In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad
from scipy.integrate import odeint
import scipy.constants as cste
from matplotlib.animation import FuncAnimation
```

TD 2: Waves

A.1 Monochromatic waves

1. Parabolic motion video

In [ ]:
```
import pylab
import os

g = 10
v0 = 10
alpha = np.pi/4

Nframes = 20 # no. of images
tini = 0 # initial time
tfini = 1.4 # final time
tstep = (tfini-tini)/Nframes
Xmin, Xmax, Ymin, Ymax = 0, 12, 0, 3

for n in range(Nframes):
    t = tini + n*tstep
    y = -1/2*g*t**2 + v0*np.sin(alpha)*t
    x = v0*np.cos(alpha)*t
    pylab.plot(x, y, 'o', color='b')
    if n == (Nframes-1):
        pylab.text(6, 2, "Boum !", fontsize=20)
    pylab.axis([Xmin, Xmax, Ymin, Ymax])
    filename = 'figs/fichierTemp'+str('%02d' %n)+'.pdf' # creating file f
    pylab.savefig(filename)
    print(f"Nplot = {n}")
    pylab.clf()

# assemble images into an animation
cmd = "convert -delay 50 -loop 0 figs/fichierTemp*.pdf figs/TrajectoireBo
os.system(cmd)
os.system(f"rm figs/fichierTemp*.pdf")
print("Its done!")
```

```
Nplot = 0
Nplot = 1
Nplot = 2
Nplot = 3
Nplot = 4
Nplot = 5
Nplot = 6
Nplot = 7
Nplot = 8
Nplot = 9
Nplot = 10
Nplot = 11
Nplot = 12
Nplot = 13
Nplot = 14
Nplot = 15
Nplot = 16
Nplot = 17
Nplot = 18
Nplot = 19
Its done!
<Figure size 640x480 with 0 Axes>
```

Ex. 12 Progressive and stationary waves

1. $\psi(x,t) = A\cos(\omega t - kx) = A\cos(\frac{2\pi}{T}t - \frac{2\pi}{\lambda}x)$

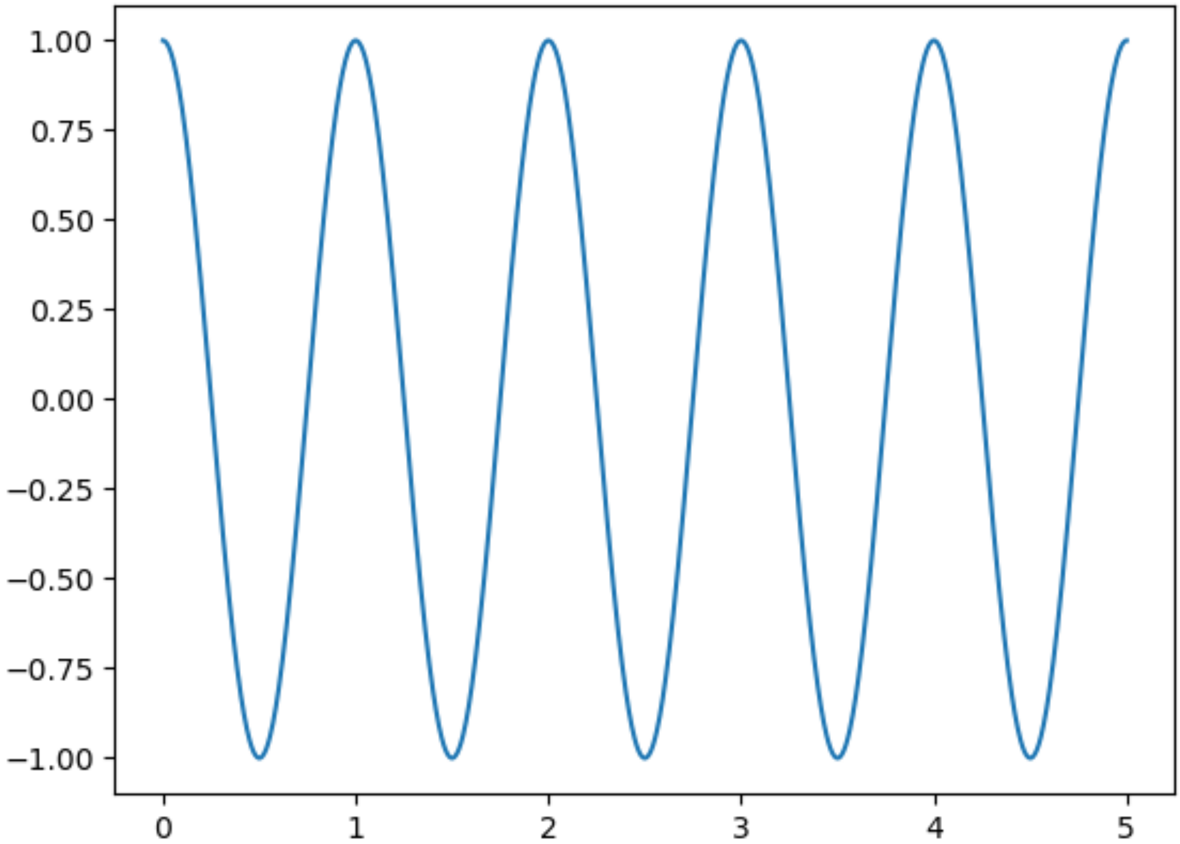
$\text{phase speed} = \frac{\omega}{k} = \frac{\lambda}{T}$

In [ ]:

```
T = 1
lamda = 1
A = 1

omega = 2*np.pi/T
k = 2*np.pi/lamda

x = np.linspace(0, 5, 1000)
psi_t0 = A*np.cos(-k*x)

pylab.plot(x, psi_t0)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x13a41e340>]
```



Progressive

In [ ]:
```
t_range = np.linspace(0, 1, 15)
Xmin, Xmax, Ymin, Ymax = -6, 6, 1.2, 1.2

for i in range(len(t_range)):
    t = t_range[i]
    psi = A*np.cos(omega*t - k*x)
    pylab.plot(x, psi)
    #pylab.axis([Xmin, Xmax, Ymin, Ymax])
    filename = 'figs/fichierTemp'+str('%02d' %i)+'.pdf' # creating file f
    pylab.savefig(filename)
    print(f"Nplot = {i}")
    pylab.clf()
# assemble images into an animation
cmd = "convert -delay 50 -loop 0 figs/fichierTemp*.pdf figs/wave_pro.gif"
os.system(cmd)
os.system(f"rm figs/fichierTemp*.pdf")
print("Its done!")
```

```
Nplot = 0
Nplot = 1
Nplot = 2
Nplot = 3
Nplot = 4
Nplot = 5
Nplot = 6
Nplot = 7
Nplot = 8
Nplot = 9
Nplot = 10
Nplot = 11
Nplot = 12
Nplot = 13
Nplot = 14
Its done!
<Figure size 640x480 with 0 Axes>
```

Regressive

In [ ]:
```
for i in range(len(t_range)):
    t = t_range[i]
    psi = A*np.cos(omega*t + k*x)
    pylab.plot(x, psi)
    #pylab.axis([Xmin, Xmax, Ymin, Ymax])
    filename = 'figs/fichierTemp'+str('%02d' %i)+'.pdf' # creating file f
    pylab.savefig(filename)
    print(f"Nplot = {i}")
    pylab.clf()
# assemble images into an animation
cmd = "convert -delay 50 -loop 0 figs/fichierTemp*.pdf figs/wave_reg.gif"
os.system(cmd)
os.system(f"rm figs/fichierTemp*.pdf")
print("Its done!")
```

