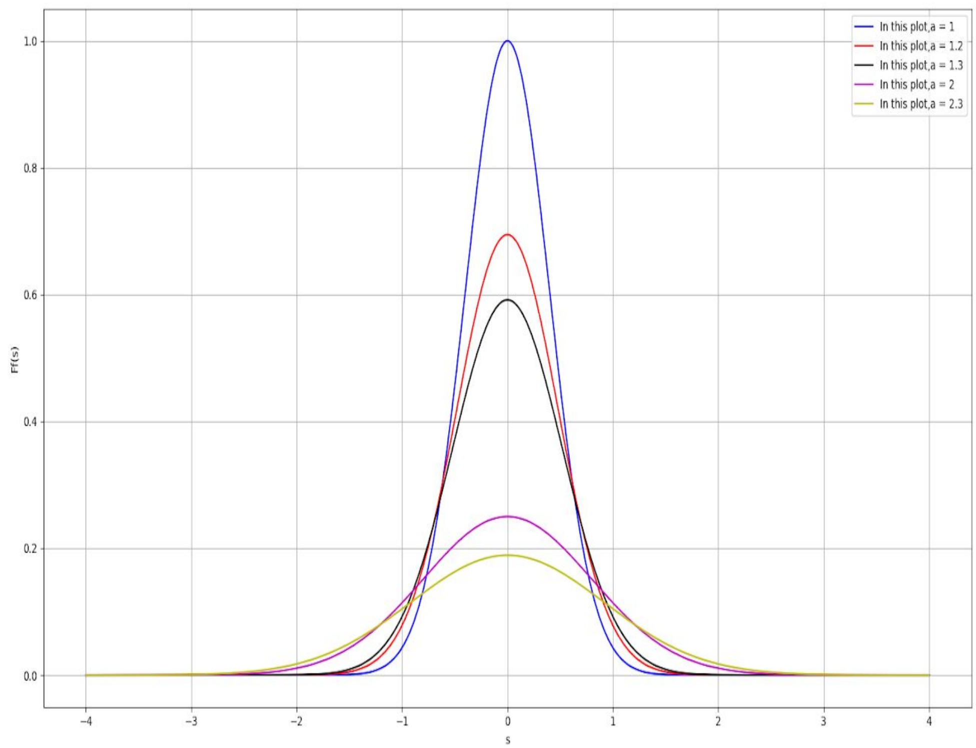
Solution for Q.3:

Q.3_(a).py (Part (a) in Question.3) :

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
s= np.linspace(-4,4,1000)

def FourierTransform(a):
        Ff = (1/(a**2))*np.exp((-np.pi*(s**2))/(a**2))
        return Ff
Ff1 = FourierTransform(1)
Ff2 = FourierTransform(1.2)
Ff3 = FourierTransform(1.3)
Ff4 = FourierTransform(2)
Ff5 = FourierTransform(2.3)

