [[0 for x in range(w)] for y in range(h)]
for i in range(h):
	for j in range(w):
		Matrix[i][j] = ((i/2) -2) + j/18
		if(i%2 == 0):
			output1[i][j] = 1
			output2[i][j] = Matrix[i][0] + 1/2 - Matrix[i][j]
		else:
			output1[i][j] = -1
			output2[i][j] = Matrix[i][j] - Matrix[i][0]

subplot(1,2,1)
for i in range(8):
	plt.plot(Matrix[i],output1[i],'b')
List = [-3/2,-1,-1/2,0,1/2,1,3/2]
for i in range(7):
	plt.vlines(x=List[i], ymin=-1, ymax=1.0, color='b')
plt.hlines(y=0,xmin=-2,xmax=2,color='k')
plt.vlines(x=0,ymin=-2,ymax=2,color='k')
plt.text(2,0.004,'x-axis',fontsize=18)
plt.text(0.09,2,'y-axis',fontsize=18)

plt.grid()
plt.xlabel('t')
plt.ylabel('f1(t)')
subplot(1,2,2)
for i in range(8):
```

```
        plt.plot(Matrix[i],output2[i],'r')
plt.hlines(y=0,xmin=-2,xmax=2,color='k')
plt.vlines(x=0,ymin=-2,ymax=2,color='k')
plt.text(2,0.004,'x-axis',fontsize=18)
plt.text(0.09,2,'y-axis',fontsize=18)