```
Nplot = 0
Nplot = 1
Nplot = 2
Nplot = 3
Nplot = 4
Nplot = 5
Nplot = 6
Nplot = 7
Nplot = 8
Nplot = 9
Nplot = 10
Nplot = 11
Nplot = 12
Nplot = 13
Nplot = 14
Its done!
<Figure size 640x480 with 0 Axes>
```

In [ ]:
```
phase = 0
t_range = np.linspace(0, 0.5, 15)
Xmin, Xmax, Ymin, Ymax = 0, 6, -2, 2
for i in range(len(t_range)):
    t = t_range[i]
    psi_1 = A*np.cos(omega*t - k*x)
    psi_2 = A*np.cos(omega*t + k*x + phase)
    psi = psi_1 + psi_2
    pylab.plot(x, psi)
    pylab.axis([Xmin, Xmax, Ymin, Ymax])
    filename = 'figs/fichierTemp'+str('%02d' %i)+'.pdf' # creating file f
    pylab.savefig(filename)
    print(f"Nplot = {i}")
    pylab.clf()
# assemble images into an animation
cmd = "convert -delay 50 -loop 0 figs/fichierTemp*.pdf figs/wave_statio.g
os.system(cmd)
os.system(f"rm figs/fichierTemp*.pdf")
print("Its done!")
```

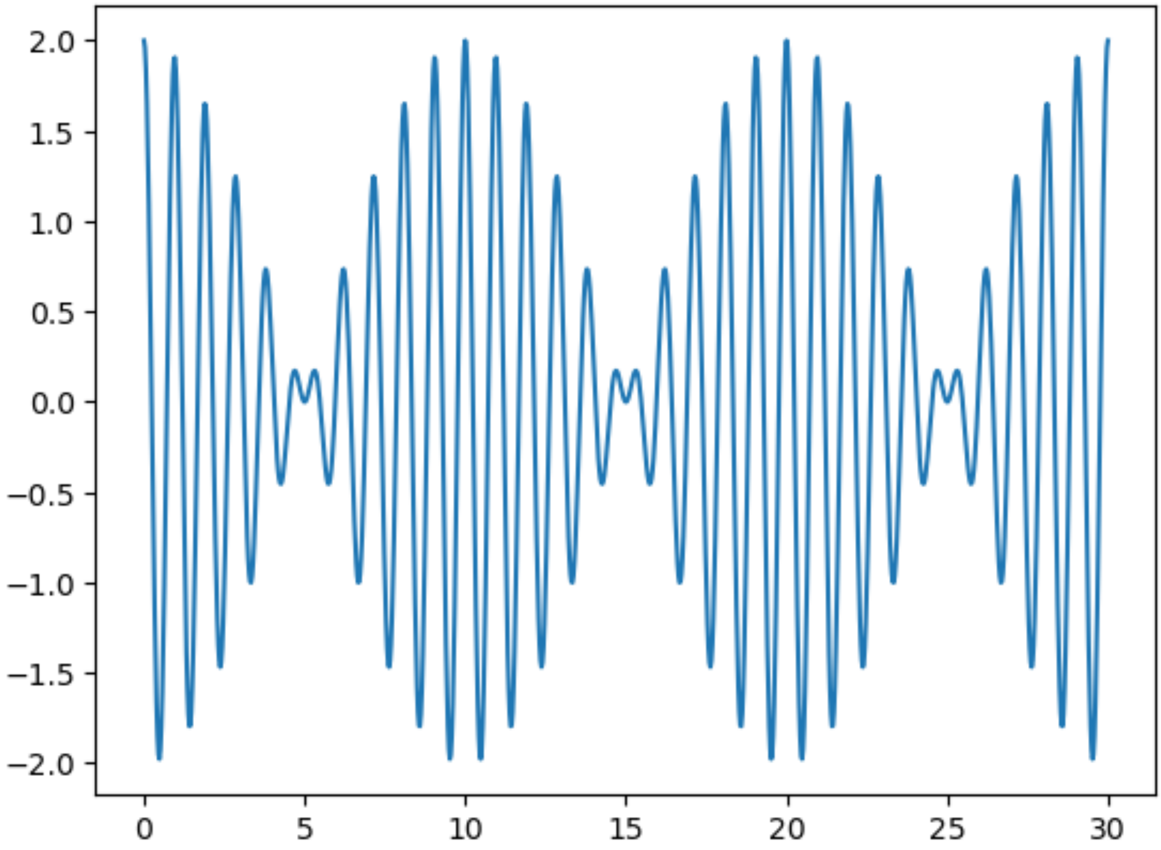
```
Nplot = 0
Nplot = 1
Nplot = 2
Nplot = 3
Nplot = 4
Nplot = 5
Nplot = 6
Nplot = 7
Nplot = 8
Nplot = 9
Nplot = 10
Nplot = 11
Nplot = 12
Nplot = 13
Nplot = 14
Its done!
<Figure size 640x480 with 0 Axes>
```

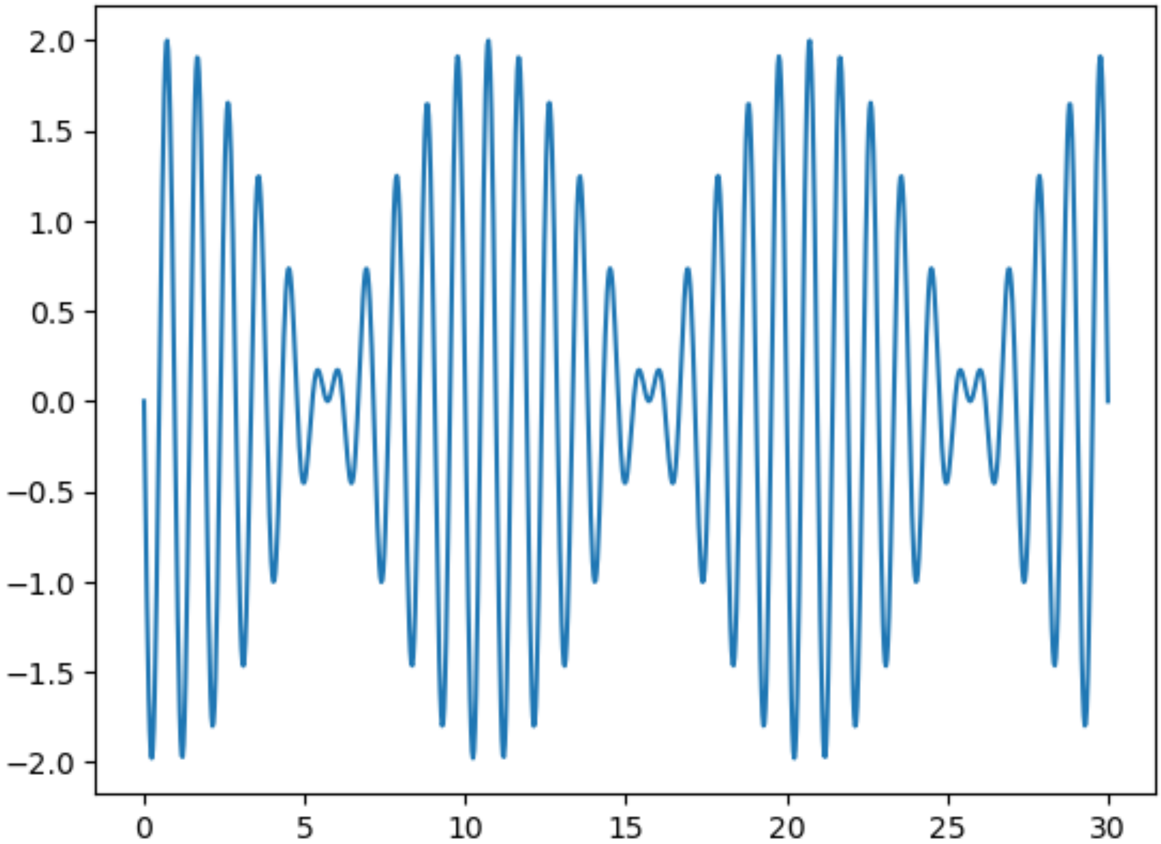
A.2 Wave packets

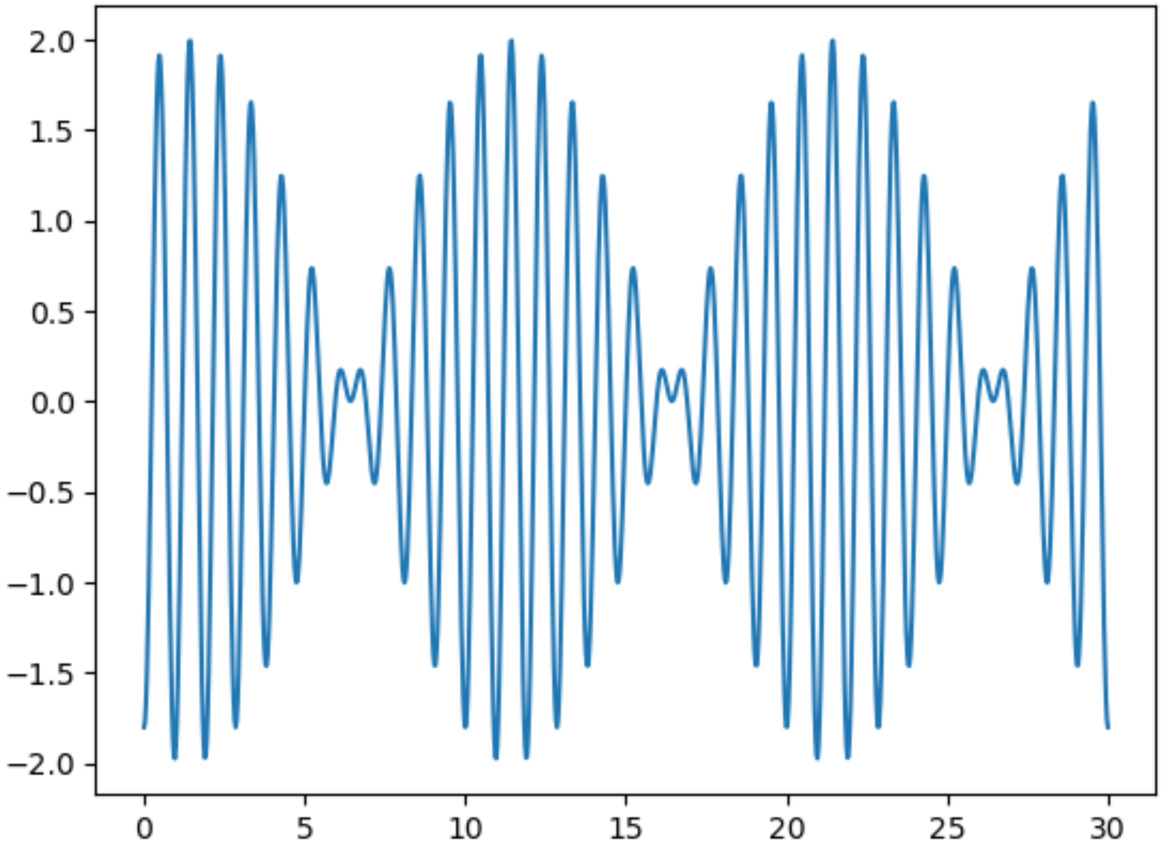
Ex. 13 Superposition of 2 waves

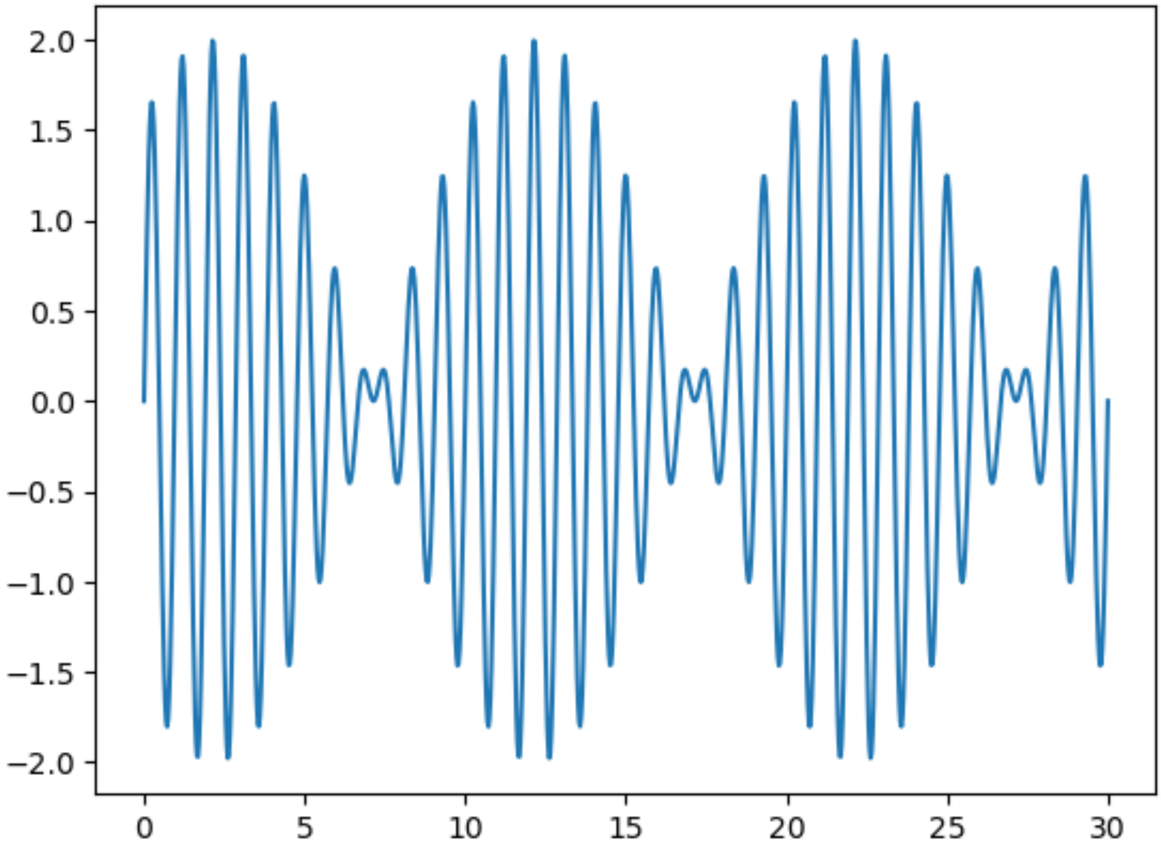
1. $\psi_1(x,t) = A\cos(2\pi f_1 t - \frac{2\pi f_1}{v_1}x)$ Same thing for ψ_2

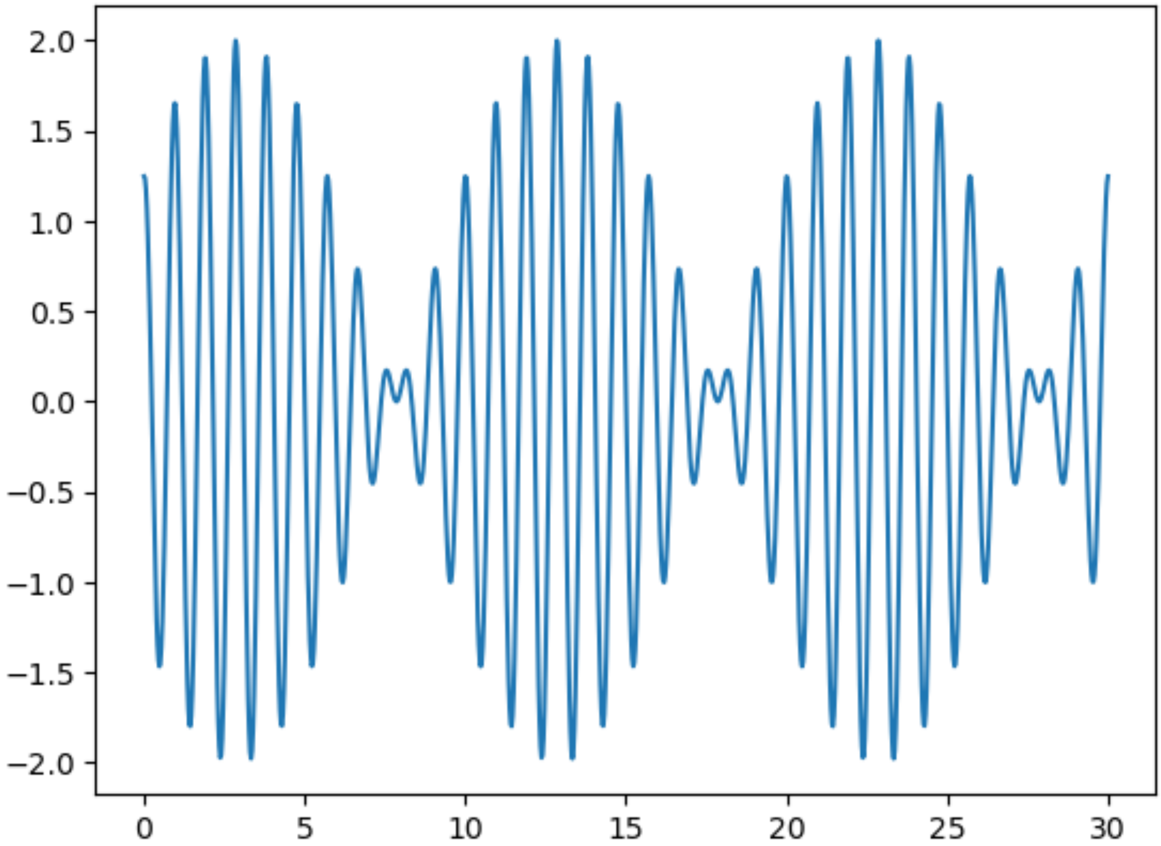
In [ ]:
```
v1 = 1
v2 = 1

f1 = 1
f2 = 1.1

x = np.linspace(0, 30, 1000)
t_range = np.linspace(0, 10, 15)