plt.plot(s,Ff1,'b',label='In this plot,a = 1')
plt.plot(s,Ff2,'r',label='In this plot,a = 1.2')
plt.plot(s,Ff3,'k',label='In this plot,a = 1.3')
plt.plot(s,Ff4,'m',label='In this plot,a = 2')
plt.plot(s,Ff5,'y',label='In this plot,a = 2.3')
plt.legend()
plt.ylabel('Ff(s)')
plt.xlabel('s')
plt.grid()
plt.show()
```

Figure Obtained after running the code:



(Part (b),(c),(d),(e) in Question.3) :

(b) We are given that $f(t) = e^{-a^2 \pi t^2}$

In this part, we have to assume $a = 1$.

Sampling rate R_{time} is picked to construct the sequence $\{f(nR_{time})\}$ for $n = -\infty$ to $+\infty$

Let $p(t) = \sum_n \delta(t - nR_{time})$, $f_p(t) = f(t) \cdot p(t) \Rightarrow$ Contains set of impulses at integral multiples of R_{time}.

$$\therefore f_p(t) = \sum_n f(nR_{time}) \cdot \delta(t - nR_{time})$$

Now if we want to calculate the fourier transform of $f_p(t)$ i.e $F_p(j\Omega)$

$$\therefore F_p(j\Omega) = \int_{-\infty}^{+\infty} f_p(t) \cdot e^{-j\Omega t} \, dt = \int_{-\infty}^{+\infty} \sum_n f(nR_{time}) \cdot \delta(t - nR_{time}) \, e^{-j\Omega t} \, dt$$

Hence, $\underbrace{F_p(j\Omega)}_{\text{in terms of } \Omega} = \sum_n f(nR_{time}) \int_{-\infty}^{+\infty} \delta(t - nR_{time}) \cdot e^{-j\Omega t} \, dt = \sum_n f(nR_{time}) \, e^{-j\Omega (nR_{time})}$

Now if we have $f(t)$, then in the problem it is given that –

$$\underbrace{Ff(s)}_{\text{in terms of } s} = \int_{-\infty}^{+\infty} f(t) \cdot e^{-2\pi j s t} \, dt \qquad \left(Ff(s) \equiv \text{Fourier transform of } f(t) \right)$$

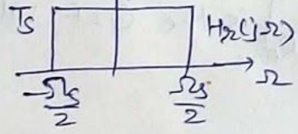
Hence, $Ff_p(s) = \int_{-\infty}^{+\infty} f_p(t) \cdot e^{-j2\pi s t} \, dt = \sum_n f(nR_{time}) \cdot e^{-j(2\pi s) nR_{time}}$

Basically, $F_p(j\Omega)$ i.e the fourier transform of $f_p(t)$ (in terms of Ω) $\Rightarrow$

$$F_p(j\Omega) = \frac{1}{T_s} \sum_K F(j(\Omega - K\Omega_s)) \qquad \text{where } \Omega_s = \frac{2\pi}{T_s} \text{ and } T_s = \text{Sampling period.}$$

If $f(t)$ is bounded, then :–

So if $\Omega_s > 2\Omega_N$, then we can get the $F(j\Omega)$ from $F_p(j\Omega)$ by just applying window function in frequency domain



So we also can get $f(t)$ from $f_p(t)$.

So If we are considering only $\{f(nR_{time})\}$ samples, then

$$\boxed{F_{fp}(s) = \sum_{n=-\infty}^{+\infty} f(nR_{time})\, e^{-j(2\pi s)(nR_{time})}}$$

But if we want $F_f(s)$ back from $F_{fp}(s)$ provided $\Omega_s > 2\Omega_N$ ie Sampling frequency is greater than two times the maximum frequency content of the signal, then

$$\boxed{F_f(s) = \sum_{n=-\infty}^{+\infty} f(nR_{time})\, H_r(s)\, e^{-j(2\pi s\, nR_{time})}}$$ where $H_r(s) = \begin{cases} R_{time}; & 2\pi|s| \leq \frac{\Omega_s}{2} \\ 0; & 2\pi|s| > \frac{\Omega_s}{2} \end{cases}$

and $\Omega_s = \frac{2\pi}{T_s} = \frac{2\pi}{R_{time}}$

$$\therefore H_r(s) = \begin{cases} R_{time}; & |s| \leq \frac{1}{2R_{time}} \\ 0; & |s| > \frac{1}{2R_{time}} \end{cases}$$

But for now, we can consider the approximate integral from (5) using the samples $\{f(nR_{time})\}$ to be equal to

$$\boxed{F_{fp}(s) = \sum_{n=-\infty}^{+\infty} f(nR_{time}) \cdot e^{-j(2\pi s) nR_{time}}}$$ where $f(nR_{time}) = e^{-a^2\pi (nR_{time})^2}$

(c) The approximation to (5) that we constructed in (b) above can be modified if we want the sum to be finite, ie we consider the samples $f(nR_{time})$ to be non-zero only when $-L/2 \leq nR_{time} < L/2 \Rightarrow \frac{-L}{2R_{time}} \leq n < \frac{L}{2R_{time}}$

$$\text{ie } \boxed{\left(F_{fp}(s)\right)_{app} = \sum_{\left[\frac{-L}{2R_{time}}\right]+1}^{\left[\frac{L}{2R_{time}}\right]} f(nR_{time}) \cdot \omega_1(nR_{time})\, e^{-j(2\pi s) nR_{time}}}$$

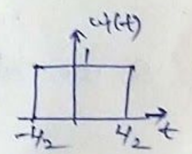
Here ① $\left[\frac{L}{2R_{time}}\right]$ = Greatest integer function value of $L/2n$

$\left[\frac{L}{2R_{time}}\right] \leq \frac{L}{2R_{time}}$ but of integer value

② $\left[\frac{-L}{2R_{time}}\right]$ is the GIF value for $\left(\frac{-L}{2R_{time}}\right)$

So finally, Modifying approximation to finite sum $\Rightarrow$

$$\boxed{\left(F_{fp}(s)\right)_{app} = \sum_{\left[\frac{-L}{2R_{time}}\right]+1}^{\left[\frac{L}{2R_{time}}\right]} f(nR_{time})\, \omega_1(nR_{time})\, e^{-j(2\pi s)(nR_{time})}}$$ where $\omega_1(t) =$



ⓓ The expression which we get in ⓑ gives a continuous function of the frequency s. We evaluate it at $s = mR_{freq}$ for m such that

$$-\frac{B}{2} \leq mR_{freq} < \frac{B}{2}$$

We have to write an approximate expression for $Ff(mR_{freq})$.

$$Ff_p(mR_{freq}) = \left(Ff(mR_{freq})\right)_{approx} = \sum_{n=-\infty}^{+\infty} f(nR_{time})\, e^{-j2\pi(mR_{freq})(nR_{time})}$$

$$\therefore \boxed{\left(Ff(mR_{freq})\right)_{approx} = \sum_{n=-\infty}^{+\infty} f(nR_{time})\, e^{-j2\pi(mR_{freq})(nR_{time})}} \; ; \; -\frac{B}{2} \leq mR_{freq} < \frac{B}{2}$$

where $f(nR_{time}) = e^{-a^2\pi(nR_{time})^2}$

If we consider $f[n]$ to be of finite length $\Rightarrow N \geq L$

Ans $$\boxed{\left(Ff(mR_{freq})\right)_{app\ Limited} = \sum_{n=\left[\frac{-L}{2R_{time}}\right]+1}^{\left[\frac{L}{2R_{time}}\right]} f(nR_{time})\, e^{-j2\pi(mR_{freq})(nR_{time})} \quad -\frac{B}{2} \leq mR_{freq} \leq B/2}$$

If we observe carefully, we observe that ⓑ is actually Discrete time Fourier transform (DTFT) and, In ⓓ, it is basically Discrete Fourier transform (DFT).

So In general, if we have $x(t)$ signal, then its DTFT $X(j\omega)$ is given by –

$$X(j\omega) = \sum_{n} x[n]\, e^{-j\omega n} \qquad x[n] = x(nT_s) \text{ where } T_s = \text{Sampling period}$$

and if $x[n]$ is of finite length and if we are calculating DFT wrt N values s.t $N \geq L$ (Length of signal), then

DFT equations are –

$$\boxed{\begin{aligned} x[n] &= \frac{1}{N}\sum_{K=0}^{N-1} X[K]\, e^{j\frac{2\pi K n}{N}} \; ; \quad 0 \leq n < N \\ X[K] &= \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi K n}{N}} \; ; \quad 0 \leq K < N \end{aligned}}$$

(e) We know that $X(j\omega) = \sum_n x[n] \cdot e^{-j\omega n}$

where we got this expression from $X_p(j\Omega) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-j\Omega t} dt$

Here $\omega = \Omega T_s$, Here $T_s = R_{time}$

In our case, when we started the part (b), $\Omega = 2\pi s$

$\therefore \boxed{\omega = (2\pi s) R_{time}}$

$\therefore$ The expression which we got in (b) is

$$F_{fp}(s) = \sum_{n=-\infty}^{+\infty} f(nR_{time}) \cdot e^{-j(2\pi s) n R_{time}}$$

Here $f(nR_{time}) = f[n]$

and $\omega = (2\pi s) R_{time}$

$$\therefore F_{fp}(s) = \sum_n f[n] \cdot e^{-j\omega n} \quad \text{where } \omega = (2\pi R_{time}) s$$

$$\| $$

$$F(j\,2\pi s R_{time}) = F(j\omega) = \sum_n f[n] \cdot e^{-j\omega n}$$

Now In ~~Synthesis~~ Analysis equation of DFT i-e

$$X[K] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi K n}{N}} \quad 0 \le K < N$$

Here $X[m] = X\left(\frac{j 2\pi m}{N}\right)$

Hence for Calculating $F[m]$ i-e $F\left(j \frac{2\pi m}{N}\right)$, $\omega = \frac{2\pi m}{N}$

$$\therefore (2\pi) R_{time} s = \frac{2\pi m}{N}$$

Hence $\boxed{s = \frac{m}{N R_{time}}}$

Hence $F\left(j \frac{2\pi m}{N}\right) = \sum_n f[n] \cdot e^{-j(2\pi R_{time}) \frac{m n}{N R_{time}}} = F_f(mR_{freq})$ which we got in (d) part

In (d) part, we got

$$\left(F_f(mR_{freq})\right)_{approx} = \sum_n f[n] \cdot e^{-j2\pi (R_{freq} m)(n R_{time})}$$

Hence by Comparison, we say that

$$R_{freq} = \frac{1}{N R_{time}}$$

Hence $\boxed{N = \frac{1}{R_{time} R_{freq}}}$ where N is actually the number of samples to be stored in time domain as well as in freq. domain.

∴ The number of samples need to be stored in time domain is equal to

$$\boxed{N = \frac{1}{R_{time} R_{freq}} = LB}$$ because $LR_{freq} = BR_{time} = 1$

$R_{freq} = \frac{1}{L}$ and $R_{time} = \frac{1}{B}$.

Also, the number of values computed in the frequency domain is also equal to $\boxed{N = LB = \frac{1}{R_{time} \cdot R_{freq}}}$.

Now the number of samples considered for getting $x[n]$ back from $x_p[n]$ where $x_p[n] = x[n] * p[n]$ ie We are dividing the freq. domain analysis $X_p(j\omega) = X(j\omega) \cdot P(j\omega)$ as $\boxed{X_p(j\omega) = \sum_{K=0}^{N-1} X\left(j \frac{2\pi K}{N}\right) \cdot \delta\left(\omega - \frac{2\pi K}{N}\right)}$ N is the no. of samples taken in 2π freq. range

is equal to $\left(\frac{L}{2R_{time}} + \frac{L}{2R_{time}}\right) = \frac{L}{R_{time}} = n_1$

and the number of values which we get in freq. domain is equal to $\frac{B}{2R_{freq}} + \frac{B}{2R_{freq}} = \frac{B}{R_{freq}} = n_2$

∴ For DFT, $n_1 = n_2$ (must condition)

$$\therefore \frac{L}{R_{time}} = \frac{B}{R_{freq}} \Rightarrow \boxed{LR_{freq} = BR_{time} = C}$$

∴ We would like to pick $LR_{freq} = BR_{time} = 1$. $(C=1)$

Q.3_(f).py (Part (f) in Question.3) :

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

def Original_function(a,Rtime):
        t = np.arange(0,2,Rtime) # time vector
        y = np.exp((-a**2)*np.pi*((t)**2))
        return t,y

def DFT_cal_for_givenFunction(a,Rtime):
        Fs = 1.0/Rtime; # sampling rate
        t,y = Original_function(a,Rtime)
        N = len(y) # length of the signal
        Rfreq = 1/(N*Rtime)
        k = np.arange(N)
        frq = k*Rfreq # two sides frequency range
        frq = frq[range(int(N/2))] # one side frequency range
        Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
        Y = Y[range(int(N/2))]
        return t,y,N,frq,Y

t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction(1,1/100)
t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction(2,1/500)
subplot(2, 2,1)
plt.plot(t1,y1,'b',label='Original signal when a = 1')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()

subplot(2,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff(s)) when a = 1 and Rtime = 1/100
s,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,20)
plt.grid()
plt.legend()
plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s ')
```

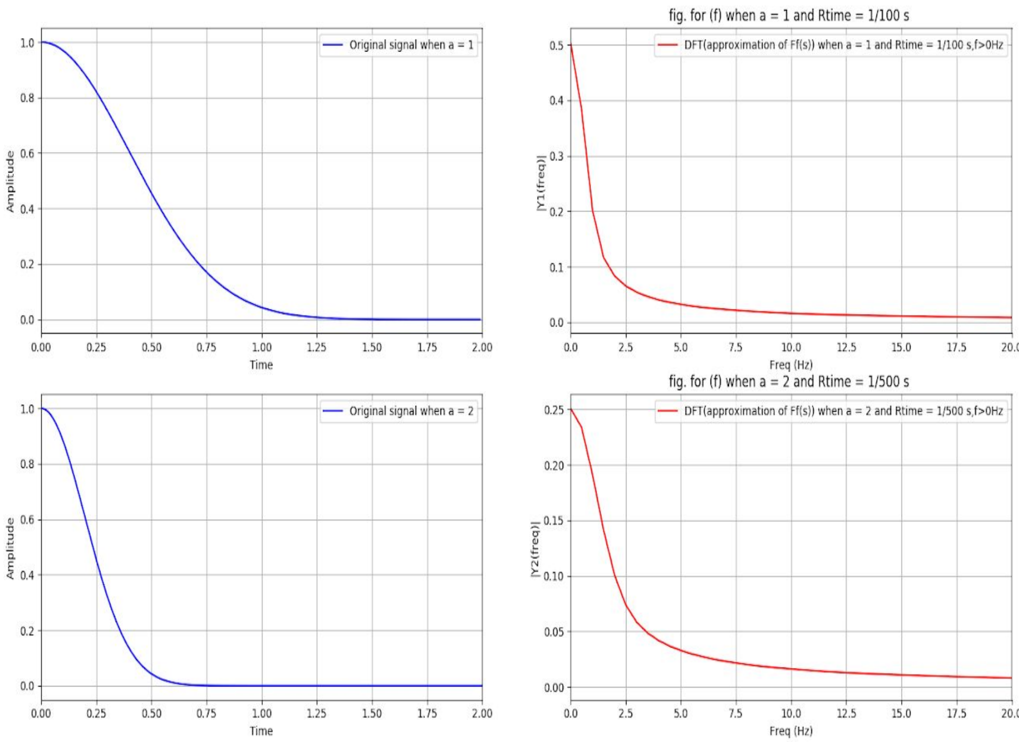
```
print('In first two plots for signal and its Ff(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,hence Rfreq = 1/2 Hz,N = 200')

print('****************************************************')
subplot(2,2,3)
plt.plot(t2,y2,'b',label='Original signal when a = 2')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()

subplot(2,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff(s)) when a = 2 and Rtime = 1/500 s,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,20)
plt.grid()
plt.legend()
plt.title('fig. for (f) when a = 2 and Rtime = 1/500 s ')
print('In last two plots for signal and its Ff(s) approx,we observe that we have taken a = 2 and Rtime = 1/500 s ,hence Rfreq = 1/2 Hz,N = 1000')
print('---------------------------------------------')
print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
plt.show()
```