plt.grid()
plt.xlabel('t')
plt.ylabel('f2(t)')
plt.show()
```

Figure Obtained after running the code:



##Answer of Q2_(b):

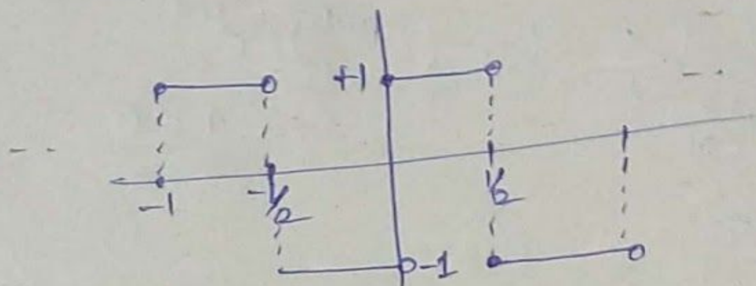
(1) f1(t) is discontinous and f2(t) is continuous.(By observing the plot obtained after running the code.)

(2) f1(t) is odd function (f1(t) = -f1(-t)) and f2(t) is even function (f2(t) = f2(-t)).(By observing the plot obtained after running the code.)

(3) For Half wave symmetry,f(t-L) = -f(t) where Period of the function i.e T = 2L . So by taking this condition into mind,f1(t) posseses HALF WAVE SYMMETRY and f2(t) doesnot posses HALF WAVE SYMMETRY.(in our case, the independent variable is t.)

(4) For Quarter wave symmetry,f(t) must be half wave symmetric and it must be symmetric about the midpoint of either positive or negative half cycle.So by taking this condition into mind,f1(t) doesnot posses QUARTER WAVE SYMMETRY and also f2(t) doesnot posses QUARTER WAVE SYMMETRY.i.e. both functions f1(t) and f2(t) donot posses QUARTER WAVE SYMMETRY. (in our case, the independent variable is t.)

Q.2 (e) for function $f_1(t) \Rightarrow$



Now we know that $f_1(t)$ can be written as -

$$f_1(t) = \sum_n C_n e^{j\omega_0 n t}$$

Now ω_0 = fundamental frequency $= \frac{2\pi}{1} = 2\pi$ rad/s as T = 1 sec. (In SI system.)

$$\therefore f_1(t) = \sum_n C_n e^{j2\pi n t}$$

Now $$C_n = \frac{1}{T_0}\int_0^{T_0} f_1(t) \cdot e^{-jn\omega_0 t} dt$$

$$\therefore C_n = \frac{1}{1}\left[\int_0^{1/2} 1\, e^{-j2\pi n t} dt + \int_{1/2}^{1} (-1) e^{-j2\pi n t} dt\right]$$

$$\therefore C_n = \int_0^{1/2} e^{-j2\pi n t} dt - \int_{1/2}^{1} e^{-j2\pi n t} dt$$

$$C_n = \left.\frac{e^{-j2\pi n t}}{-j2\pi n}\right|_0^{1/2} - \left.\left(\frac{e^{-j2\pi n t}}{-j2\pi n}\right)\right|_{1/2}^{1}$$

$$C_n = \frac{1 - \cos\pi n}{j\pi n}$$

Hence when n = even, $C_n = 0$ and when n = odd, $C_n = \frac{2}{j\pi n}$

i.e $C_n = \begin{cases} 0 & \text{if } n = \text{even} \\ -\frac{2j}{\pi n} & \text{if } n = \text{odd} \end{cases}$

$\therefore f_1(t) = \sum_n C_n e^{j2\pi nt}$

$\therefore f_1(t) = \sum_{n=1,3,5}^{N} \left(C_n e^{j2\pi nt} + C_{-n} e^{-j2\pi nt} \right)$

Now $\left(C_n e^{j2\pi nt} + C_{-n} e^{-j2\pi nt} \right)$ for $n = \text{odd}$ is given by –

$$-\frac{2j}{\pi n} e^{j2\pi nt} + \frac{2j}{\pi n} e^{-j2\pi nt} = \frac{2j}{\pi n} \left(-2j \sin(2\pi nt) \right)$$

$$= \frac{4}{\pi n} \sin(2\pi nt)$$

Hence $f_1(t) = \sum_{n=1,3,5,..}^{N} \left(\frac{4}{\pi n} \right) \sin(2\pi nt)$

(for sufficiently large value of N)

Hence $a_K = 0$ for both $K = \text{even}$ or odd

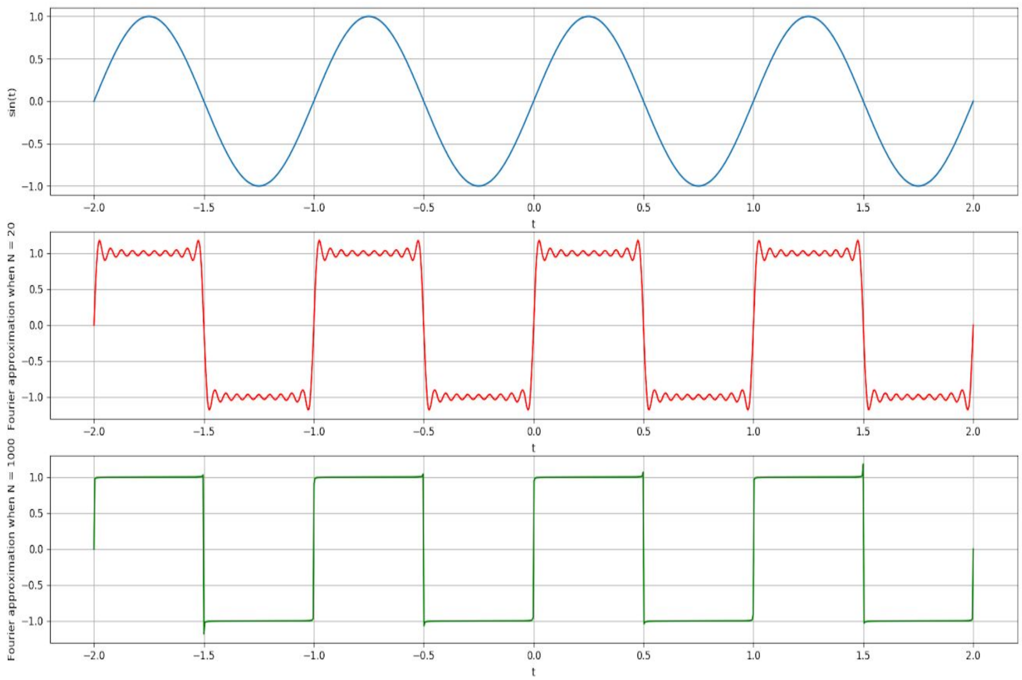
and $b_K = \begin{cases} \frac{4}{\pi K} & \text{if } K = \text{odd.} \\ 0 & \text{if } K = \text{even.} \end{cases}$

Q.2_(e)_approximating_f1(t)_using_FOURIER_SERIES.py (Part (e) in Question.2) :

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import cmath
t = np.linspace(-2,2,1000)

def Fourier_cal(N):
        y = 0
        z = complex(0,1)
        for k in range(1,N,2):
                y = y + (4/(np.pi*k))*np.sin(2*np.pi*k*t)
        return y
y1 = np.sin(2*np.pi*t)
y2 = Fourier_cal(20)
y3 = Fourier_cal(1000)

subplot(3,1,1)
plt.plot(t,y1)
plt.xlabel('t')
plt.ylabel('sin(t)')
plt.grid()

subplot(3,1,2)
plt.plot(t,y2,'r')
plt.xlabel('t')
plt.ylabel('Fourier approximation when N = 20')
plt.grid()

subplot(3,1,3)
plt.plot(t,y3,'g')
plt.grid()
plt.xlabel('t')
plt.ylabel('Fourier approximation when N = 1000')
plt.show()
```