for i in range(len(t_range)):
    t = t_range[i]
    psi_1 = A*np.cos(2*np.pi*f1*(t - 1/v1*x))
    psi_2 = A*np.cos(2*np.pi*f2*(t - 1/v2*x))
    psi = psi_1 + psi_2
    plt.plot(x, psi)
    plt.show()
    plt.clf()
```

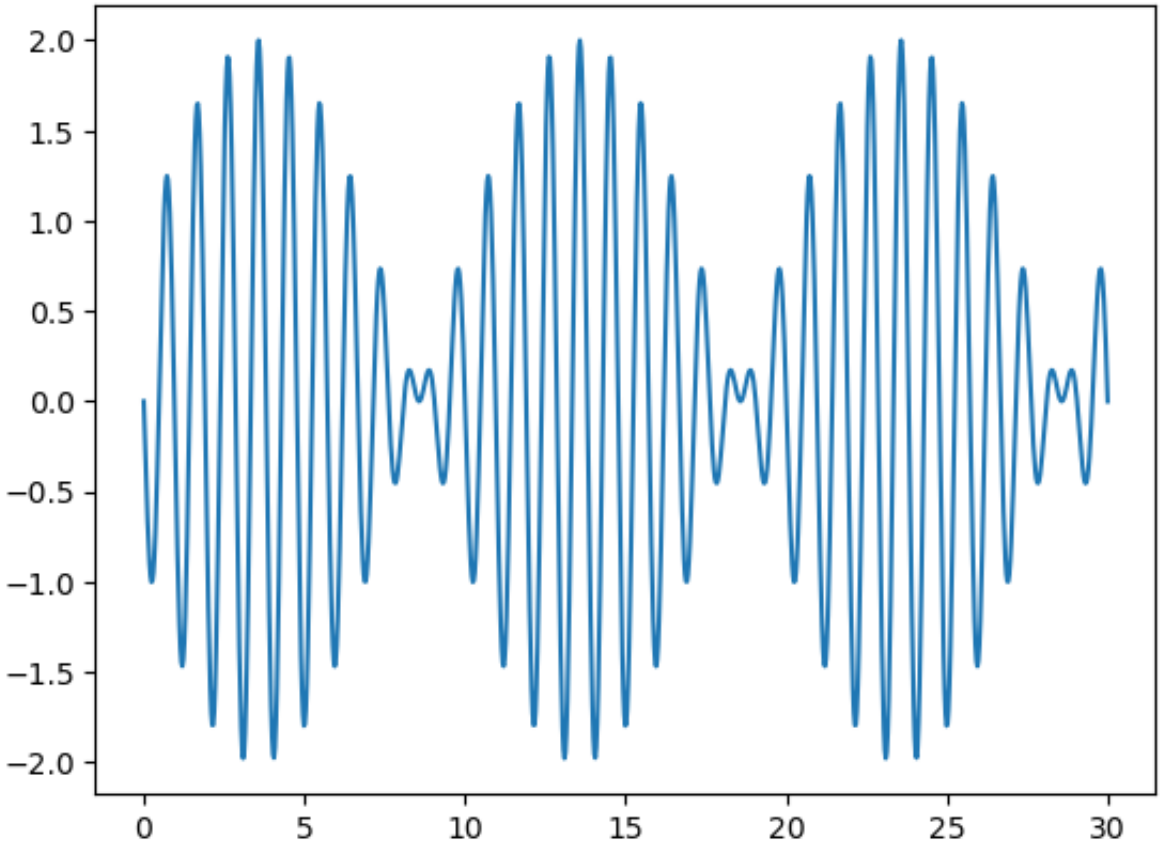


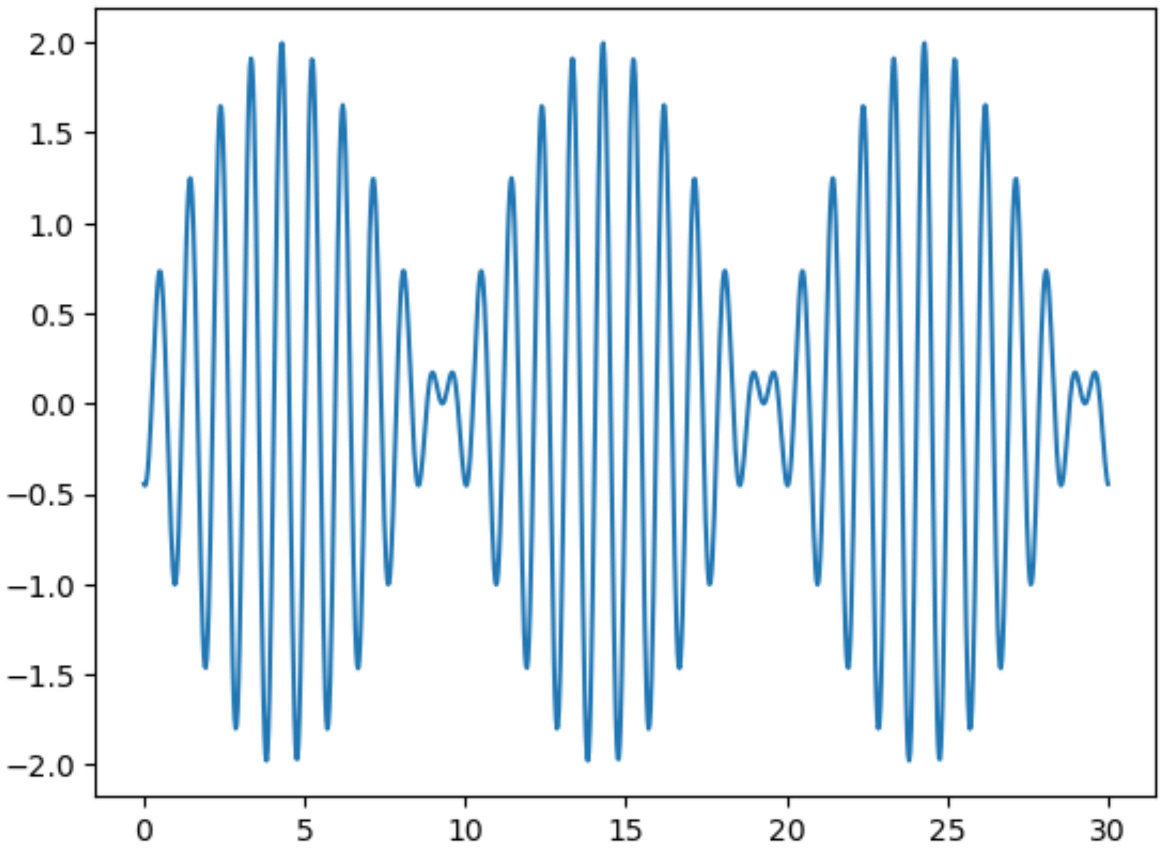

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30

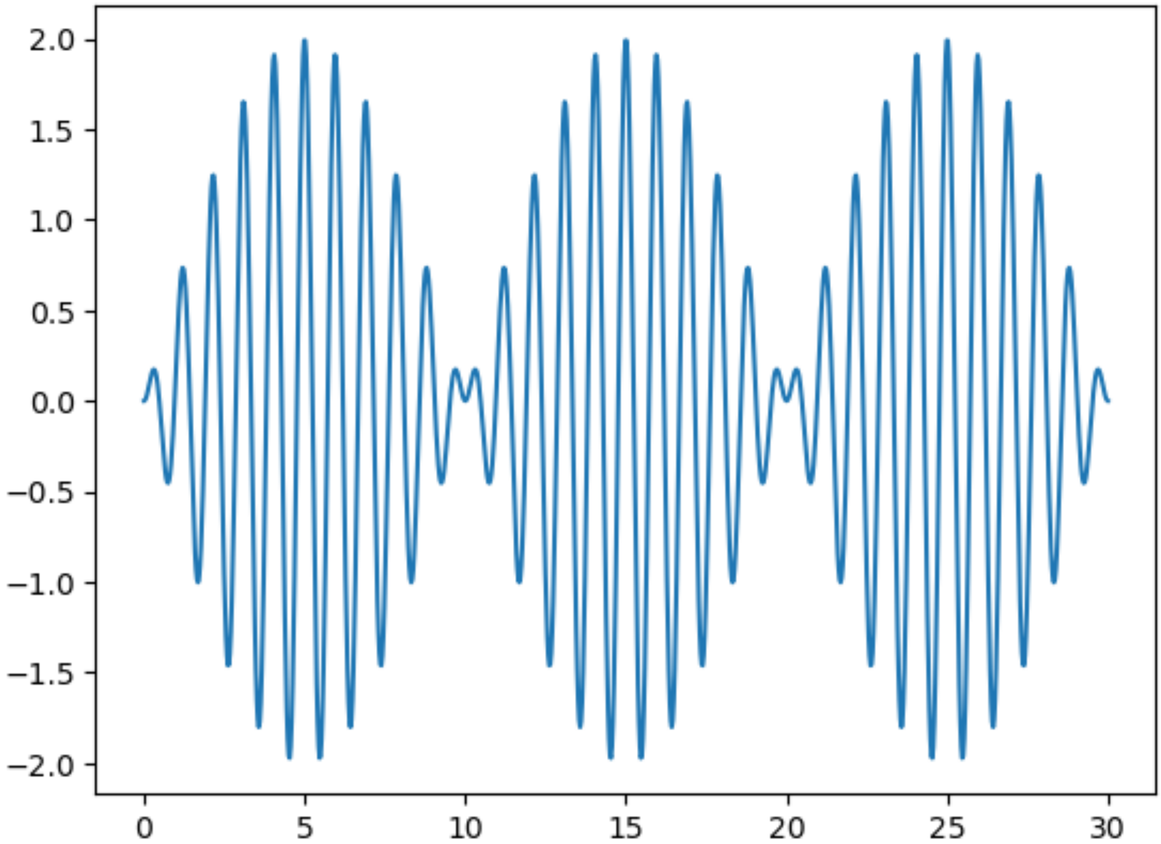

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30

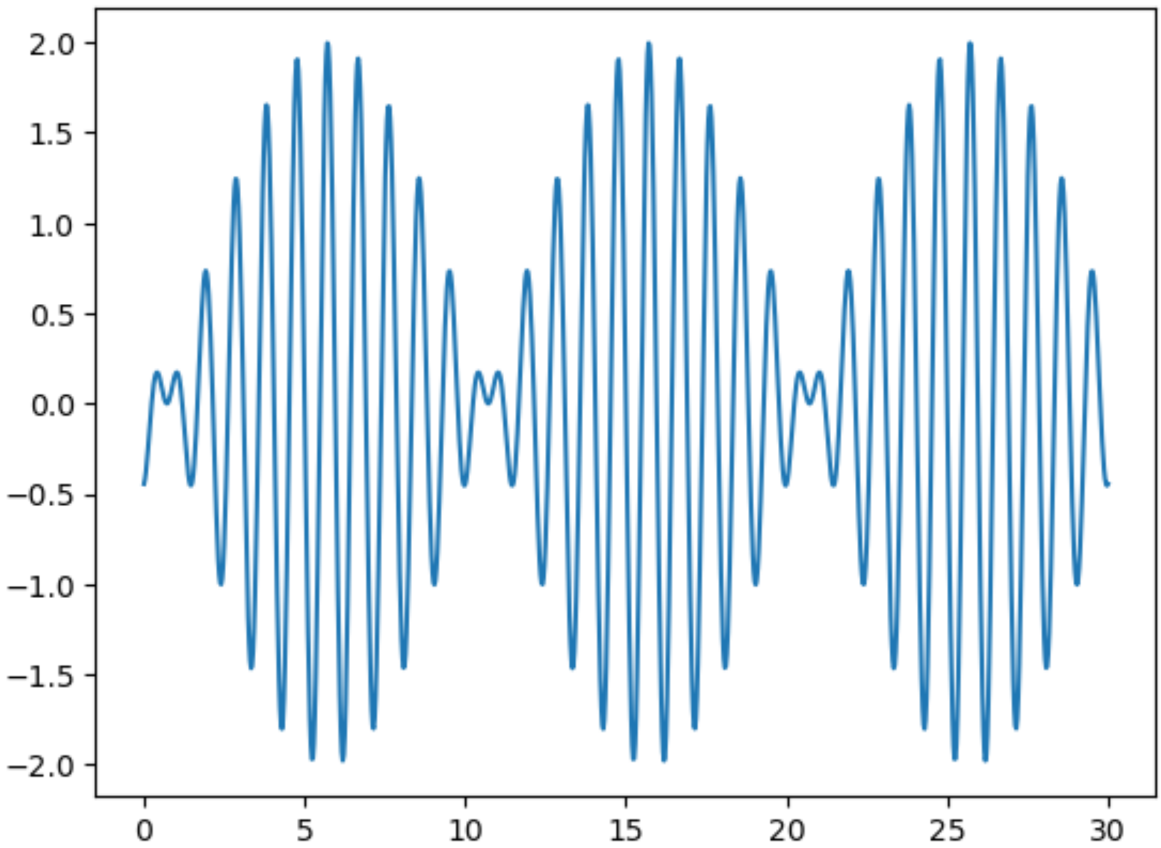

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30

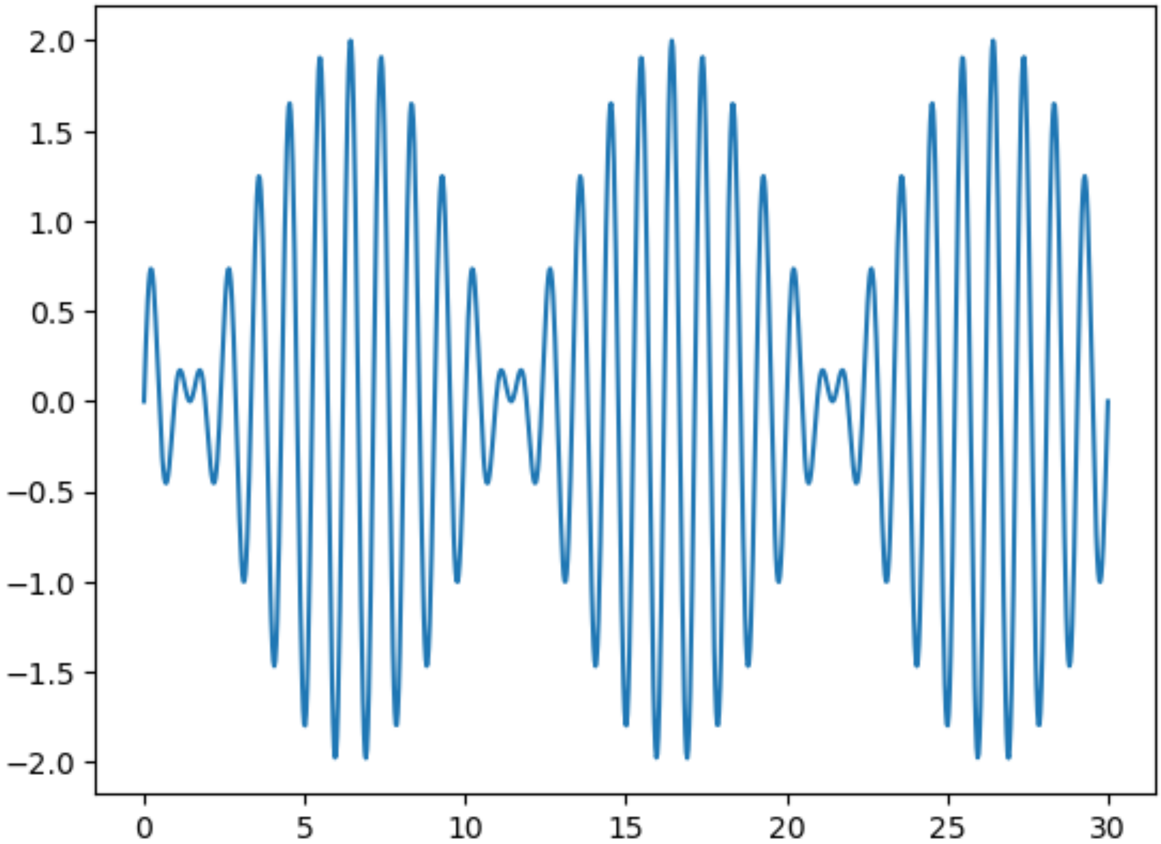