Output in terminal:

In first two plots for signal and its Ff(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,hence Rfreq = 1/2 Hz,N = 200
****
In last two plots for signal and its Ff(s) approx,we observe that we have taken a = 2 and Rtime = 1/500 s ,hence Rfreq = 1/2 Hz,N = 1000

Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

Figure Obtained after running the code:



Q.3_(g).py (Part (g) in Question.3) :

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

def Original_function(a,Rtime,Rfreq):
        t = np.arange(0,1/Rfreq,Rtime) # time vector
        y = np.exp((-a**2)*np.pi*((t)**2))
        return t,y

def DFT_cal_for_givenFunction(a,Rtime,Rfreq):
        Fs = 1.0/Rtime; # sampling rate
        t,y = Original_function(a,Rtime,Rfreq)
        N = len(y)# length of the signal
        Rfreq = 1/(N*Rtime)
        k = np.arange(N)
```

```
        frq = k*Rfreq # two sides frequency range
        frq = frq[range(int(N/2))] # one side frequency range
        Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
        Y = Y[range(int(N/2))]
        return t,y,N,frq,Y

t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction(1,1/100,1/3)

t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction(2,1/500,1/4)

t3,y3,N3,frq3,Y3 = DFT_cal_for_givenFunction(3,1/1000,1/6.5)

t4,y4,N4,frq4,Y4 = DFT_cal_for_givenFunction(3.5,1/1200,4)

subplot(4, 2,1)
plt.plot(t1,y1,'b',label='Original signal when a = 1')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()

subplot(4,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff(s)) when a = 1 and Rtime = 1/100
s,Rfreq = 1/3Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,40)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')



subplot(4,2,3)
plt.plot(t2,y2,'b',label='Original signal when a = 2')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
```

```
plt.legend()

subplot(4,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff(s)) when a = 2 and Rtime = 1/500 s,
Rfreq = 1/4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,40)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 2 and Rtime = 1/500 s,Rfreq = 1/4 Hz ')
print('In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 2 and Rtime = 1/500 s ,Rfreq = 1/4 Hz,N = 2000')
print('---------------------------------------------')



subplot(4,2,5)
plt.plot(t3,y3,'b',label='Original signal when a = 3')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()

subplot(4,2,6)
plt.plot(frq3,abs(Y3),'r',label = 'DFT(approximation of Ff(s)) when a = 3 and Rtime = 1/1000 s,
Rfreq = 1/6.5 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y3(freq)|')
plt.xlim(0,40)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3 and Rtime = 1/1000 s,Rfreq = 8 Hz ')
print('In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 3 and Rtime = 1/1000 s ,Rfreq = 1/6.5 Hz,N = 650')
print('---------------------------------------------')



subplot(4,2,7)
plt.plot(t4,y4,'b',label='Original signal when a = 3.5')
plt.xlabel('Time')
```

```
plt.ylabel('Amplitude')
plt.xlim(0,0.25)
plt.grid()
plt.legend()

subplot(4,2,8)
plt.plot(frq4,abs(Y4),'r',label = 'DFT(approximation of Ff(s)) when a = 3.5 and Rtime = 1/1200 s,
Rfreq = 4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y4(freq)|')
plt.xlim(0,40)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3.5 and Rtime = 1/1200 s,Rfreq = 4 Hz ')
print('In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 3.5 and Rtime = 1/1200 s ,Rfreq = 4 Hz,N = 300')
print('---------------------------------------------')

print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of
fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
plt.show()
```

Output in terminal:

In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300

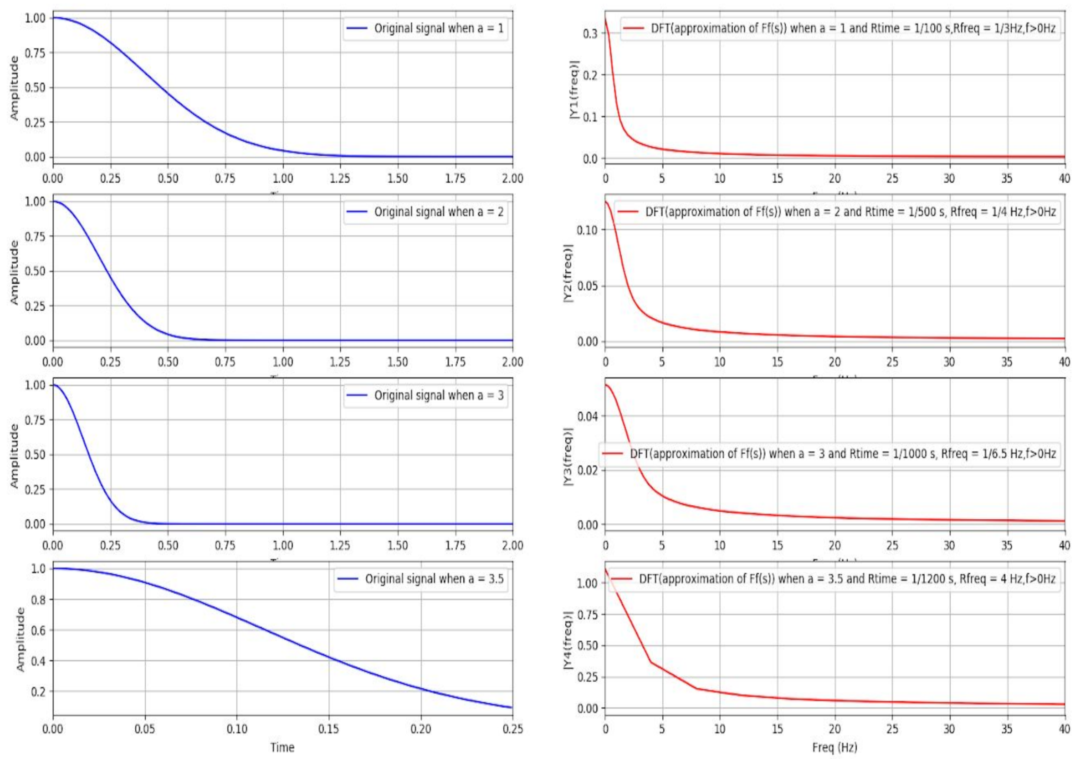
In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 2 and Rtime = 1/500 s ,Rfreq = 1/4 Hz,N = 2000

In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 3 and Rtime = 1/1000 s ,Rfreq = 1/6.5 Hz,N = 650

In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 3.5 and Rtime = 1/1200 s ,Rfreq = 4 Hz,N = 300

Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

Figure Obtained after running the code:



Q.3_(h)(w2).py (Part (h) taking w2(t) as window function in Question.3) :

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

def Original_function(a,Rtime,Rfreq):
        L = 1/Rtime
        t = np.arange(-L/2,L/2,Rtime) # time vector
        y = (np.exp((-a**2)*np.pi*((t)**2)))*(1-((2*np.abs(t))/L))
        return t,y,L

def DFT_cal_for_givenFunction(a,Rtime,Rfreq):
```

```
        Fs = 1.0/Rtime; # sampling rate
        t,y,L = Original_function(a,Rtime,Rfreq)
        N = len(y)# length of the signal
        Rfreq = 1/(N*Rtime)
        k = np.arange(N)
        frq = k*Rfreq # two sides frequency range
        frq = frq[range(int(N/2))] # one side frequency range
        Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
        Y = Y[range(int(N/2))]
        return L,t,y,N,frq,Y

L1,t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction(1,1/100,1/3)

L2,t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction(2,1/500,1/4)

L3,t3,y3,N3,frq3,Y3 = DFT_cal_for_givenFunction(3,1/1000,1/6.5)

L4,t4,y4,N4,frq4,Y4 = DFT_cal_for_givenFunction(3.5,1/1200,4)

subplot(4, 2,1)
plt.plot(t1,y1,'b',label='function1 when a = 1')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(4,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff(s)) when a = 1 and Rtime = 1/100
s,Rfreq = 1/3Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,10)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')



subplot(4,2,3)
```

```
plt.plot(t2,y2,'b',label='function1 when a = 2')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(4,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff(s)) when a = 2 and Rtime = 1/500 s,
Rfreq = 1/4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,10)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 2 and Rtime = 1/500 s,Rfreq = 1/4 Hz ')
print('In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 2 and Rtime = 1/500 s ,Rfreq = 1/4 Hz,N = 2000')
print('--------------------------------------------')



subplot(4,2,5)
plt.plot(t3,y3,'b',label='function1 when a = 3')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(4,2,6)
plt.plot(frq3,abs(Y3),'r',label = 'DFT(approximation of Ff(s)) when a = 3 and Rtime = 1/1000 s,
Rfreq = 1/6.5 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y3(freq)|')
plt.xlim(0,10)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3 and Rtime = 1/1000 s,Rfreq = 8 Hz ')
print('In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 3 and Rtime = 1/1000 s ,Rfreq = 1/6.5 Hz,N = 650')
print('--------------------------------------------')
```

```
subplot(4,2,7)
plt.plot(t4,y4,'b',label='function1 when a = 3.5')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-0.25,0.25)
plt.grid()
plt.legend(loc = 'best')

subplot(4,2,8)
plt.plot(frq4,abs(Y4),'r',label = 'DFT(approximation of Ff(s)) when a = 3.5 and Rtime = 1/1200 s,
Rfreq = 4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y4(freq)|')
plt.xlim(0,10)
plt.grid()
plt.legend(loc = 'best')
#plt.title('fig. for (f) when a = 3.5 and Rtime = 1/1200 s,Rfreq = 4 Hz ')
print('In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 3.5 and Rtime = 1/1200 s ,Rfreq = 4 Hz,N = 300')
print('---------------------------------------------')

print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of
fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
print('---------------------------------------------')
print('Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 ,
and for t>L/2 or t<-L/2,the value of the function f(t)w2(t) = 0')
plt.show()
```