#From above, we can say that [ak = 0 (for both k = even or k = odd)] and [(bk = 4/(pi*k) if k = odd) and (bk = 0 if k = even.)] ---> for f1(t)

Figure Obtained after running the code:



Q.2 (e) for function $f_2(t)$:-

$$f_2(t) = \begin{cases} \frac{1}{2}+t \,; & -\frac{1}{2} \le t < 0 \\ \frac{1}{2}-t \,; & 0 \le t < \frac{1}{2} \end{cases}$$

$T = 1\,sec$

$f_2(t) = \sum_K C_K e^{jK\omega_0 t}$ Here also $\omega_0 = \frac{2\pi}{1} = 2\pi$ rad/s (In SI System)

$$C_K = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} f_2(t)\, e^{-jK\omega_0 t}\, dt$$

$$\therefore \quad C_K = \int_{-1/2}^{0}\left(\frac{1}{2}+t\right)e^{-j2\pi Kt}\,dt + \int_{0}^{1/2}\left(\frac{1}{2}-t\right)e^{-j2\pi Kt}\,dt$$

$$C_K = \left[\left(\frac{1}{2}+t\right)\frac{e^{-j2\pi Kt}}{(-j2\pi K)}\bigg|_{-1/2}^{0} - \int_{-1/2}^{0}\frac{e^{-j2\pi Kt}}{-j2\pi K}\,dt\right]$$

$$+ \left(\frac{1}{2}-t\right)\frac{e^{-j2\pi Kt}}{-j2\pi K} - \int_{0}^{1/2} -\frac{e^{-j2\pi Kt}}{-j2\pi K}\,dt$$

After solving the above integrals,

we get $C_K = \frac{1}{2\pi^2 K^2}(1-\cos\pi K)$ for $K \neq 0$

If $K=0$, then

$$C_0 = \int_{-1/2}^{0}\left(\frac{1}{2}+t\right)dt + \int_{0}^{1/2}\left(\frac{1}{2}-t\right)dt = \frac{1}{4}$$

$$\therefore \quad \boxed{C_K = \begin{cases} 0 & \cancel{K = \text{even}}\ K \in 2n\ \forall\, n \in \mathbb{Z}-\{0\} \\ \frac{1}{4}\,, & K = 0 \\ \frac{1}{\pi^2 K^2} & K = 2n+1\ \forall\, n \in \mathbb{Z} \text{ i.e } K = \text{odd} \end{cases}}$$