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30

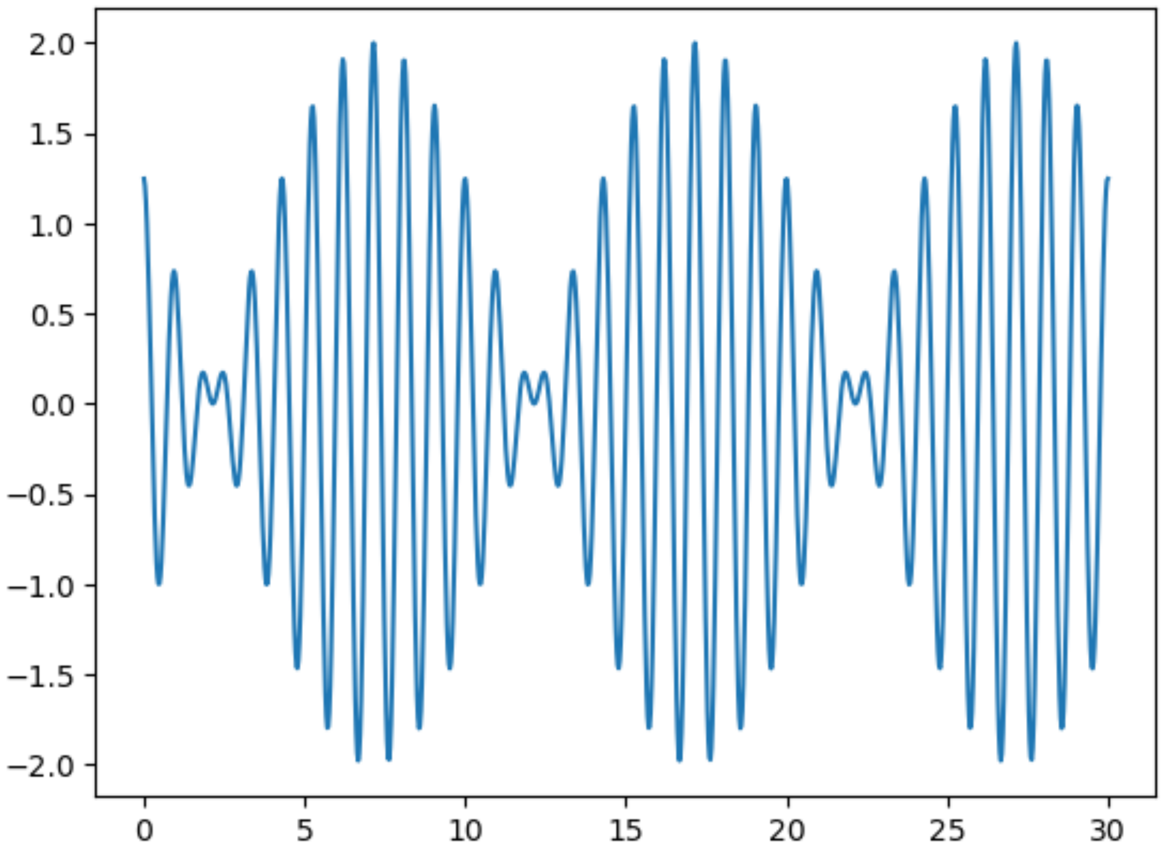

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0 5 10 15 20 25 30

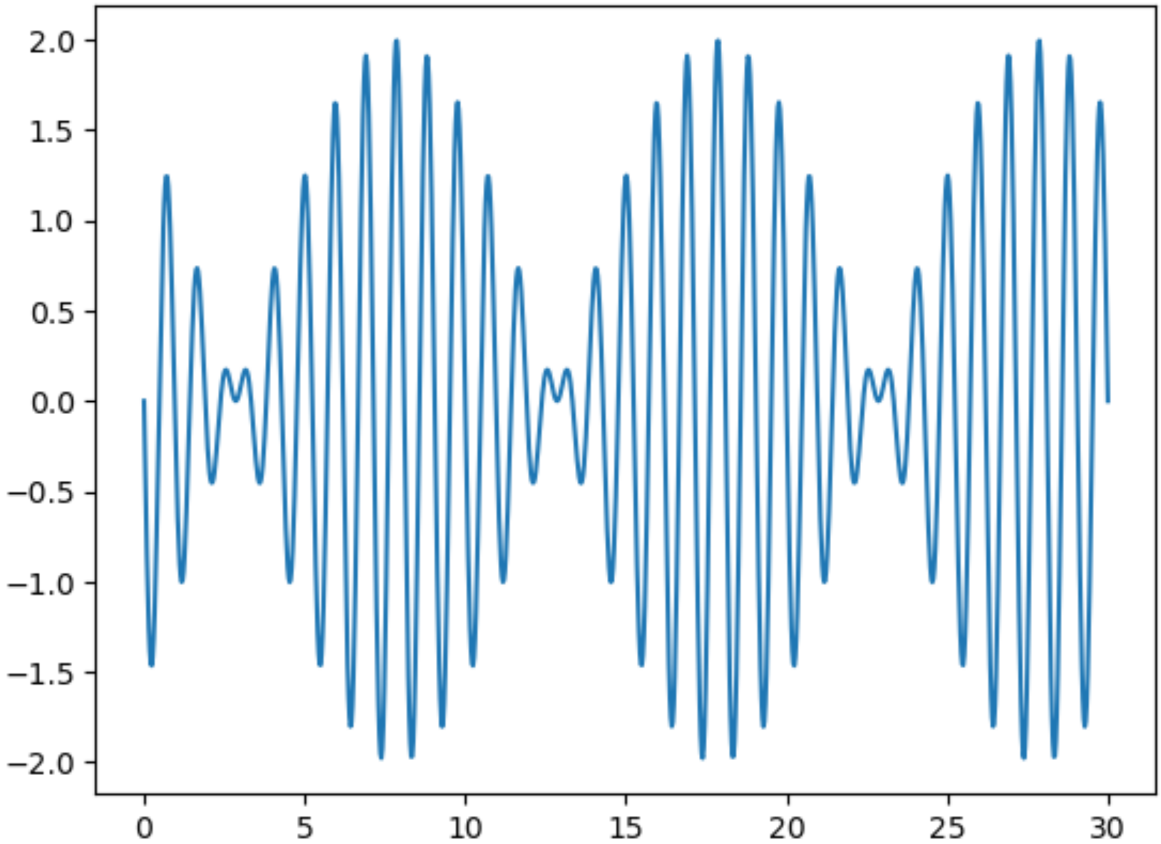

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0 5 10 15 20 25 30

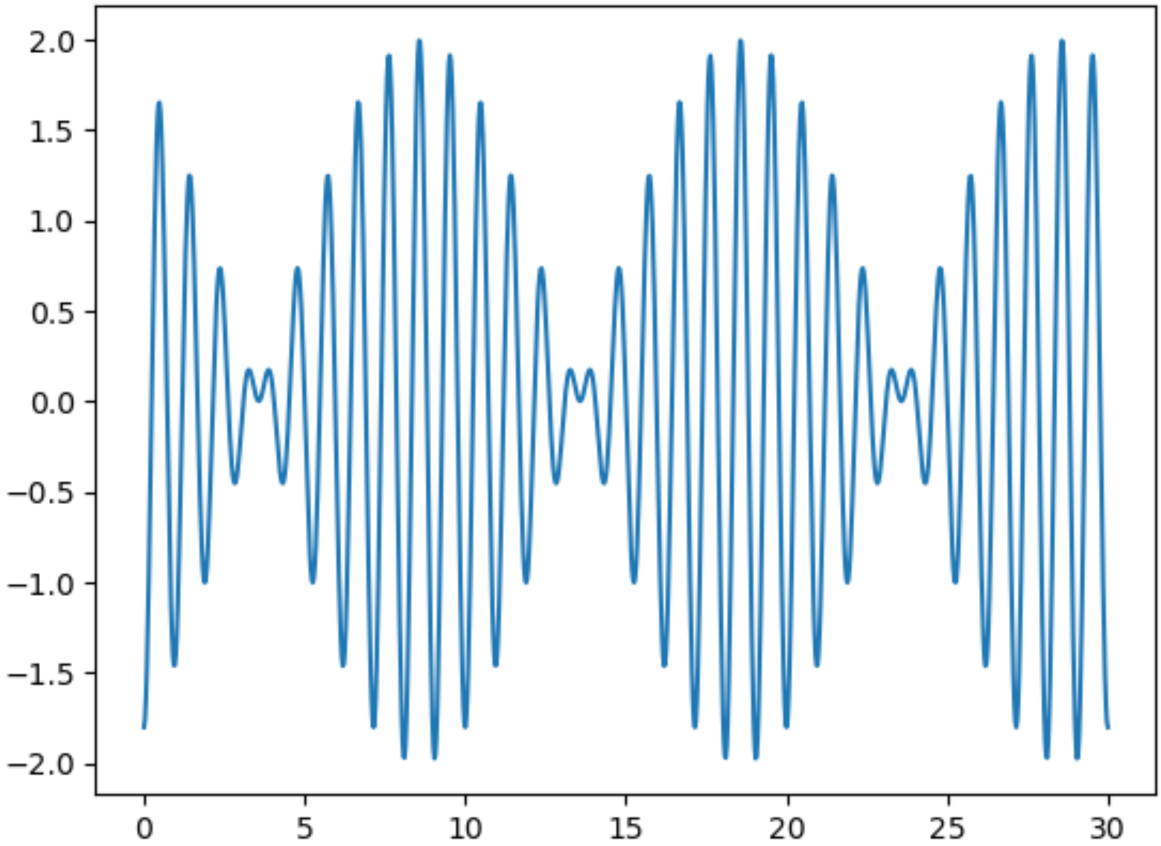

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30

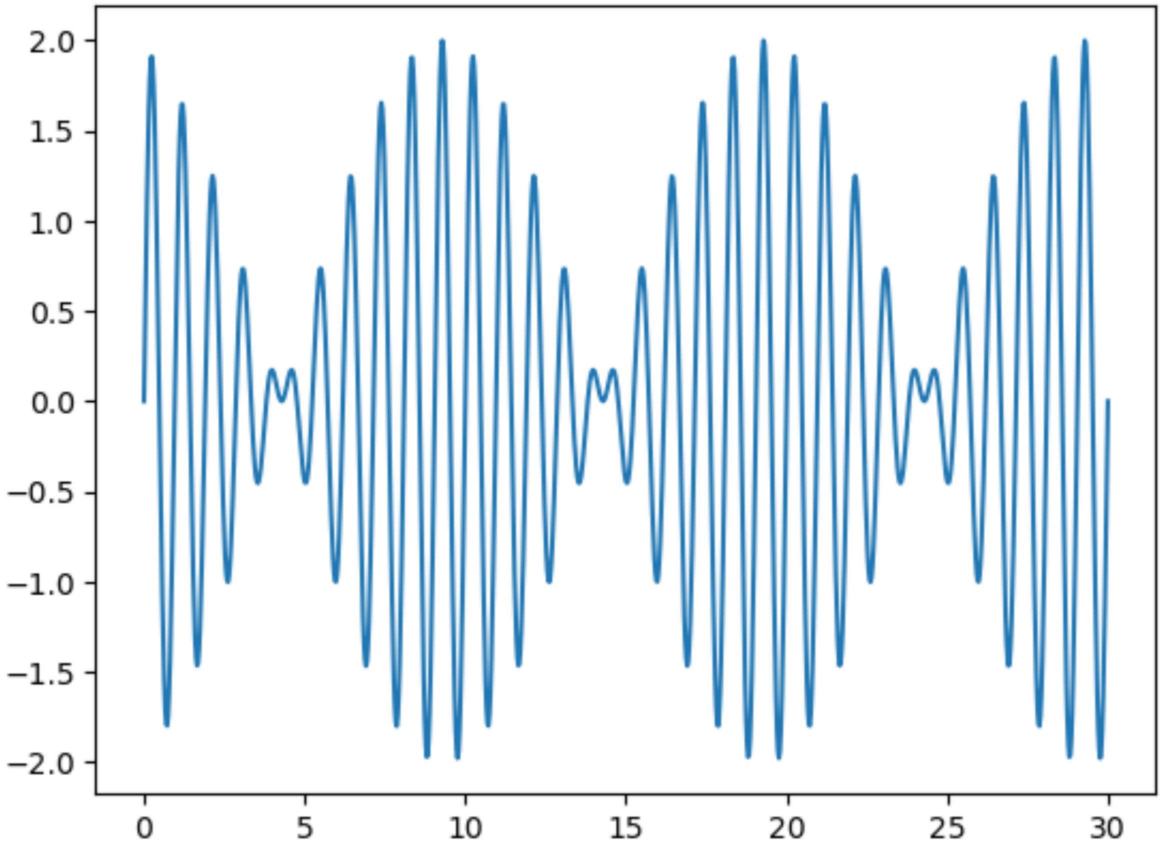

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30

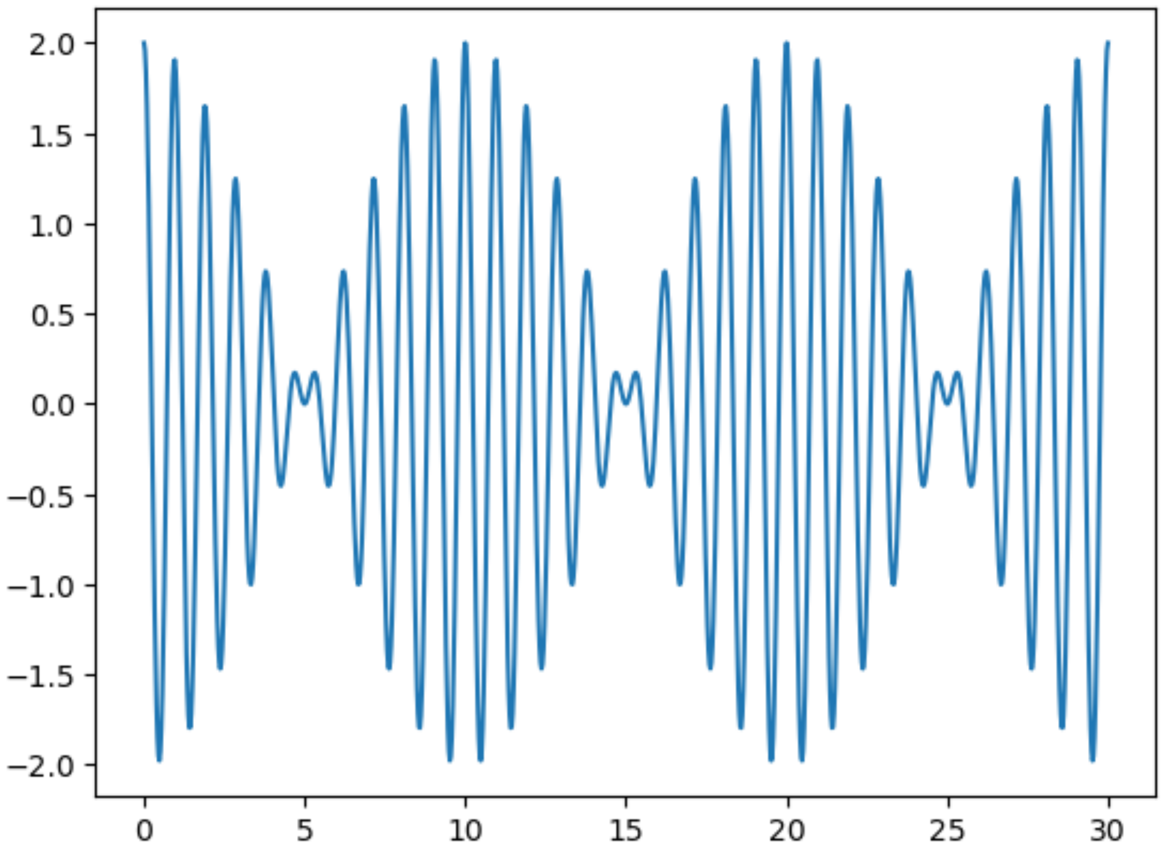

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30