Output in terminal:

In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300

In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 2 and Rtime = 1/500 s ,Rfreq = 1/4 Hz,N = 2000

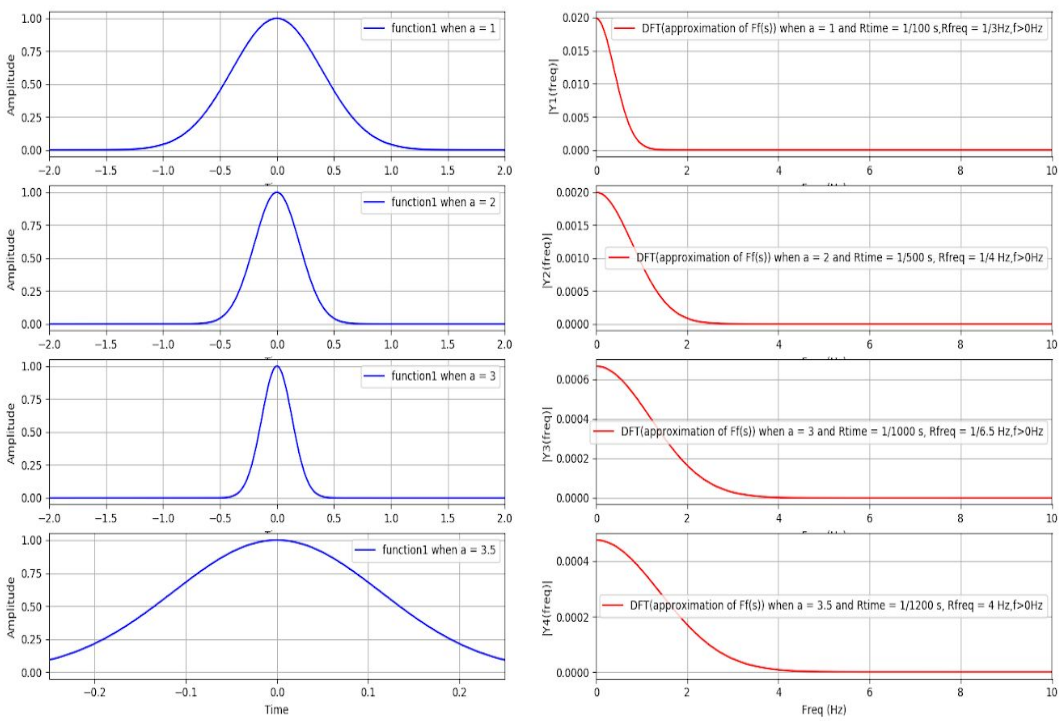
In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 3 and Rtime = 1/1000 s ,Rfreq = 1/6.5 Hz,N = 650

In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 3.5 and Rtime = 1/1200 s ,Rfreq = 4 Hz,N = 300

Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , and for t>L/2 or t<-L/2,the value of the function f(t)w2(t) = 0

Figure Obtained after running the code:



Q.3_(h)(w3).py (Part (h) taking w3(t) as window function in Question.3) :

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

```
def Original_function(a,Rtime,Rfreq):
        L = 1/Rtime
        t = np.arange(-L/2,L/2,Rtime) # time vector
        y = (np.exp((-a**2)*np.pi*((t)**2)))*((np.sin(2*np.pi*t/L))**2)
        return t,y,L

def DFT_cal_for_givenFunction(a,Rtime,Rfreq):
        Fs = 1.0/Rtime; # sampling rate
        t,y,L = Original_function(a,Rtime,Rfreq)
        N = len(y)# length of the signal
        Rfreq = 1/(N*Rtime)
        k = np.arange(N)
        frq = k*Rfreq # two sides frequency range
        frq = frq[range(int(N/2))] # one side frequency range
        Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
        Y = Y[range(int(N/2))]
        return L,t,y,N,frq,Y

L1,t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction(1,1/100,1/3)

L2,t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction(2,1/500,1/4)

L3,t3,y3,N3,frq3,Y3 = DFT_cal_for_givenFunction(3,1/1000,1/6.5)

L4,t4,y4,N4,frq4,Y4 = DFT_cal_for_givenFunction(3.5,1/1200,4)

subplot(4, 2,1)
plt.plot(t1,y1,'b',label='Original signal when a = 1')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(4,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff(s)) when a = 1 and Rtime = 1/100
s,Rfreq = 1/3Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,10)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
```

```
print('In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')



subplot(4,2,3)
plt.plot(t2,y2,'b',label='Original signal when a = 2')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(4,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff(s)) when a = 2 and Rtime = 1/500 s,
Rfreq = 1/4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,10)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 2 and Rtime = 1/500 s,Rfreq = 1/4 Hz ')
print('In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 2 and Rtime = 1/500 s ,Rfreq = 1/4 Hz,N = 2000')
print('---------------------------------------------')



subplot(4,2,5)
plt.plot(t3,y3,'b',label='Original signal when a = 3')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(4,2,6)
plt.plot(frq3,abs(Y3),'r',label = 'DFT(approximation of Ff(s)) when a = 3 and Rtime = 1/1000 s,
Rfreq = 1/6.5 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y3(freq)|')
plt.xlim(0,10)
```

```
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3 and Rtime = 1/1000 s,Rfreq = 8 Hz ')
print('In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 3 and Rtime = 1/1000 s ,Rfreq = 1/6.5 Hz,N = 650')
print('---------------------------------------------')



subplot(4,2,7)
plt.plot(t4,y4,'b',label='Original signal when a = 3.5')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-0.25,0.25)
plt.grid()
plt.legend(loc = 'best')

subplot(4,2,8)
plt.plot(frq4,abs(Y4),'r',label = 'DFT(approximation of Ff(s)) when a = 3.5 and Rtime = 1/1200 s,
Rfreq = 4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y4(freq)|')
plt.xlim(0,10)
plt.grid()
plt.legend(loc = 'best')
#plt.title('fig. for (f) when a = 3.5 and Rtime = 1/1200 s,Rfreq = 4 Hz ')
print('In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken
a = 3.5 and Rtime = 1/1200 s ,Rfreq = 4 Hz,N = 300')
print('---------------------------------------------')

print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of
fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
print('---------------------------------------------')
print('Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 ,
and for t>L/2 or t<-L/2,the value of the function f(t)w3(t) = 0')
plt.show()
```

Output in terminal:

In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300

In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 2 and Rtime = 1/500 s ,Rfreq = 1/4 Hz,N = 2000