Set of Integers

$$\therefore f_2(t) = \sum_{K} C_K e^{jK2\pi t}$$

$$f_2(t) = C_0 + \sum_{K=1,3,5} \left(C_K e^{jK2\pi t} + C_{-K} e^{-jK2\pi t} \right)$$

Now $C_K e^{jK2\pi t} + C_{-K} e^{-jK2\pi t}$ for $K = \text{odd}$ is given by:-

$$\Rightarrow \frac{1}{\pi^2 k^2} \left(e^{j2\pi k t} + e^{-j2\pi k t} \right) = \frac{2\cos(2\pi k t)}{\pi^2 k^2}$$

$$\therefore f_2(t) = \frac{1}{4} + \sum_{n=1,3,5}^{N} \left(\frac{2}{\pi^2 k^2} \right) \cos(2\pi n t)$$

Hence $a_K = \begin{cases} 1/4 \; ; & K = 0 \\ \frac{2}{\pi^2 k^2} \; ; & K = 2n+1 \;\; \forall n \in \mathbb{Z} \\ 0 \; ; & K = 2n \;\; \forall n \in \mathbb{Z} - \{0\} \end{cases}$

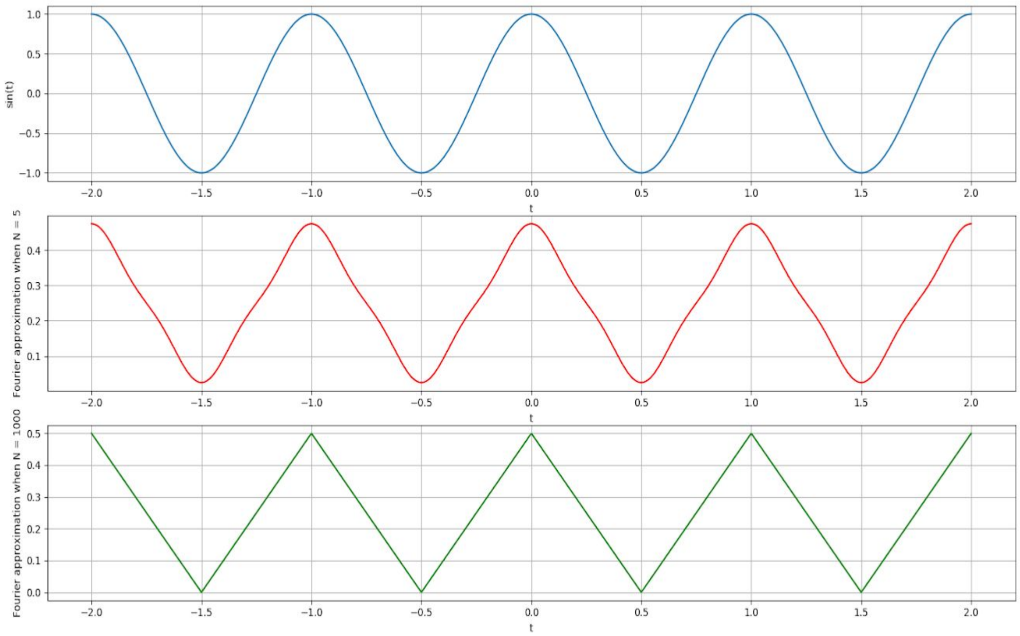
and $b_K = 0$ for all $K \in \mathbb{Z}$

Q.2_(e)_approximating_f2(t)_using_FOURIER_SERIES.py (Part (e) in Question.2) :