```
<Figure size 640x480 with 0 Axes>
```

In [ ]:
```
Xmin, Xmax, Ymin, Ymax = 0, 30, -2, 2
x = np.linspace(0, 30, 1000)
t_range = np.linspace(0, 10, 8)

for i in range(len(t_range)):
    t = t_range[i]
    psi_1 = A*np.cos(2*np.pi*f1*(t - 1/v1*x))
    psi_2 = A*np.cos(2*np.pi*f2*(t - 1/v2*x))
    psi = psi_1 + psi_2
    pylab.plot(x, psi)
    pylab.axis([Xmin, Xmax, Ymin, Ymax])

    filename = 'figs/fichierTemp'+str('%02d' %i)+'.pdf' # creating file f
    pylab.savefig(filename)
    print(f"Nplot = {i}")
    pylab.clf()

# assemble images into an animation
cmd = "convert -delay 50 -loop 0 figs/fichierTemp*.pdf figs/wave_packets_
os.system(cmd)
os.system(f"rm figs/fichierTemp*.pdf")
print("Its done!")
```

```
Nplot = 0
Nplot = 1
Nplot = 2
Nplot = 3
Nplot = 4
Nplot = 5
Nplot = 6
Nplot = 7
Its done!
<Figure size 640x480 with 0 Axes>
```

In [ ]:
```
v1 = 1
v2 = 1.3

f1 = 1
f2 = 1.1

Xmin, Xmax, Ymin, Ymax = 0, 30, -2, 2
x = np.linspace(0, 30, 1000)
t_range = np.linspace(0, 10, 8)

for i in range(len(t_range)):
    t = t_range[i]
    psi_1 = A*np.cos(2*np.pi*f1*(t - 1/v1*x))
    psi_2 = A*np.cos(2*np.pi*f2*(t - 1/v2*x))
    psi = psi_1 + psi_2
    pylab.plot(x, psi)
    pylab.axis([Xmin, Xmax, Ymin, Ymax])

    filename = 'figs/fichierTemp'+str('%02d' %i)+'.pdf' # creating file f
    pylab.savefig(filename)
    print(f"Nplot = {i}")
    pylab.clf()

# assemble images into an animation
cmd = "convert -delay 50 -loop 0 figs/fichierTemp*.pdf figs/wave_packets_
os.system(cmd)
os.system(f"rm figs/fichierTemp*.pdf")
print("Its done!")
```

```
Nplot = 0
Nplot = 1
Nplot = 2
Nplot = 3
Nplot = 4
Nplot = 5
Nplot = 6
Nplot = 7
Its done!
<Figure size 640x480 with 0 Axes>
```

Theory

$$v_g = \frac{\delta\omega}{\delta k} = \frac{\omega_2 - \omega_1}{k_2 - k_2} = \frac{f_2 - f_1}{f_2/v_2 - f_1/v_1} = \frac{1.1 - 1}{1.1/1.3 - 1/1} = -0.65$$

from the gif

Tracking the same maxima between the first and last frame (10 s)

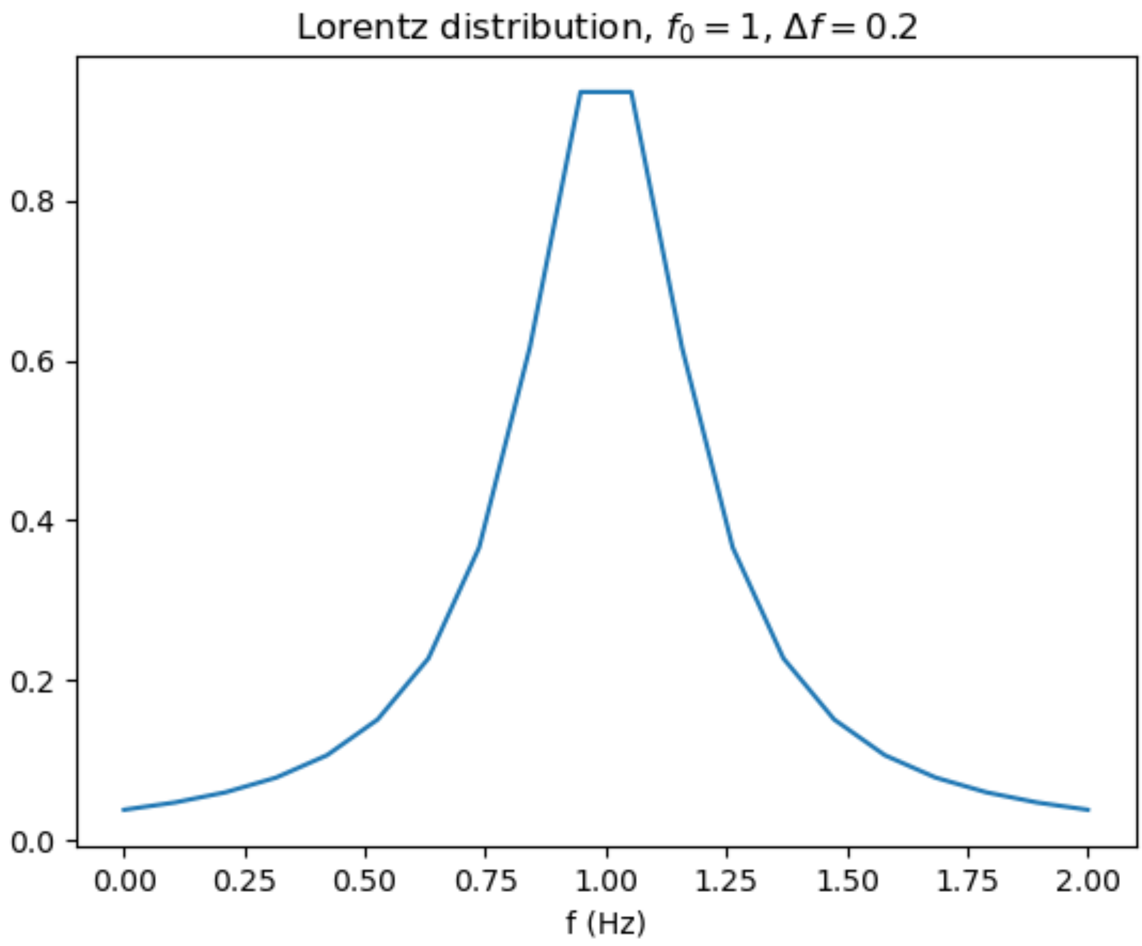
$d \approx -7, t = 10$ Therefore, $v_{g,exp} \approx -0.7$

Ex. 14 Superposition of N waves

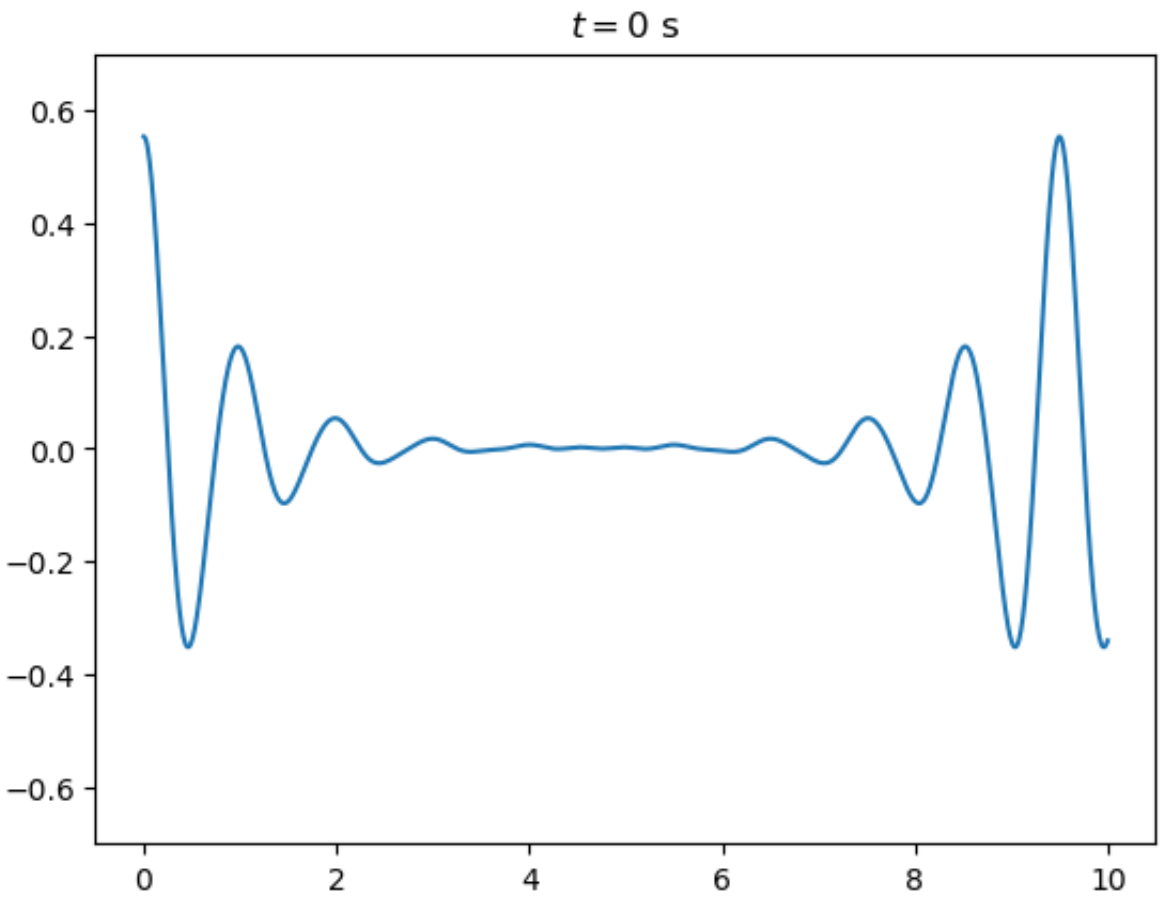
Non-dispersif medium

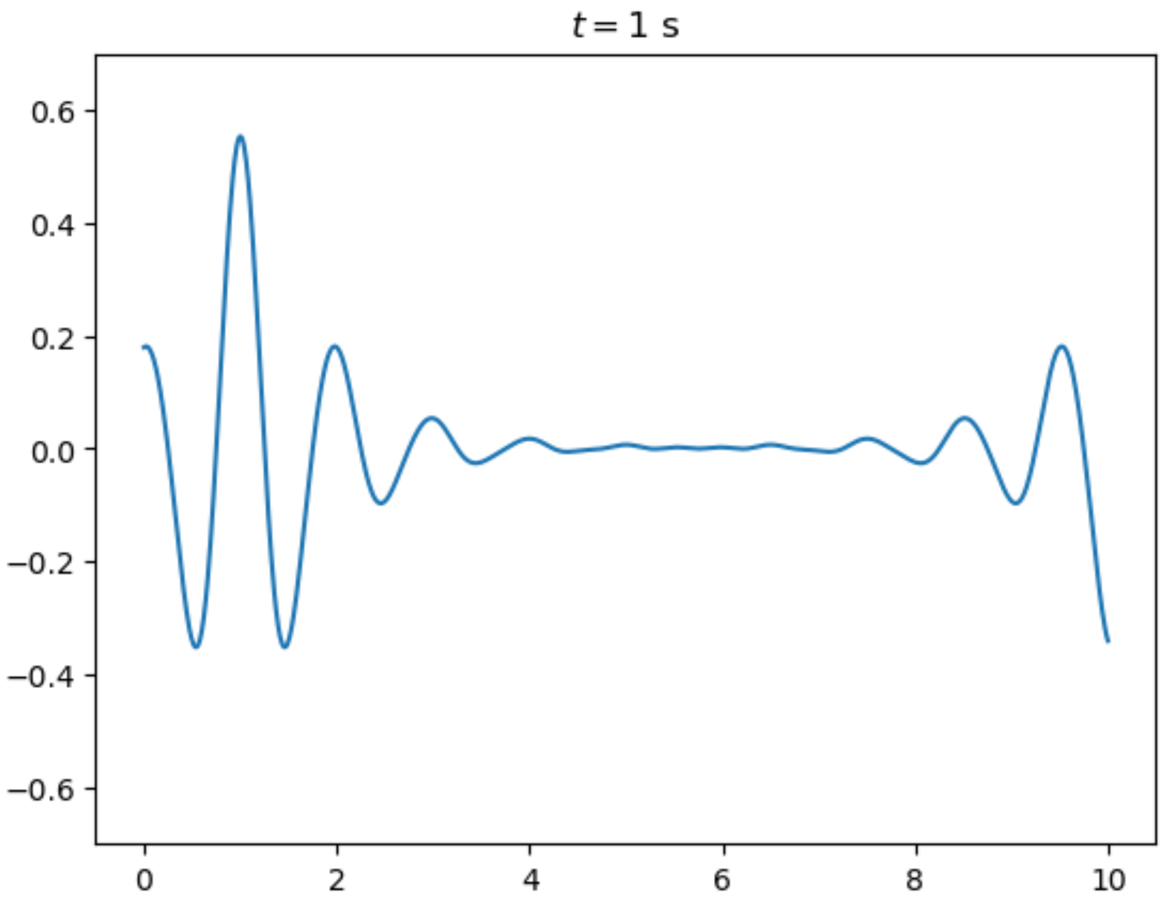
N = 20

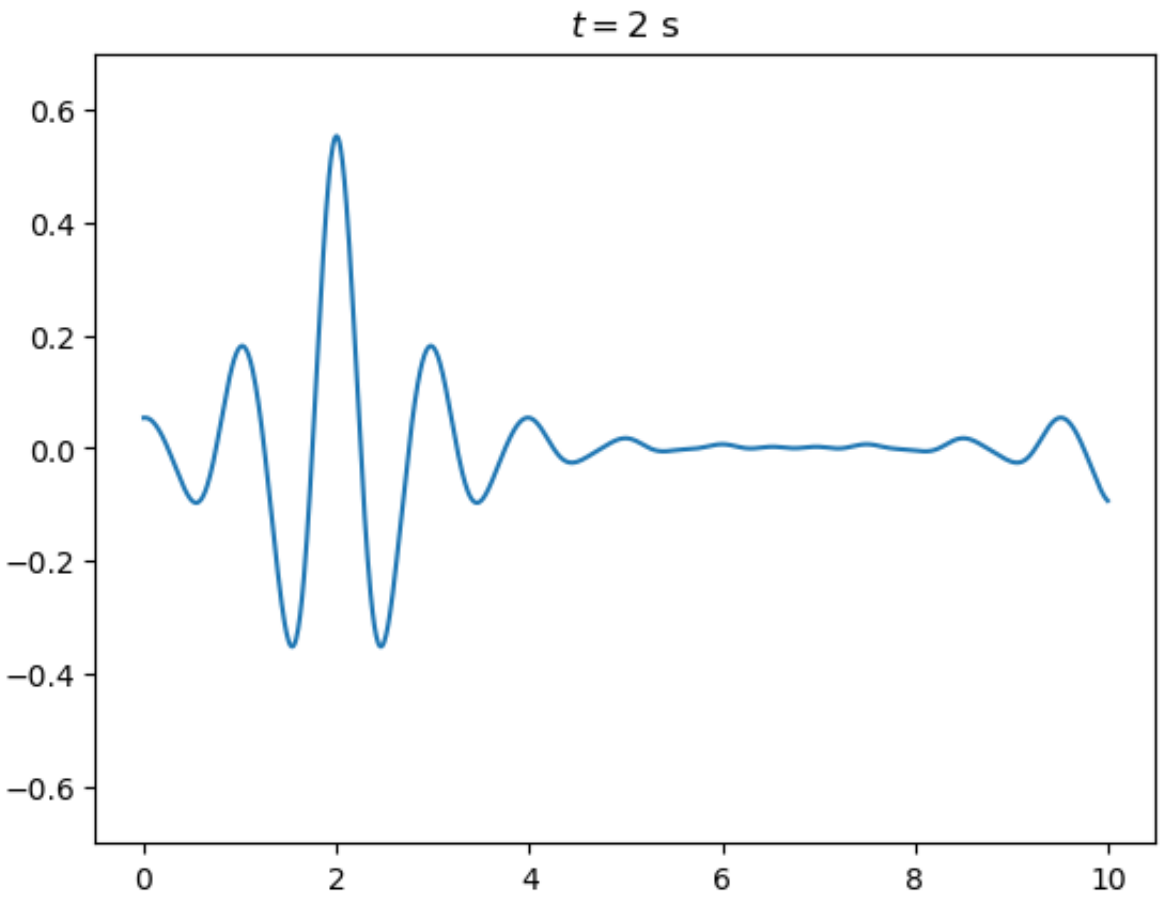
In [ ]:
```
def g(f, f_0, D_f):
    A = 1
    return A / (1 + ((f-f_0)/D_f)**2)