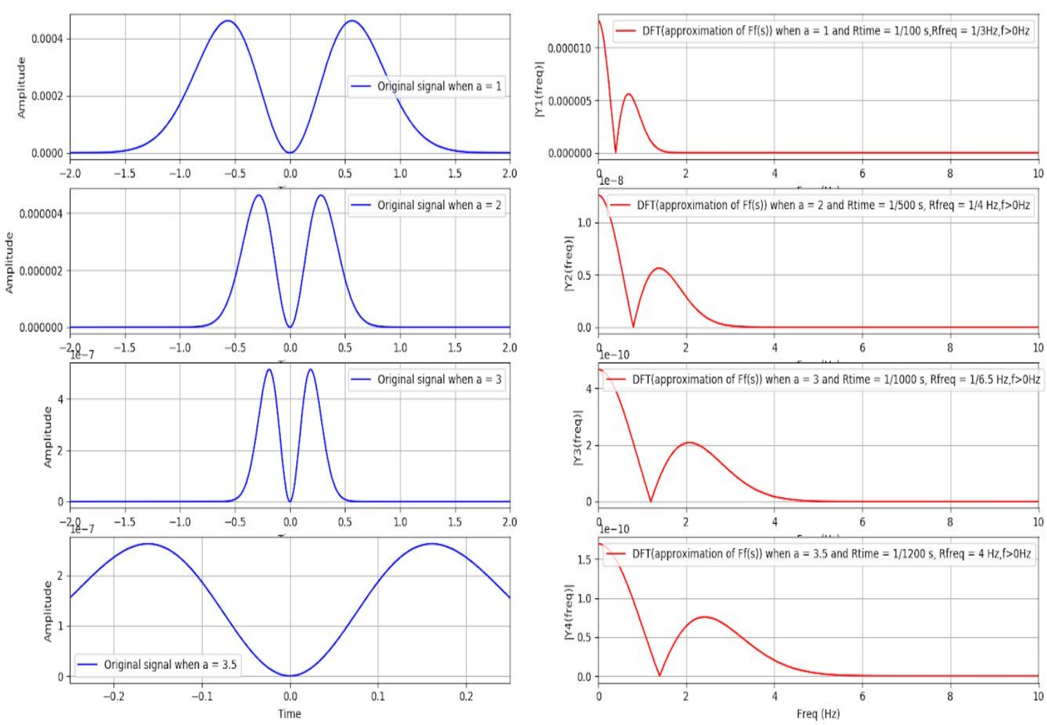
In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 3 and Rtime = 1/1000 s ,Rfreq = 1/6.5 Hz,N = 650

In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken a = 3.5 and Rtime = 1/1200 s ,Rfreq = 4 Hz,N = 300

Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , and for t>L/2 or t<-L/2,the value of the function f(t)w3(t) = 0

Figure Obtained after running the code:



```
#  Q.3_(h)_compare.py (Part (h) comparing wrt window functions in Question.3) :

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

def function1(a,Rtime,Rfreq):
	L = 1/Rtime
	t = np.arange(-L/2,L/2,Rtime) # time vector
	y = (np.exp((-a**2)*np.pi*((t)**2)))*(1-((2*np.abs(t))/L))
	return t,y,L

def function2(a,Rtime,Rfreq):
	L = 1/Rtime
	t = np.arange(-L/2,L/2,Rtime) # time vector
	y = (np.exp((-a**2)*np.pi*((t)**2)))*(np.sin(2*np.pi*t/L))**2
	return t,y,L

def DFT_cal_for_givenFunction1(a,Rtime,Rfreq):
	Fs = 1.0/Rtime; # sampling rate
	t,y,L = function1(a,Rtime,Rfreq)
	N = len(y)# length of the signal
	Rfreq = 1/(N*Rtime)
	k = np.arange(N)
	frq = k*Rfreq # two sides frequency range
	frq = frq[range(int(N/2))] # one side frequency range
	Y = np.abs(np.fft.fft(y)/N) # fft computing and normalization
	Y = Y[range(int(N/2))]
	return L,t,y,N,frq,Y

def DFT_cal_for_givenFunction2(a,Rtime,Rfreq):
	Fs = 1.0/Rtime; # sampling rate
	t,y,L = function2(a,Rtime,Rfreq)
	N = len(y)# length of the signal
	Rfreq = 1/(N*Rtime)
	k = np.arange(N)
	frq = k*Rfreq # two sides frequency range
	frq = frq[range(int(N/2))] # one side frequency range
	Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
	Y = Y[range(int(N/2))]
	return L,t,y,N,frq,Y

L1,t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction1(1,1/100,1/3)
```

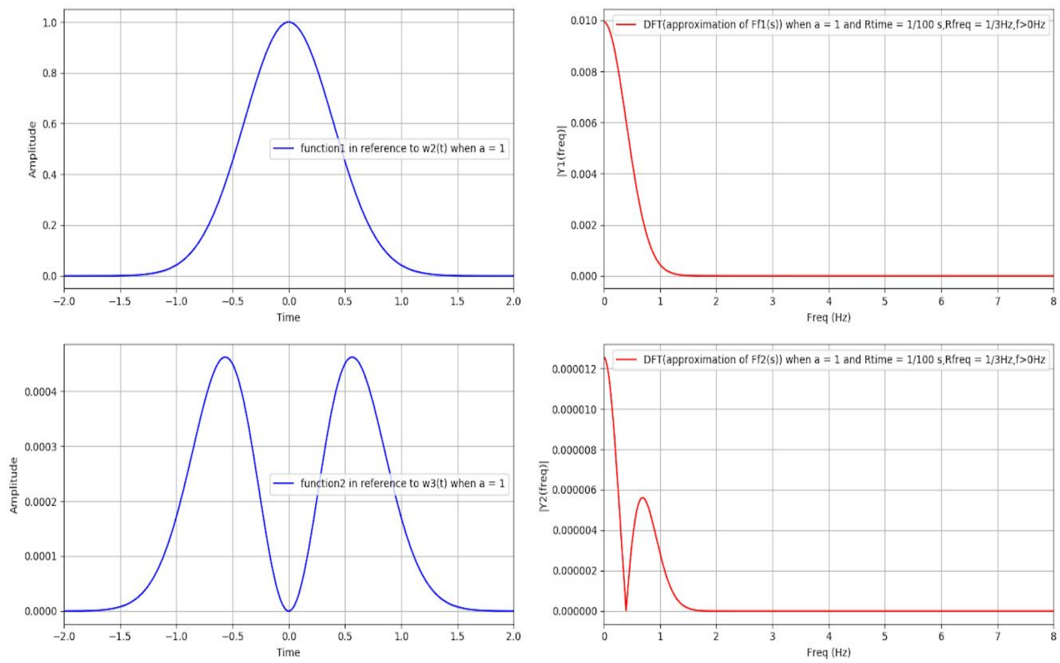
```
L2,t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction2(1,1/100,1/3)

subplot(2, 2,1)
plt.plot(t1,y1,'b',label='function1 in reference to w2(t) when a = 1')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(2,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff1(s)) when a = 1 and Rtime = 1/100
s,Rfreq = 1/3Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,8)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row1,there are two plots for function1 and its Ff1(s) approx,we observe that we have
taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')

subplot(2, 2,3)
plt.plot(t2,y2,'b',label='function2 in reference to w3(t) when a = 1')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(2,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff2(s)) when a = 1 and Rtime = 1/100
s,Rfreq = 1/3Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,8)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row2,there are two plots for function2 and its F2(s) approx,we observe that we have
taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')
```

```
print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
print('--------------------------------------------')
print('Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , and for t>L/2 or t<-L/2,the value of the function f(t)w2(t) = 0')
plt.show()
```

Output in terminal:

In Row1,there are two plots for function1 and its Ff1(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300
--
In Row2,there are two plots for function2 and its F2(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300
--
Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.
--
Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , and for t>L/2 or t<-L/2,the value of the function f(t)w2(t) and f(t)w3(t) = 0

Figure Obtained after running the code:




```
#  Q.3_(i)(f).py (Part (i) solving for (f) part in Question.3) :

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

def Original_function(Rtime):
        t = np.arange(0,2,Rtime) # time vector
        y = np.cos(2*np.pi*t) + 0.5*np.sin(4*np.pi*t)
        return t,y

def DFT_cal_for_givenFunction(Rtime):
        Fs = 1.0/Rtime; # sampling rate
        t,y = Original_function(Rtime)
        N = len(y) # length of the signal
        Rfreq = 1/(N*Rtime)
        k = np.arange(N)
        frq = k*Rfreq # two sides frequency range
        frq = frq[range(int(N/2))] # one side frequency range
```

```
    Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
    Y = Y[range(int(N/2))]
    return t,y,N,frq,Y

t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction(1/10)
t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction(1/500)
subplot(2, 2,1)
plt.plot(t1,y1,'b',label='g(t) when Rtime = 1/10 s')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,1.75)
plt.grid()
plt.legend()

subplot(2,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/50 s,f>0Hz') #
plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,2.6)
plt.grid()
plt.legend()
plt.title('fig. for (f) when Rtime = 1/10 s ')
print('In first two plots for signal and its Ff(s) approx,we observe that we have taken Rtime =
1/10 s ,hence Rfreq = 1/2 Hz,N = 20')

print('****************************************************')
subplot(2,2,3)
plt.plot(t2,y2,'b',label='g(t) when time = 1/500 s')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,1.75)
plt.grid()
plt.legend()

subplot(2,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/500 s,f>0Hz') #
plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,20)
plt.grid()
plt.legend()
```

```
plt.title('fig. for (f) when Rtime = 1/500 s ')
print('In last two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/500 s ,hence Rfreq = 1/2 Hz,N = 1000')
print('---------------------------------------------')
print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
plt.show()
```