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from pylab import *
import cmath
t = np.linspace(-2,2,1000)

def Fourier_cal(N):
	y = 0
	z = complex(0,1)
	for n in range(1,N,2):
		y = y + (2/(((np.pi)*n)**2))*np.cos(2*np.pi*n*t)
	return (y + 1/4) #because Co = 1/4
y1 = np.cos(2*np.pi*t)
y2 = Fourier_cal(5)
y3 = Fourier_cal(1000)

subplot(3,1,1)
plt.plot(t,y1)
plt.xlabel('t')
plt.ylabel('sin(t)')
plt.grid()

subplot(3,1,2)
plt.plot(t,y2,’r’)
plt.xlabel('t')
plt.ylabel('Fourier approximation when N = 5')
plt.grid()

subplot(3,1,3)
plt.plot(t,y3,’g’)
plt.grid()
plt.xlabel('t')
plt.ylabel('Fourier approximation when N = 1000')

plt.show()

#  Here [(ak = 2/((pi^2)*(k^2)) if k = odd) and (ak = 1/4 for k = 0) , (otherwise for all even values of k ,ak = 0.)]
#  now bk = 0 for all k.
```

Figure Obtained after running the code:



Q.2_(f) (Part (f) in Question.2) :

(1) for f1(t) , [Ck = 0 if k = even and Ck = -2j/(pi*k) if k = odd.]
(2) for f2(t) , [(Ck = 1/((pi^2)*(k^2)) if k = odd) . And (C0 = 1/4) (except k =0 ,for all even values of k , Ck = 0).]

Q.2_(g) (Part (g) in Question.2) :

(1)

Fourier series converge for f1(t) and f2(t).because if we look at the plots obtained by running the codes (Q.2_(e)_approximating_f1(t)_using_FOURIER_SERIES.py) and (Q.2_(e)_approximating_f2(t)_using_FOURIER_SERIES.py),we observe that after a significant value of N , the plots are not changing much.for e.g, after running the code (Q.2_(e)_approximating_f2(t)_using_FOURIER_SERIES.py),we observe that after N

=1000,the plot of fourier approximation of f2(t) doesnot change much.In fact we see that it is same[Similar thing can be observed after running (Q.2_(e)_approximating_f1(t)_using_FOURIER_SERIES.py)].This means that for both f1(t) and f2(t),fourier series converges(for both functions).

(2)

Also there is a theorem that states a sufficient condition for the convergence of a given Fourier series. It also tells us to what value does the Fourier series converge to at each point on the real line.

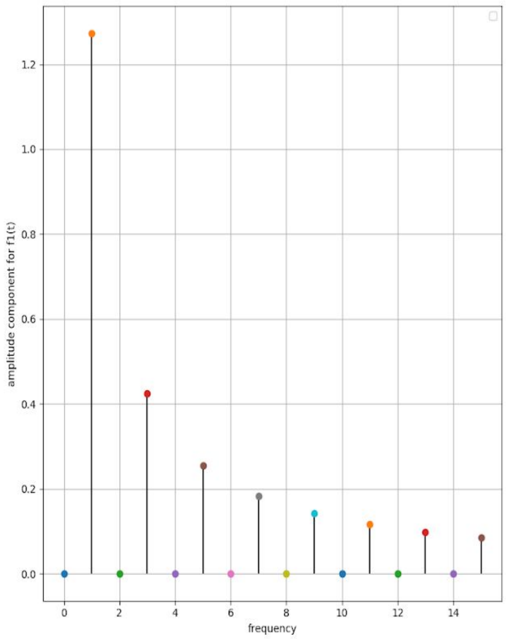
#-->Theorem: Suppose f and f ′ are piecewise continuous on the interval −L ≤ x ≤ L. Further, suppose that f is defined elsewhere so that it is periodic with period 2L. Then f has a Fourier series as stated previously whose coefficients are given by the Euler-Fourier formulas. The Fourier series converge to f (x) at all points where f is continuous, and to [((lim x --> c-)f(x) + (lim x --> c+)f(x))/2] at every point c where f is discontinuous.

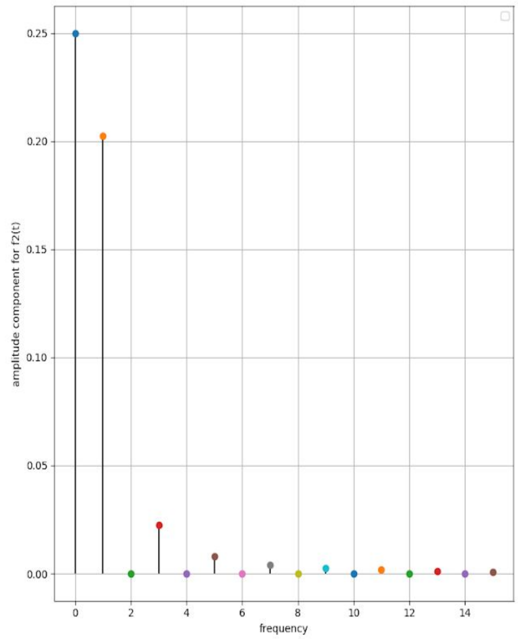
Q.2_(h).py (Part (h) in Question.2) :