f_0 = 1
f_width = 0.2
N = 20

f_domain = np.linspace(0, 2*f_0, N)
lorentz_distrib = g(f_domain, f_0, f_width)

plt.plot(f_domain, lorentz_distrib)
plt.title(f"Lorentz distribution, $f_0={f_0}$, $\Delta f={f_width}$")
plt.xlabel("f (Hz)")
#plt.ylabel("")
plt.show()
```

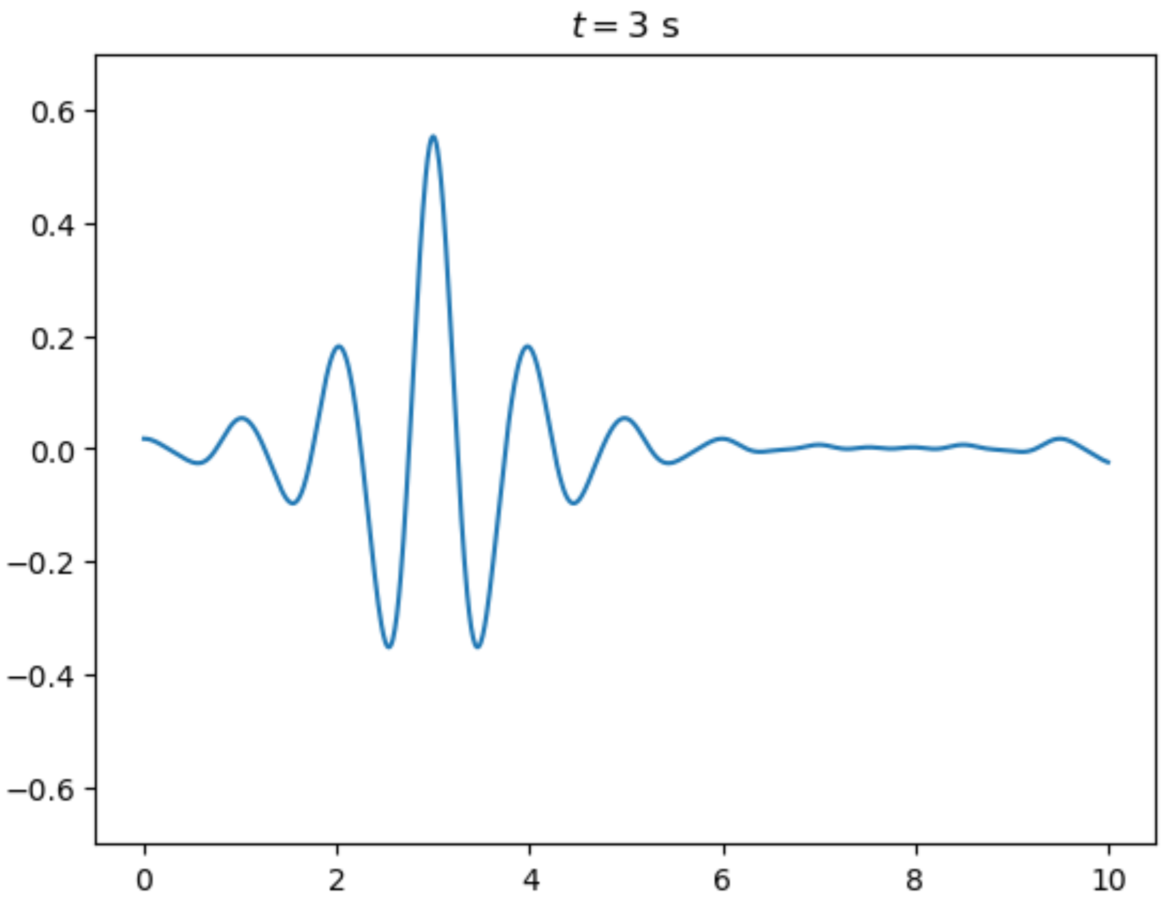


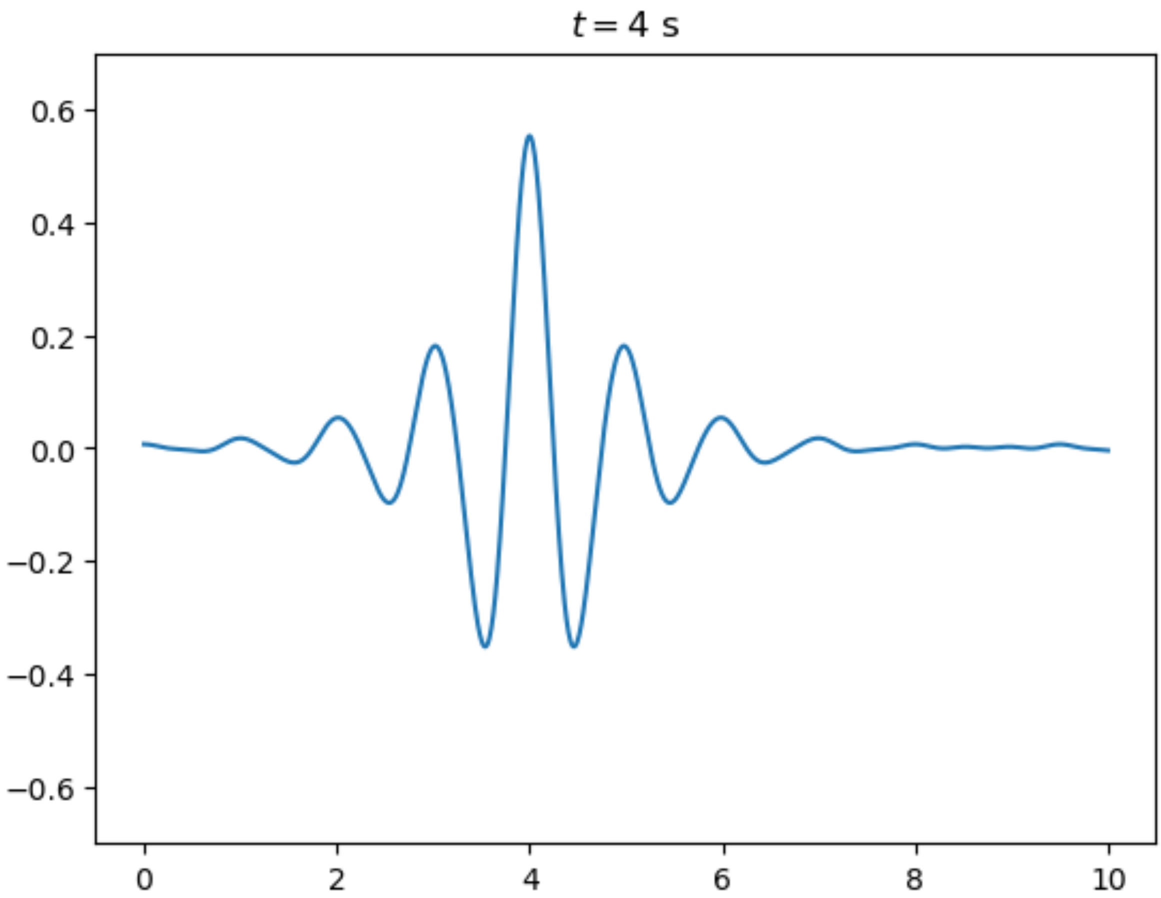
In [ ]:
```
df = f_domain[1]-f_domain[0]
c = 1 # phase speed, light in a vacuum
# fixed t

t_domain = np.arange(0, 15, 1)
for t in t_domain:
#t_1 = 0
    x_domain = np.linspace(0, 10, 1000)

    phi_1 = np.zeros(len(x_domain))
    for i in range(len(x_domain)):
        x = x_domain[i]
        for j in range(len(f_domain)):
            f = f_domain[j]
            phi_1[i] += lorentz_distrib[j]*np.cos(2*np.pi*f*(t - x/c))*df