Output in terminal:

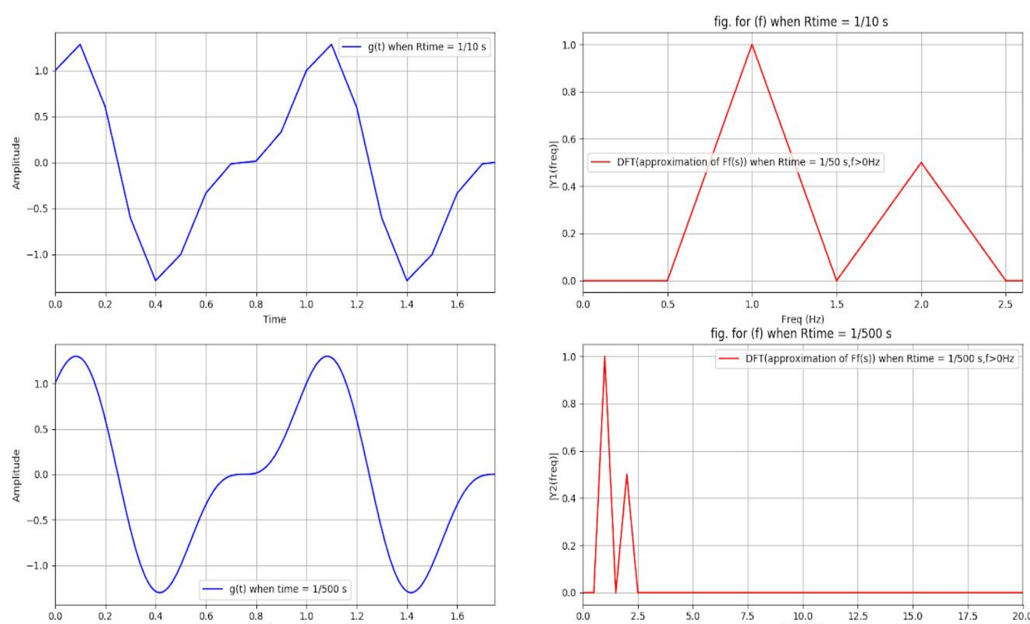
In first two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/10 s ,hence Rfreq = 1/2 Hz,N = 20

****

In last two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/500 s ,hence Rfreq = 1/2 Hz,N = 1000

Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

Figure Obtained after running the code:



Q.3_(i)(g).py (Part (i) solving for (g) part in Question.3) :

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

def Original_function(Rtime,Rfreq):
        t = np.arange(0,1/Rfreq,Rtime) # time vector
        y = np.cos(2*np.pi*t) + 0.5*np.sin(4*np.pi*t)
        return t,y

def DFT_cal_for_givenFunction(Rtime,Rfreq):
        Fs = 1.0/Rtime; # sampling rate
        t,y = Original_function(Rtime,Rfreq)
        N = len(y)# length of the signal
        Rfreq = 1/(N*Rtime)
        k = np.arange(N)
        frq = k*Rfreq # two sides frequency range
        frq = frq[range(int(N/2))] # one side frequency range
        Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
        Y = Y[range(int(N/2))]
        return t,y,N,frq,Y

t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction(1/100,1/3)

t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction(1/20,1/2.4)

t3,y3,N3,frq3,Y3 = DFT_cal_for_givenFunction(1/1000,1/9.989)

t4,y4,N4,frq4,Y4 = DFT_cal_for_givenFunction(1/1200,1.5)

subplot(4, 2,1)
plt.plot(t1,y1,'b',label='Original signal(g(t)) when Rtime = 1/100 s and Rfreq = 1/3 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()

subplot(4,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/100 s,Rfreq =
1/3Hz,f>0Hz') # plotting the spectrum
```

```
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,40)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')



subplot(4,2,3)
plt.plot(t2,y2,'b',label='Original signal(g(t)) when Rtime = 1/20 s and Rfreq = 1/2.4 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()

subplot(4,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/20 s, Rfreq = 1/2.4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,8)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 2 and Rtime = 1/500 s,Rfreq = 1/4 Hz ')
print('In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/20 s ,Rfreq = 1/2.4 Hz,N = 48')
print('---------------------------------------------')



subplot(4,2,5)
plt.plot(t3,y3,'b',label='Original signal(g(t)) when Rtime = 1/1000 s and Rfreq = 1/9.989 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()
```

```
subplot(4,2,6)
plt.plot(frq3,abs(Y3),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/1000 s, Rfreq =
1/9.989 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y3(freq)|')
plt.xlim(0,40)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3 and Rtime = 1/1000 s,Rfreq = 8 Hz ')
print('In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken
Rtime = 1/1000 s ,Rfreq = 1/9.989 Hz,N = 9898')
print('---------------------------------------------')



subplot(4,2,7)
plt.plot(t4,y4,'b',label='Original signal(g(t)) when Rtime = 1/1200 s and Rfreq = 1.5 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,0.6)

plt.grid()
plt.legend()

subplot(4,2,8)
plt.plot(frq4,abs(Y4),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/1200 s, Rfreq = 1.5
Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y4(freq)|')
plt.xlim(0,40)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3.5 and Rtime = 1/1200 s,Rfreq = 4 Hz ')
print('In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken
Rtime = 1/1200 s ,Rfreq = 1.5 Hz,N = 1800')
print('---------------------------------------------')

print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of
fft becomes very large if we don't do these,just for understanding purpose we are taking this.")

plt.show()
```

Output in terminal:

In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300

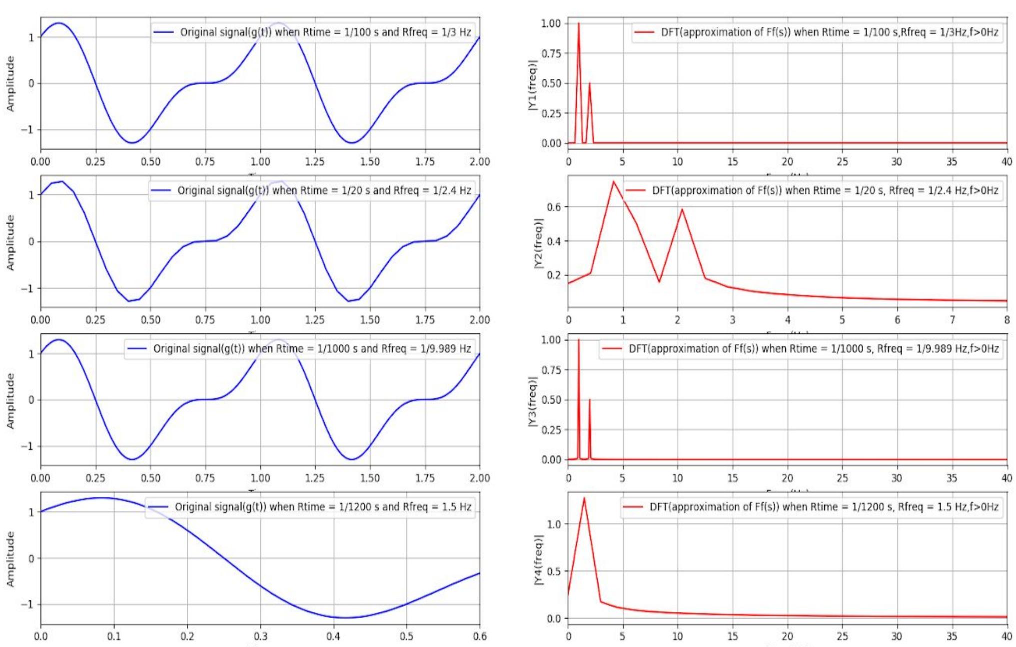
In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/20 s ,Rfreq = 1/2.4 Hz,N = 48

In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/1000 s ,Rfreq = 1/9.989 Hz,N = 9898

In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/1200 s ,Rfreq = 1.5 Hz,N = 1800

Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

Figure Obtained after running the code:



Q.3_(i)(h)(w2).py (Part (i) solving for (h) part taking w2(t) as window function in Question.3) :

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

def Original_function(Rtime,Rfreq):
	L = 1/Rtime
	t = np.arange(-L/2,L/2,Rtime) # time vector
	y = (np.cos(2*np.pi*t) + 0.5*np.sin(4*np.pi*t))*(1-((2*np.abs(t))/L))
	return t,y

def DFT_cal_for_givenFunction(Rtime,Rfreq):
	Fs = 1.0/Rtime; # sampling rate
	t,y = Original_function(Rtime,Rfreq)
	N = len(y)# length of the signal
	Rfreq = 1/(N*Rtime)
	k = np.arange(N)
	frq = k*Rfreq # two sides frequency range
	frq = frq[range(int(N/2))] # one side frequency range
	Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
	Y = Y[range(int(N/2))]
	return t,y,N,frq,Y

t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction(1/100,1/3)

t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction(1/20,1/2.4)

t3,y3,N3,frq3,Y3 = DFT_cal_for_givenFunction(1/1000,1/9.989)

t4,y4,N4,frq4,Y4 = DFT_cal_for_givenFunction(1/1200,1.5)

subplot(4, 2,1)
plt.plot(t1,y1,'b',label='g(t)w2(t) when Rtime = 1/100 s and Rfreq = 1/3 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()

subplot(4,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/100 s,Rfreq = 
1/3Hz,f>0Hz') # plotting the spectrum
```

```
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,2.5)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')



subplot(4,2,3)
plt.plot(t2,y2,'b',label='g(t)w2(t) when Rtime = 1/20 s and Rfreq = 1/2.4 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()

subplot(4,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/20 s, Rfreq = 1/2.4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,2.5)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 2 and Rtime = 1/500 s,Rfreq = 1/4 Hz ')
print('In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/20 s ,Rfreq = 1/2.4 Hz,N = 48')
print('---------------------------------------------')



subplot(4,2,5)
plt.plot(t3,y3,'b',label='g(t)w2(t) when Rtime = 1/1000 s and Rfreq = 1/9.989 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,2)
plt.grid()
plt.legend()
```

```
subplot(4,2,6)
plt.plot(frq3,abs(Y3),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/1000 s, Rfreq =
1/9.989 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y3(freq)|')
plt.xlim(0,2.5)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3 and Rtime = 1/1000 s,Rfreq = 8 Hz ')
print('In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken
Rtime = 1/1000 s ,Rfreq = 1/9.989 Hz,N = 9898')
print('---------------------------------------------')



subplot(4,2,7)
plt.plot(t4,y4,'b',label='g(t)w2(t) when Rtime = 1/1200 s and Rfreq = 1.5 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(0,0.6)

plt.grid()
plt.legend()

subplot(4,2,8)
plt.plot(frq4,abs(Y4),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/1200 s, Rfreq = 1.5
Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y4(freq)|')
plt.xlim(0,2.5)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3.5 and Rtime = 1/1200 s,Rfreq = 4 Hz ')
print('In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken
Rtime = 1/1200 s ,Rfreq = 1.5 Hz,N = 1800')
print('---------------------------------------------')

print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of
fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
print('---------------------------------------------')
print('Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , as
for t>L/2 or t<-L/2,the value of the function g(t)w2(t) = 0 ,g(t)w3(t) = 0')
plt.show()
```