```
import numpy as np
import matplotlib.pyplot as plt
import cmath
from pylab import *
##FOR f1:
ak = [0]*16
for k in range(0,16):
	if(k%2 == 0):
		ak[k] = 0
	else:
		ak[k] = 4/(np.pi*k)
#Above represents the amplitude components of f1(t).
frequency = [0]*16
for i in range(0,16):
	frequency[i] = i  #Represents the frequency harmonics of f1(t).
subplot(1,2,1)
for xc in frequency:
	plt.plot(xc,ak[xc],'o')
```

```
	plt.vlines(xc, ymin=0, ymax=ak[xc])#plotting the amplitude components v/s
frequency harmonics.


plt.xlabel('frequency')
plt.ylabel('amplitude component for f1(t)')
plt.legend()
plt.grid()
#
##############################################################################
###################

##FOR f2:

ak1 = [0]*16
for k in range(0,16):
	if(k%2 == 0):
		ak1[k] = 0
	else:
		ak1[k] = 2/(((np.pi)**2)*(k**2))
ak1[0] = 1/4 #As ak = 1/4 for k = 0.
#Above represents the amplitude components of f2(t).
frequency1 = [0]*16
for i1 in range(0,16):
	frequency1[i1] = i1 #Represents the frequency harmonics of f2(t).
subplot(1,2,2)
for xc1 in frequency1:
	plt.plot(xc1,ak1[xc1],'o')
	plt.vlines(xc1, ymin=0, ymax=ak1[xc1])#plotting the amplitude components v/s
frequency harmonics.

plt.xlabel('frequency')
plt.ylabel('amplitude component for f2(t)')
plt.legend()
plt.grid()
plt.show()
```

Figure Obtained after running the code:





Q.2_(i) (Part (i) in Question.2) :

(1) There is a theorem that states a sufficient condition for the convergence of a given Fourier series. It also tells us to what value does the Fourier series converge to at each point on the real line.

#-->Theorem: Suppose f and f ′ are piecewise continuous on the interval −L ≤ x ≤ L. Further, suppose that f is defined elsewhere so that it is periodic with period 2L. Then f has a Fourier series as stated previously whose coefficients are given by the Euler-Fourier formulas. The Fourier series converge to f (x) at all points where f is continuous, and to ((lim x --> c-)f(x) + (lim x --> c+)f(x))/2 at every point c where f is discontinuous.

(2) A consequence of this theorem is that the Fourier series of f will “fill in” any removable discontinuity the original function might have. A Fourier series will not have any removable-type discontinuity.

(3)So,if we look at the fourier series coefficients of f1(t) [f1(t) is odd function] and f2(t) [f2(t) is even function],then we observe that their respective fourier series have higher order Fourier coefficients.