    plt.plot(x_domain, phi_1)
    plt.title(f"$t={t}$ s")
    plt.ylim((-0.7, 0.7))
    plt.show()
```

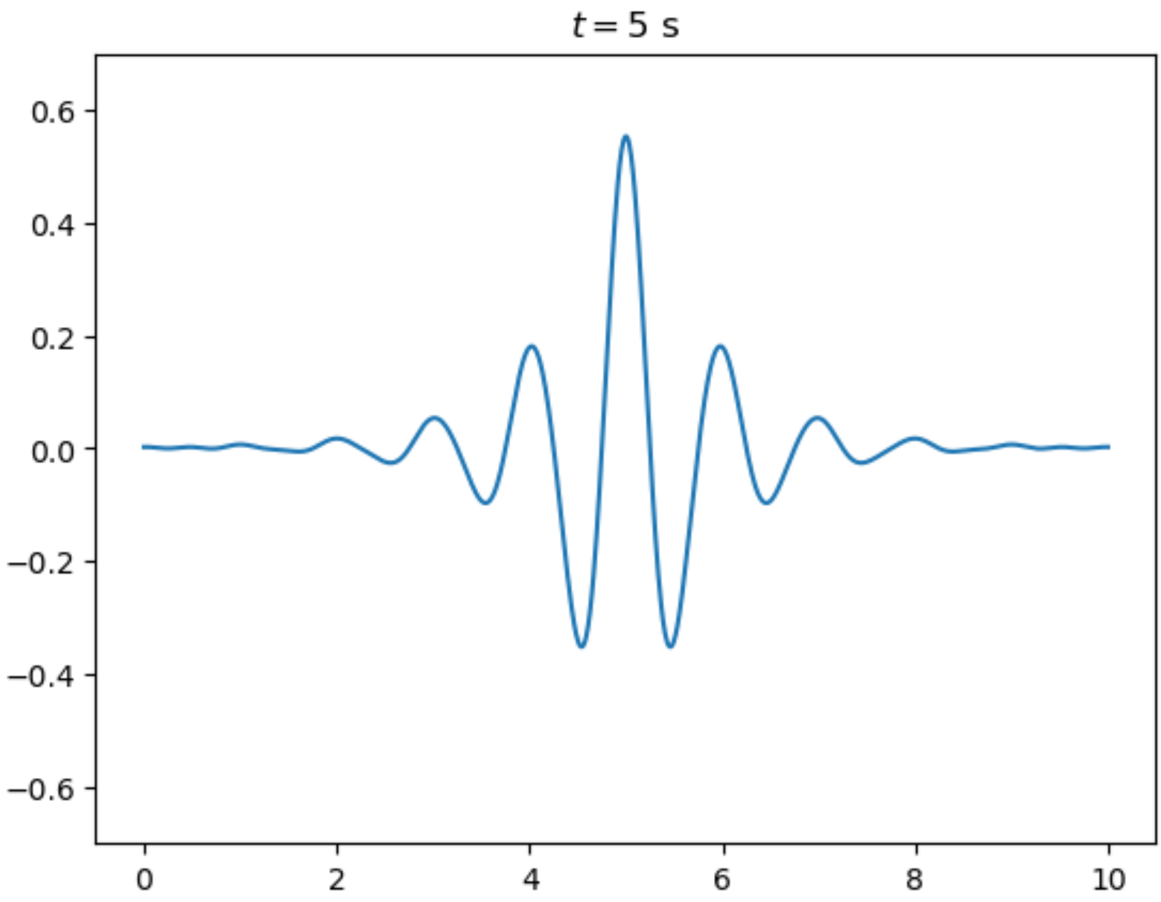


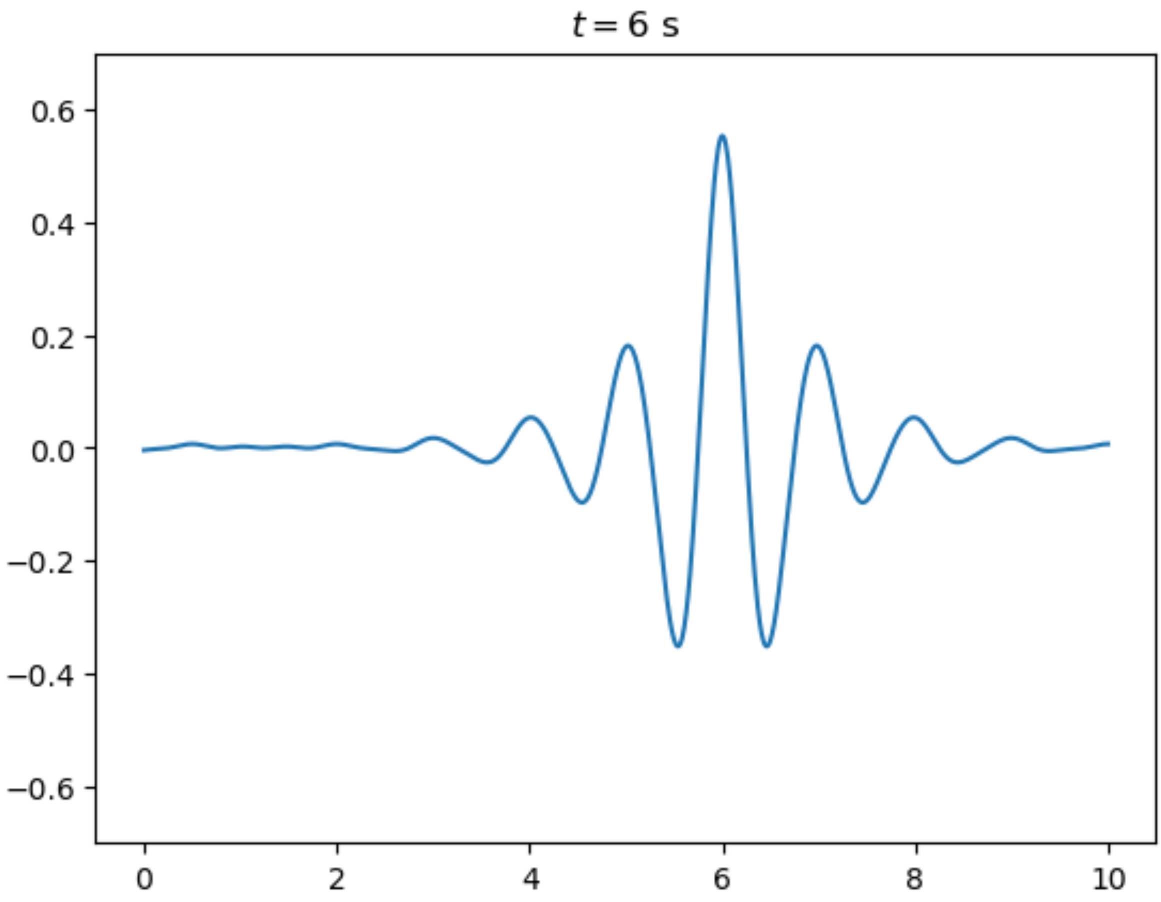

$t = 1$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

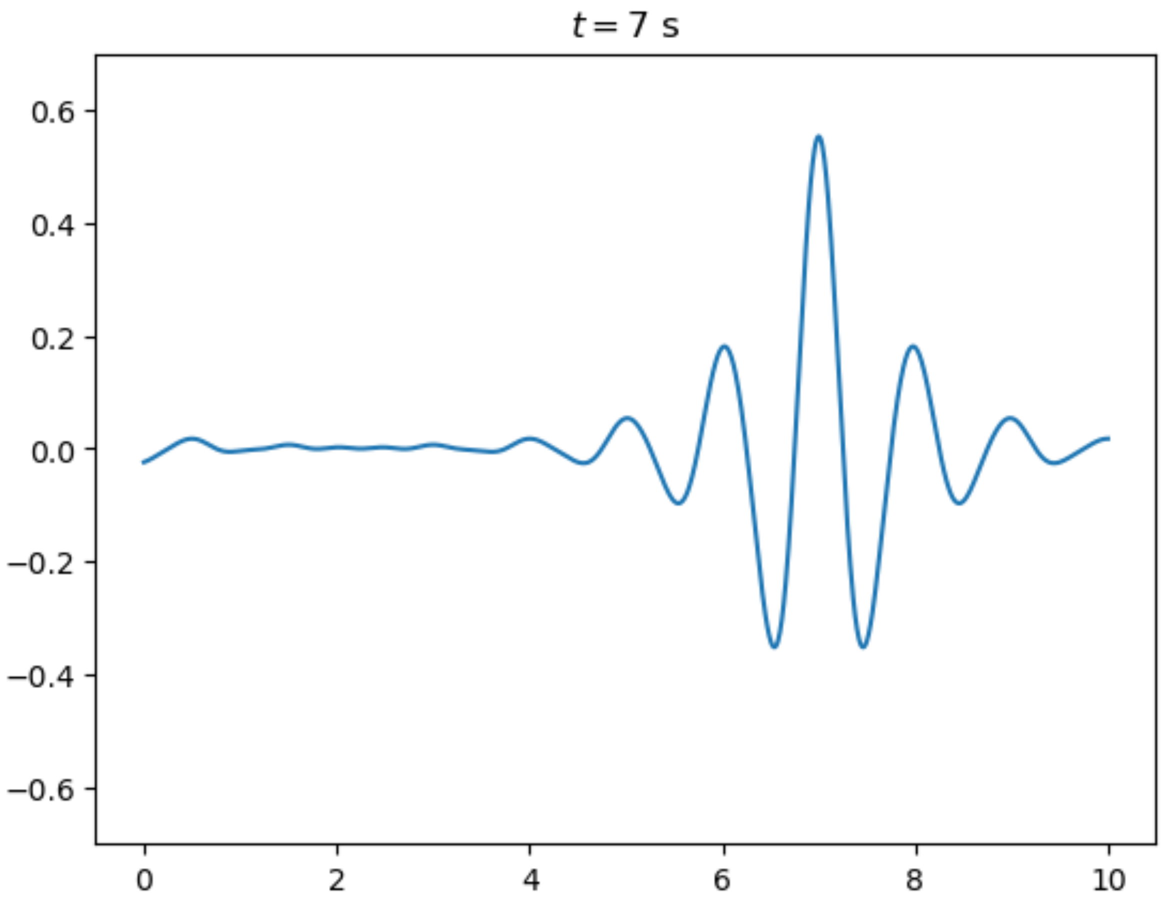

$t = 2$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

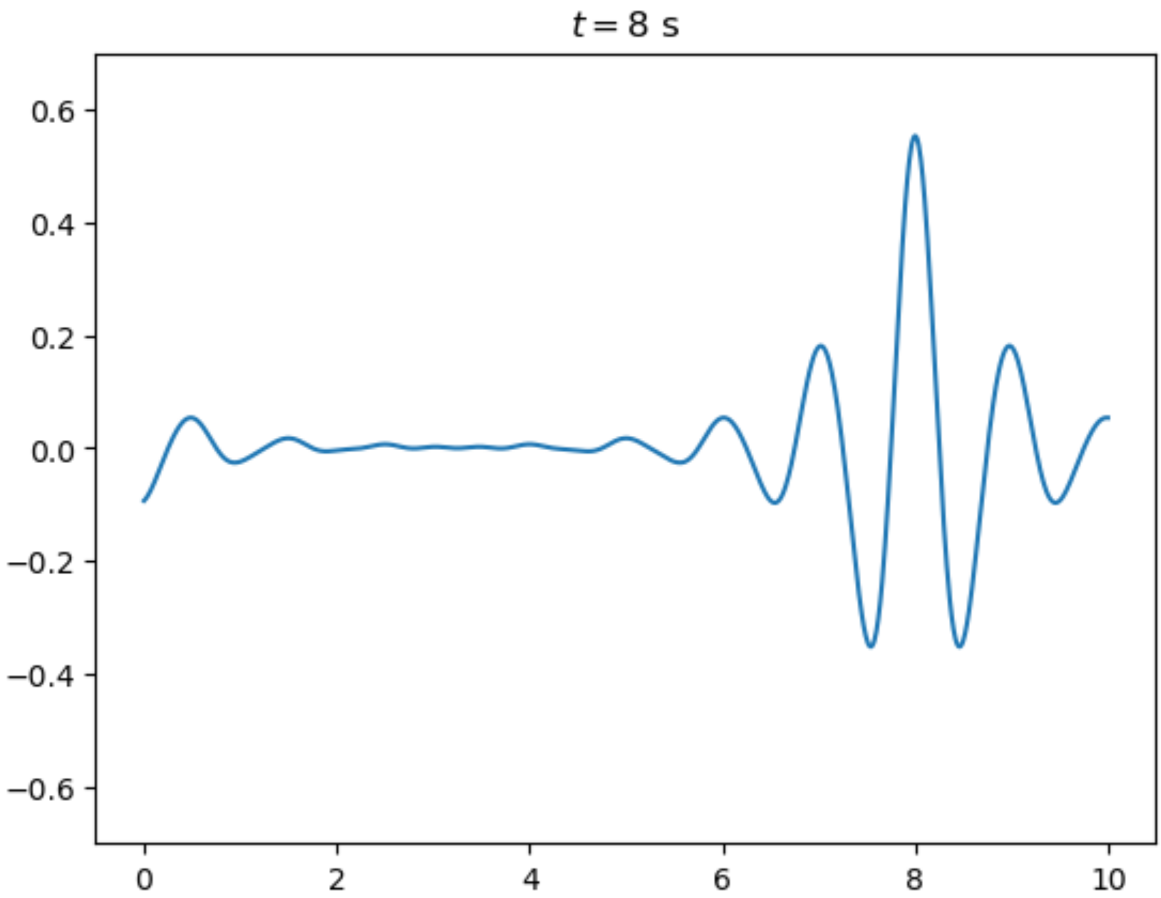

$t = 3$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

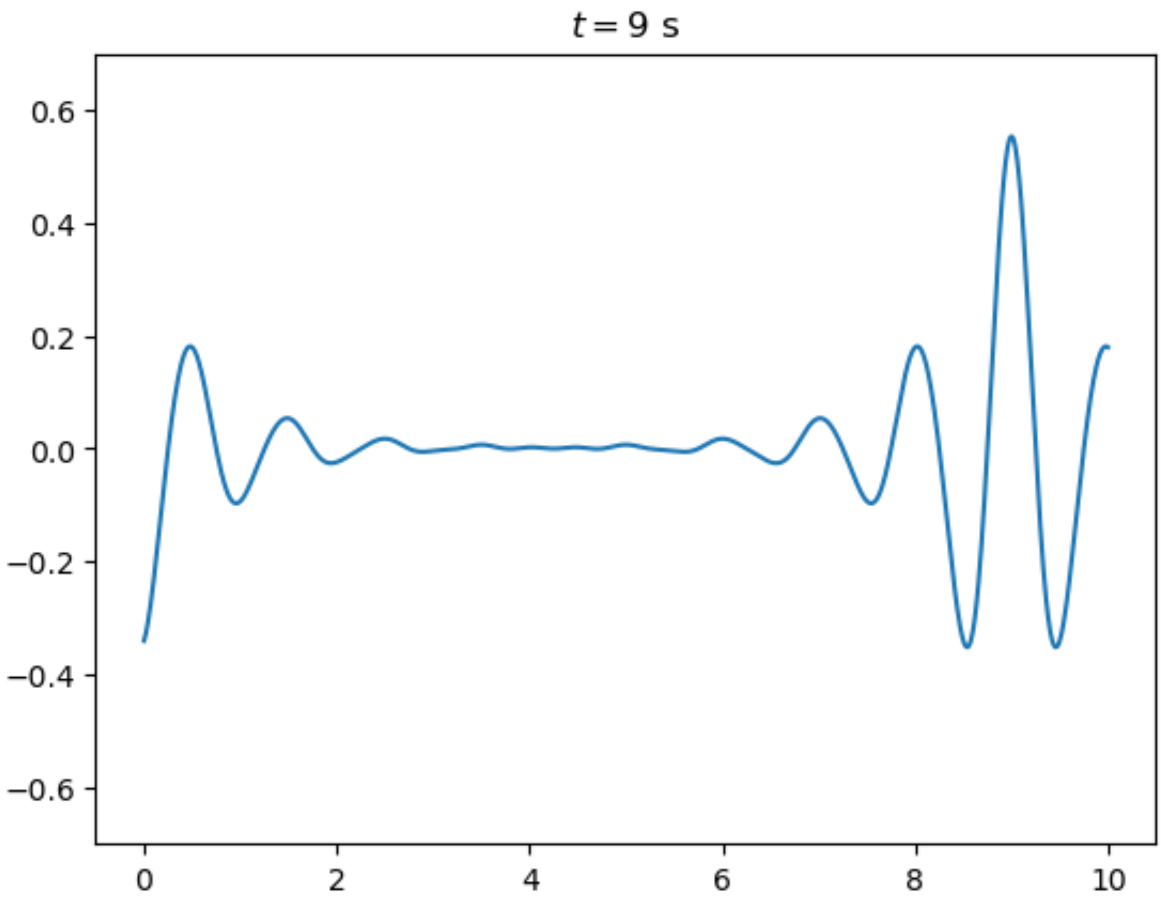

$t = 4$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 5$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