Output in terminal:

In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300

In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/20 s ,Rfreq = 1/2.4 Hz,N = 48

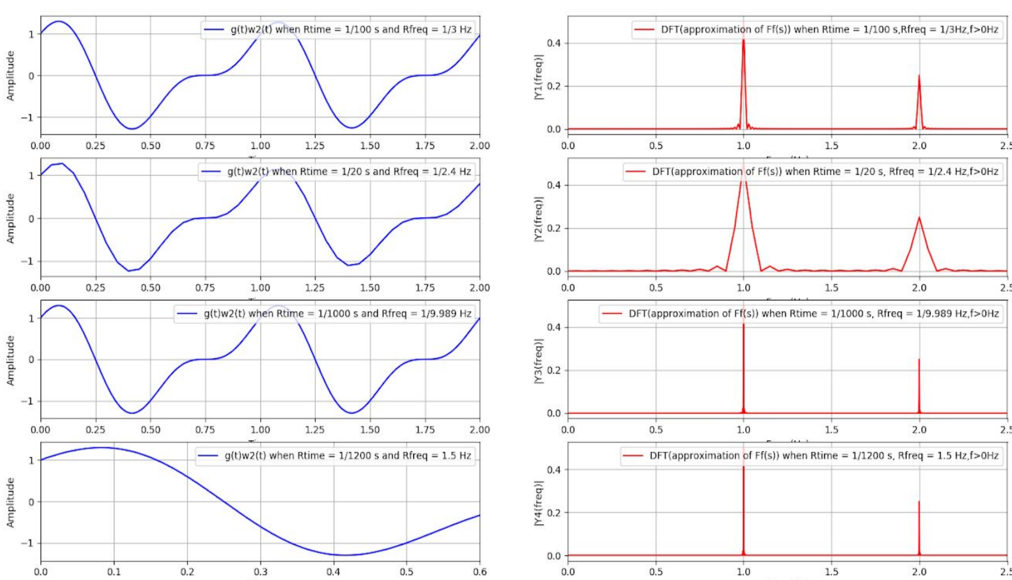
In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/1000 s ,Rfreq = 1/9.989 Hz,N = 9898

In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/1200 s ,Rfreq = 1.5 Hz,N = 1800

Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , as for t>L/2 or t<-L/2,the value of the function g(t)w2(t) = 0 ,g(t)w3(t) = 0

Figure Obtained after running the code:



Q.3_(i)(h)(w3).py (Part (i) solving for (h) part taking w3(t) as window function in Question.3) :

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

def Original_function(Rtime,Rfreq):
	L = 1/Rtime
	t = np.arange(-L/2,L/2,Rtime) # time vector
	y = (np.cos(2*np.pi*t) + 0.5*np.sin(4*np.pi*t))*((np.sin(2*np.pi*t/L))**2)
	return t,y

def DFT_cal_for_givenFunction(Rtime,Rfreq):
	Fs = 1.0/Rtime; # sampling rate
	t,y = Original_function(Rtime,Rfreq)
	N = len(y)# length of the signal
	Rfreq = 1/(N*Rtime)
	k = np.arange(N)
	frq = k*Rfreq # two sides frequency range
	frq = frq[range(int(N/2))] # one side frequency range
	Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
	Y = Y[range(int(N/2))]
	return t,y,N,frq,Y

t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction(1/100,1/3)

t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction(1/20,1/2.4)

t3,y3,N3,frq3,Y3 = DFT_cal_for_givenFunction(1/1000,1/9.989)

t4,y4,N4,frq4,Y4 = DFT_cal_for_givenFunction(1/1200,1.5)

subplot(4, 2,1)
plt.plot(t1,y1,'b',label='g(t)w3(t) when Rtime = 1/100 s and Rfreq = 1/3 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.grid()
plt.legend()

subplot(4,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/100 s,Rfreq =
1/3Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
```

```
plt.ylabel('|Y1(freq)|')
plt.xlim(0,2.5)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')



subplot(4,2,3)
plt.plot(t2,y2,'b',label='g(t)w3(t) when Rtime = 1/20 s and Rfreq = 1/2.4 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.grid()
plt.legend()

subplot(4,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/20 s, Rfreq = 1/2.4 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,2.5)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 2 and Rtime = 1/500 s,Rfreq = 1/4 Hz ')
print('In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/20 s ,Rfreq = 1/2.4 Hz,N = 48')
print('---------------------------------------------')



subplot(4,2,5)
plt.plot(t3,y3,'b',label='g(t)w3(t) when Rtime = 1/1000 s and Rfreq = 1/9.989 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.grid()
plt.legend()

subplot(4,2,6)
plt.plot(frq3,abs(Y3),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/1000 s, Rfreq = 1/9.989 Hz,f>0Hz') # plotting the spectrum
```

```
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y3(freq)|')
plt.xlim(0,2.5)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3 and Rtime = 1/1000 s,Rfreq = 8 Hz ')
print('In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/1000 s ,Rfreq = 1/9.989 Hz,N = 9898')
print('---------------------------------------------')



subplot(4,2,7)
plt.plot(t4,y4,'b',label='g(t)w3(t) when Rtime = 1/1200 s and Rfreq = 1.5 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.grid()
plt.legend()

subplot(4,2,8)
plt.plot(frq4,abs(Y4),'r',label = 'DFT(approximation of Ff(s)) when Rtime = 1/1200 s, Rfreq = 1.5 Hz,f>0Hz') # plotting the spectrum
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y4(freq)|')
plt.xlim(0,2.5)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 3.5 and Rtime = 1/1200 s,Rfreq = 4 Hz ')
print('In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/1200 s ,Rfreq = 1.5 Hz,N = 1800')
print('---------------------------------------------')

print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
print('---------------------------------------------')
print('Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , as for t>L/2 or t<-L/2,the value of the function g(t)w2(t) = 0 ,g(t)w3(t) = 0')
plt.show()
```

Output in terminal:

In Row1,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300

In Row2,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/20 s ,Rfreq = 1/2.4 Hz,N = 48

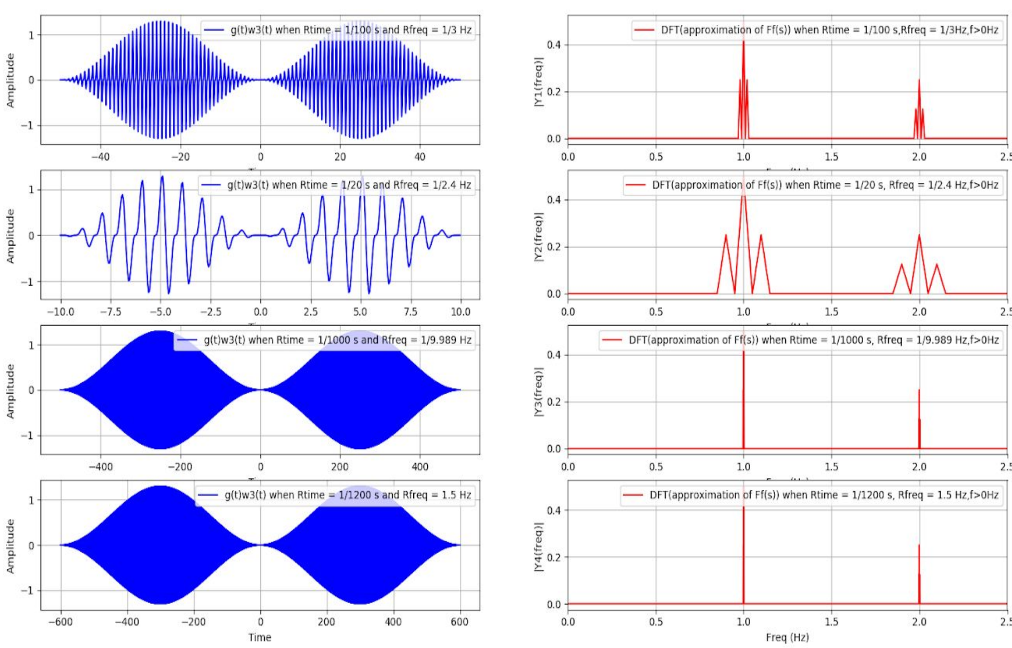
In Row3,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/1000 s ,Rfreq = 1/9.989 Hz,N = 9898

In Row4,there are two plots for signal and its Ff(s) approx,we observe that we have taken Rtime = 1/1200 s ,Rfreq = 1.5 Hz,N = 1800

Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , as for t>L/2 or t<-L/2,the value of the function g(t)w2(t) = 0 ,g(t)w3(t) = 0

Figure Obtained after running the code:



Q.3_(i)(h)_compare.py (Part (i) solving for (h) comparing wrt window functions in Question.3):

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

def function1(Rtime,Rfreq):
        L = 1/Rtime
        t = np.arange(-L/2,L/2,Rtime) # time vector
        y = (np.cos(2*np.pi*t) + 0.5*np.sin(4*np.pi*t))*(1-((2*np.abs(t))/L))
        return t,y,L

def function2(Rtime,Rfreq):
        L = 1/Rtime
        t = np.arange(-L/2,L/2,Rtime) # time vector
        y = (np.cos(2*np.pi*t) + 0.5*np.sin(4*np.pi*t))*((np.sin(2*np.pi*t/L))**2)
        return t,y,L

def DFT_cal_for_givenFunction1(Rtime,Rfreq):
        Fs = 1.0/Rtime; # sampling rate
        t,y,L = function1(Rtime,Rfreq)
        N = len(y)# length of the signal
        Rfreq = 1/(N*Rtime)
        k = np.arange(N)
        frq = k*Rfreq # two sides frequency range
        frq = frq[range(int(N/2))] # one side frequency range
        Y = np.abs(np.fft.fft(y)/N) # fft computing and normalization
        Y = Y[range(int(N/2))]
        return L,t,y,N,frq,Y

def DFT_cal_for_givenFunction2(Rtime,Rfreq):
        Fs = 1.0/Rtime; # sampling rate
        t,y,L = function2(Rtime,Rfreq)
        N = len(y)# length of the signal
        Rfreq = 1/(N*Rtime)
        k = np.arange(N)
        frq = k*Rfreq # two sides frequency range
        frq = frq[range(int(N/2))] # one side frequency range
        Y = 2*np.abs(np.fft.fft(y)/N) # fft computing and normalization
        Y = Y[range(int(N/2))]
        return L,t,y,N,frq,Y
```

```
L1,t1,y1,N1,frq1,Y1 = DFT_cal_for_givenFunction1(1/100,1/3)
L2,t2,y2,N2,frq2,Y2 = DFT_cal_for_givenFunction2(1/100,1/3)

subplot(2, 2,1)
plt.plot(t1,y1,'b',label='g(t)w2(t)')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.xlim(-2,2)
plt.grid()
plt.legend(loc = 'best')

subplot(2,2,2)
plt.plot(frq1,abs(Y1),'r',label = 'DFT(approximation of Ff1(s)) when Rtime = 1/100 s,Rfreq =
1/3Hz,f>0Hz') # plotting the spectrum
#(here f1(t) = g(t)w2(t))
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y1(freq)|')
plt.xlim(0,3)
plt.grid()
plt.legend()
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row1,there are two plots for g(t)w2(t) and its Ff1(s) approx,we observe that we have
taken Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')

subplot(2, 2,3)
plt.plot(t2,y2,'b',label='g(t)w3(t)')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.grid()
plt.legend(loc = 'best')

subplot(2,2,4)
plt.plot(frq2,abs(Y2),'r',label = 'DFT(approximation of Ff2(s)) when a = 1 and Rtime = 1/100
s,Rfreq = 1/3Hz,f>0Hz ') # plotting the spectrum
#(here f2(t) = g(t)w3(t))
plt.xlabel('Freq (Hz)')
plt.ylabel('|Y2(freq)|')
plt.xlim(0,3)
plt.grid()
plt.legend()
```

```
#plt.title('fig. for (f) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz ')
print('In Row2,there are two plots for g(t)w3(t) and its F2(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300')
print('---------------------------------------------')

print("Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.")
print('---------------------------------------------')
print('Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , as for t>L/2 or t<-L/2,the value of the function g(t)w2(t) = 0 ,g(t)w3(t) = 0')
plt.show()
```

Output in terminal:

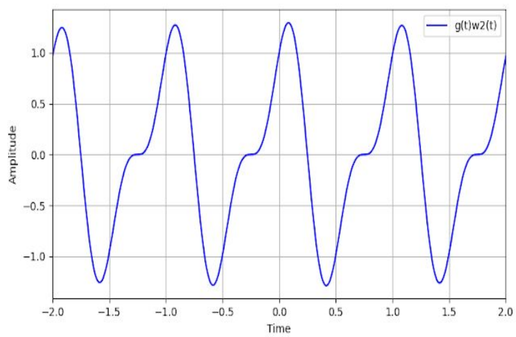
In Row1,there are two plots for g(t)w2(t) and its Ff1(s) approx,we observe that we have taken Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300

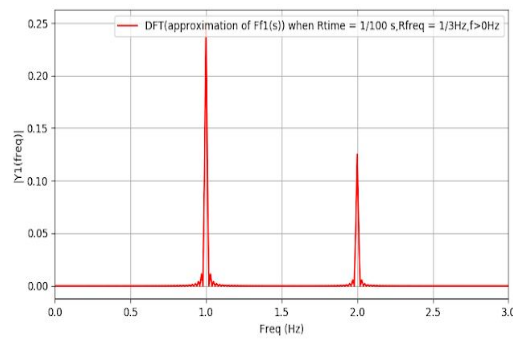
In Row2,there are two plots for g(t)w3(t) and its F2(s) approx,we observe that we have taken a = 1 and Rtime = 1/100 s ,Rfreq = 1/3 Hz,N = 300

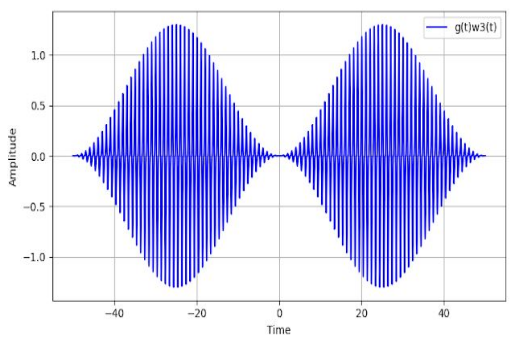
Here we are normalzing fft calculation to 2*(1/N) times the fft value,because the value of fft becomes very large if we don't do these,just for understanding purpose we are taking this.

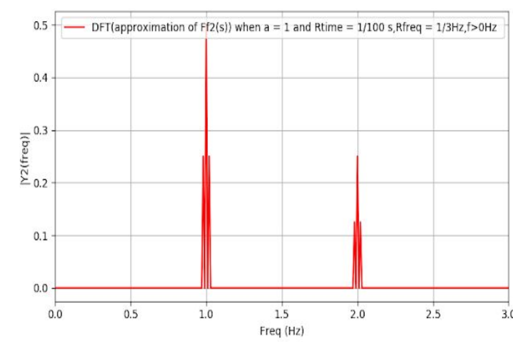
Also we can say that we are considering our calculations for t ranging from -L/2 to L/2 , as for t>L/2 or t<-L/2,the value of the function g(t)w2(t) = 0 ,g(t)w3(t) = 0

Figure Obtained after running the code:


g(t)w2(t)
Amplitude
Time


DFT(approximation of Ff1(s)) when Rtime = 1/100 s,Rfreq = 1/3Hz,f>0Hz
|Y1(freq)|
Freq (Hz)


g(t)w3(t)
Amplitude
Time


DFT(approximation of Ff2(s)) when a = 1 and Rtime = 1/100 s,Rfreq = 1/3Hz,f>0Hz
|Y2(freq)|
Freq (Hz)