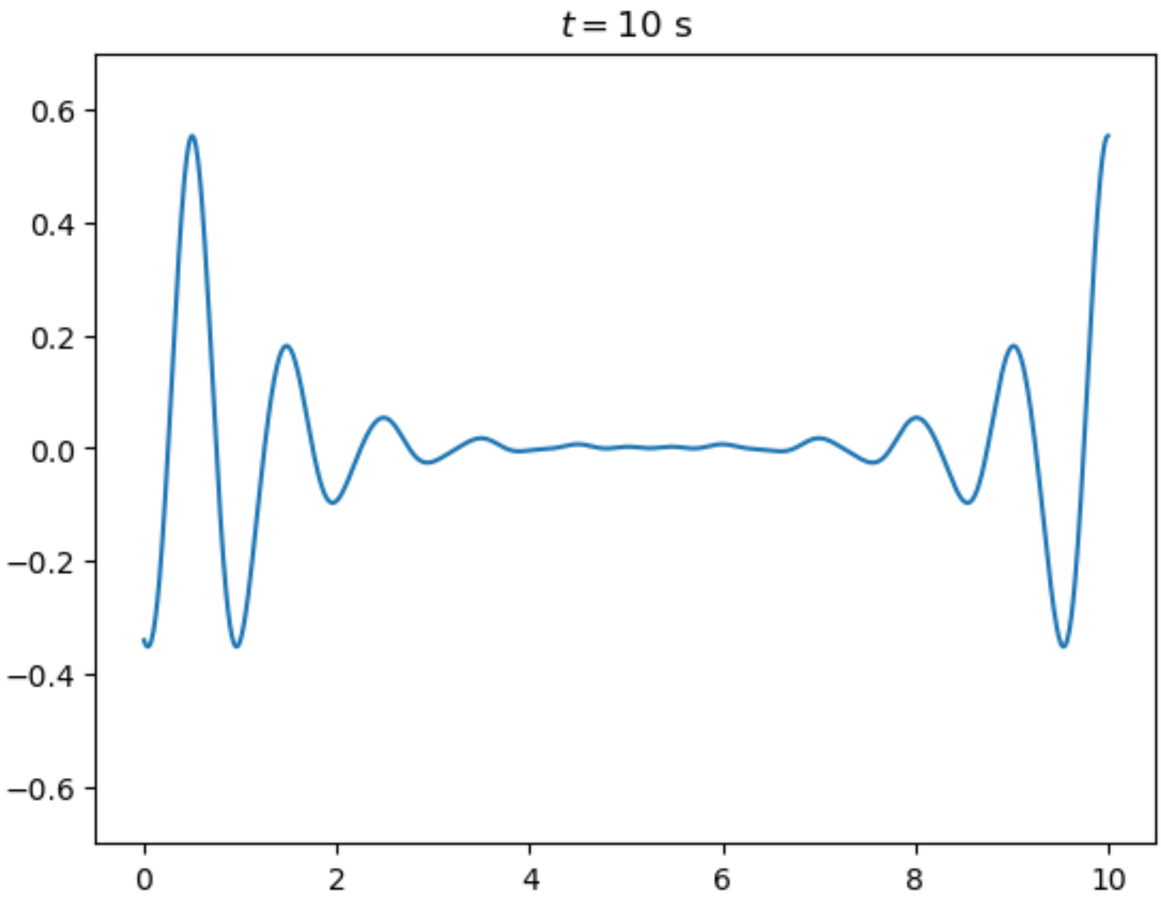

$t = 6$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 7$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 8$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

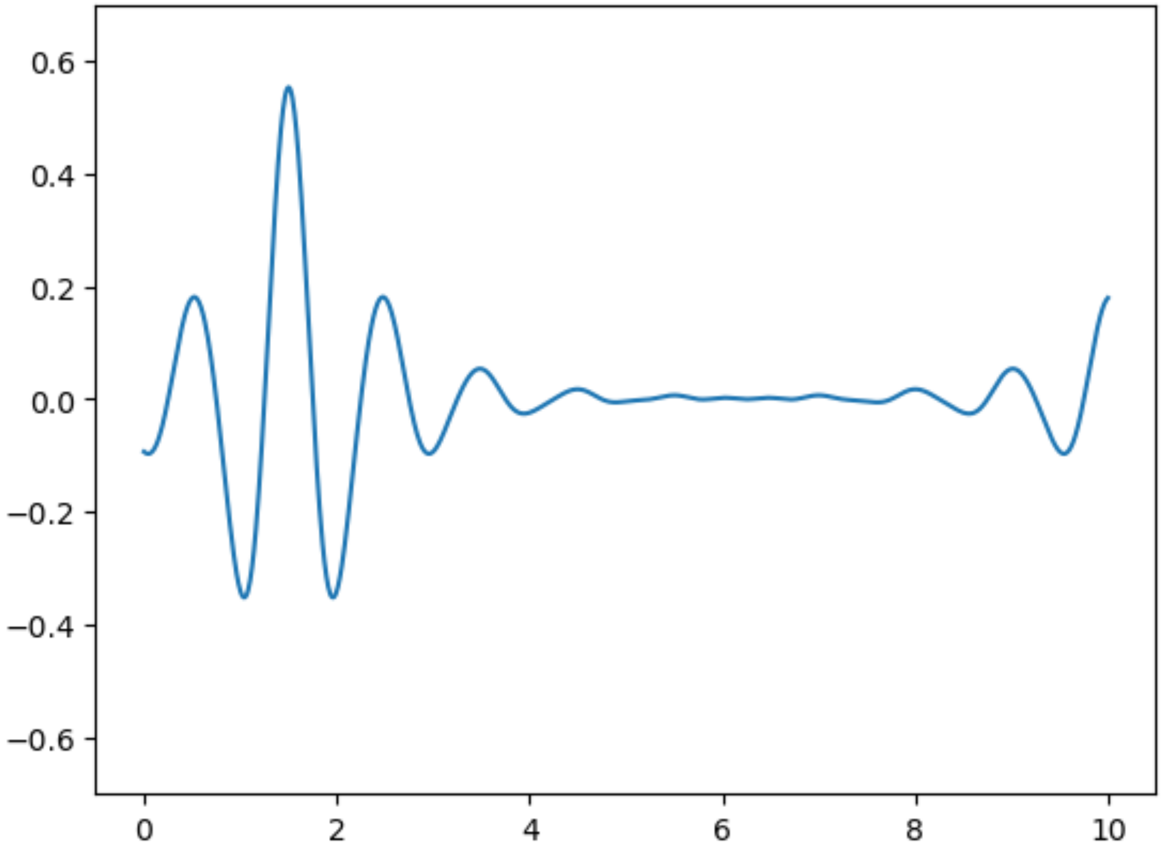


$t = 9$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

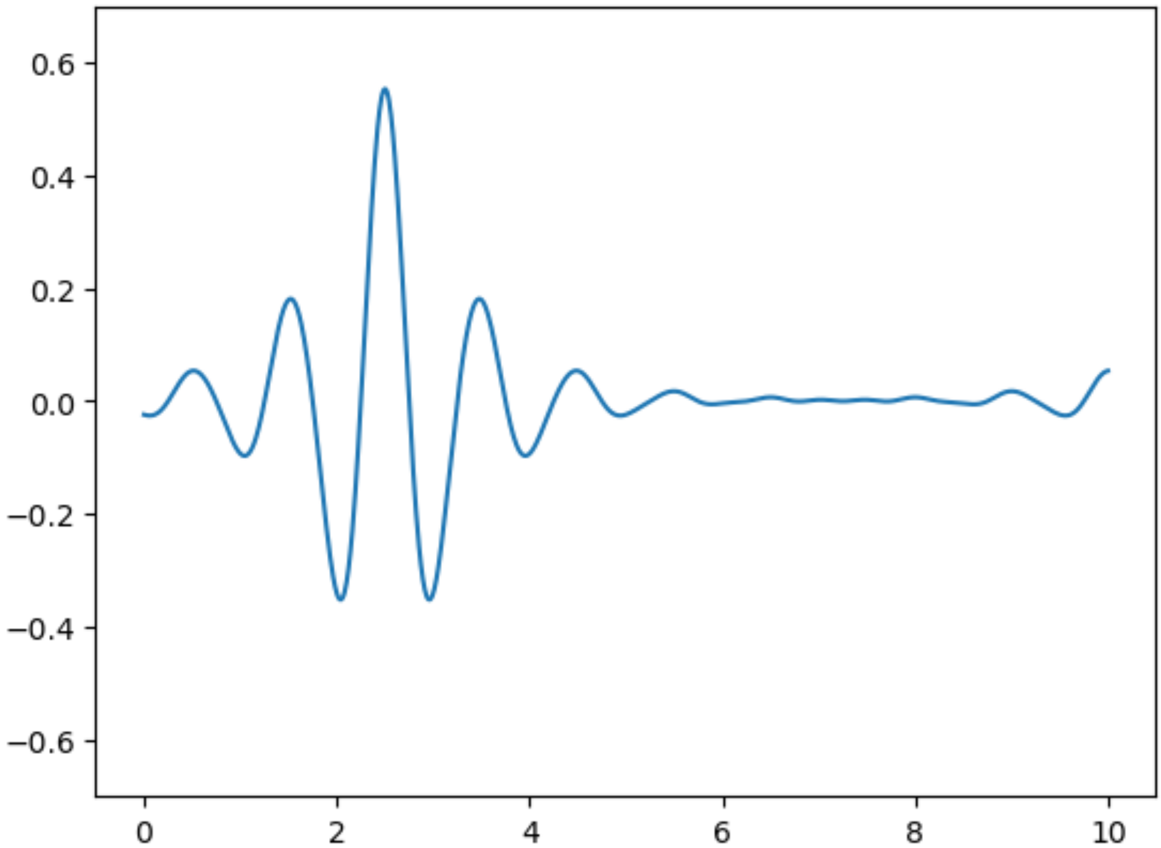


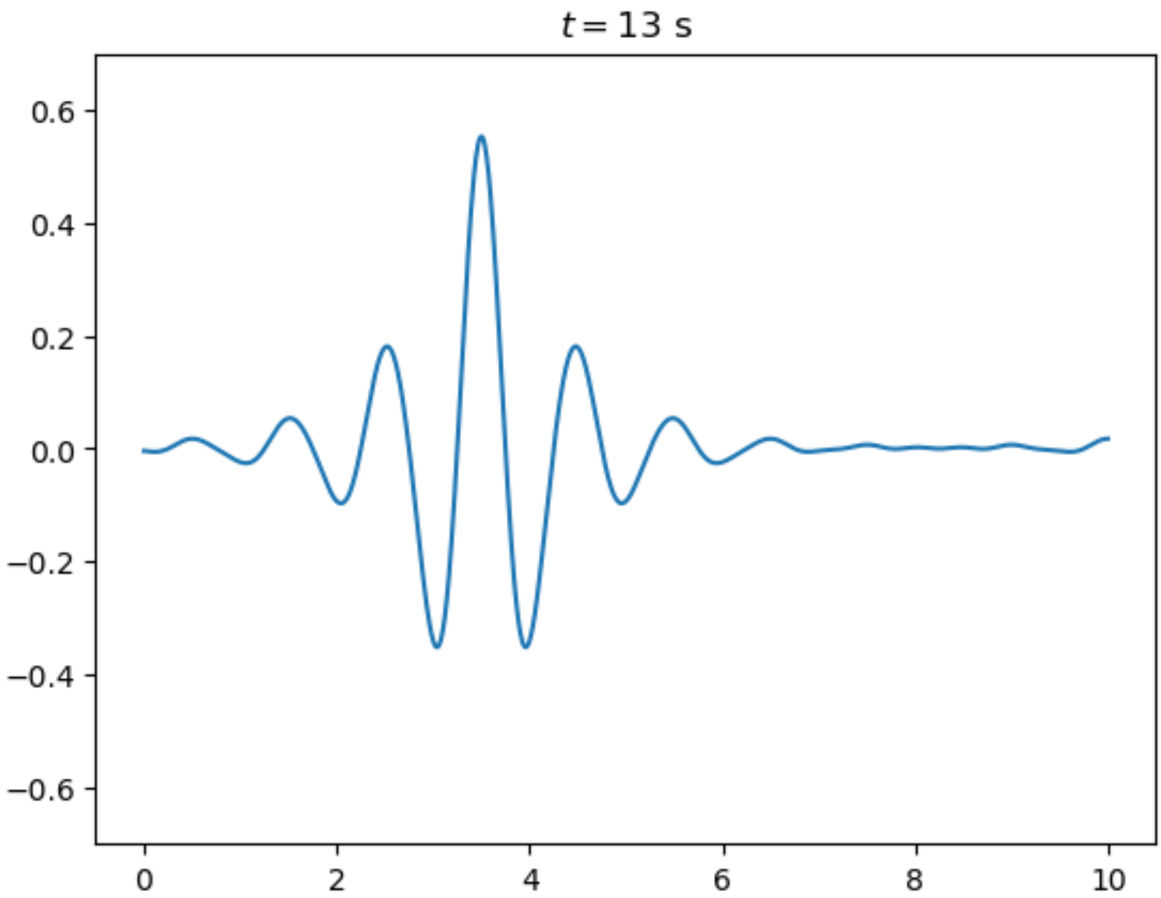
$t = 10$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

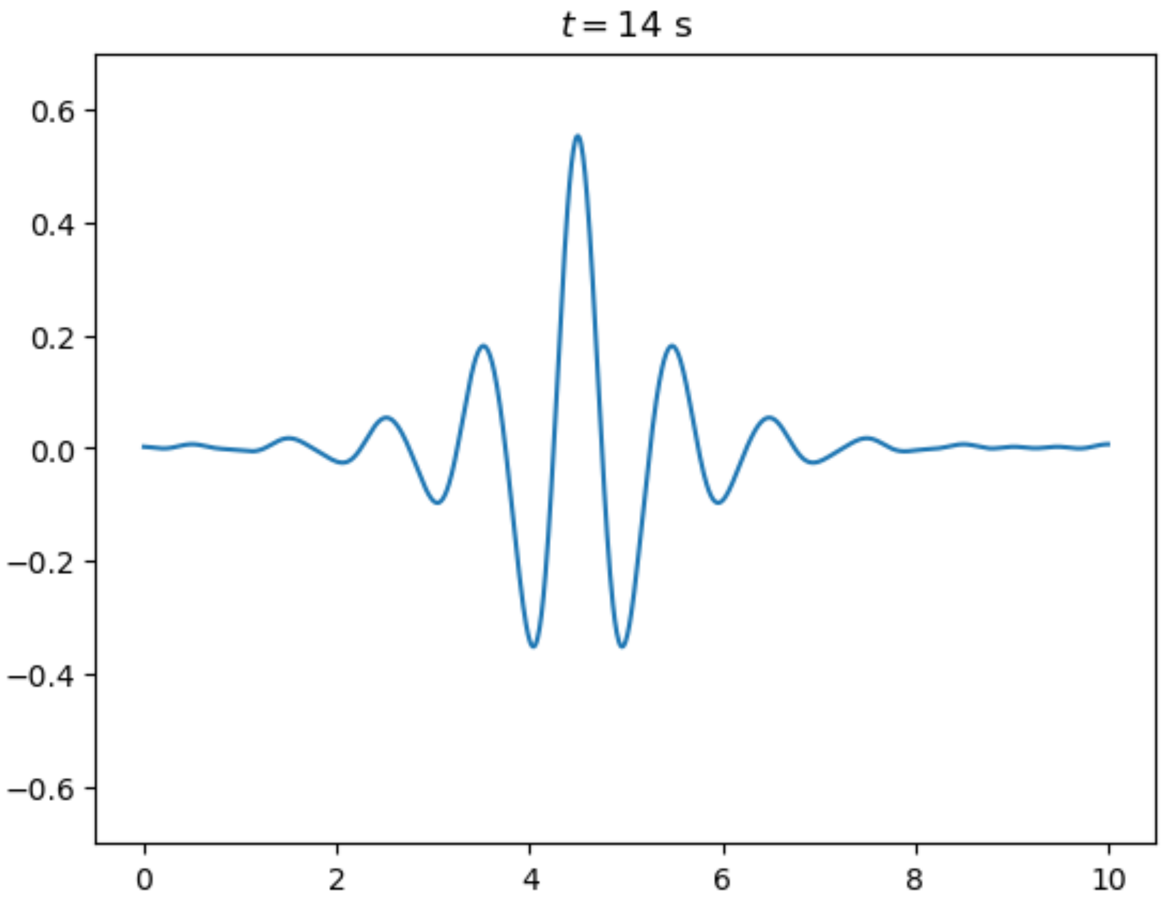
$t = 11$ s



$t = 12$ s

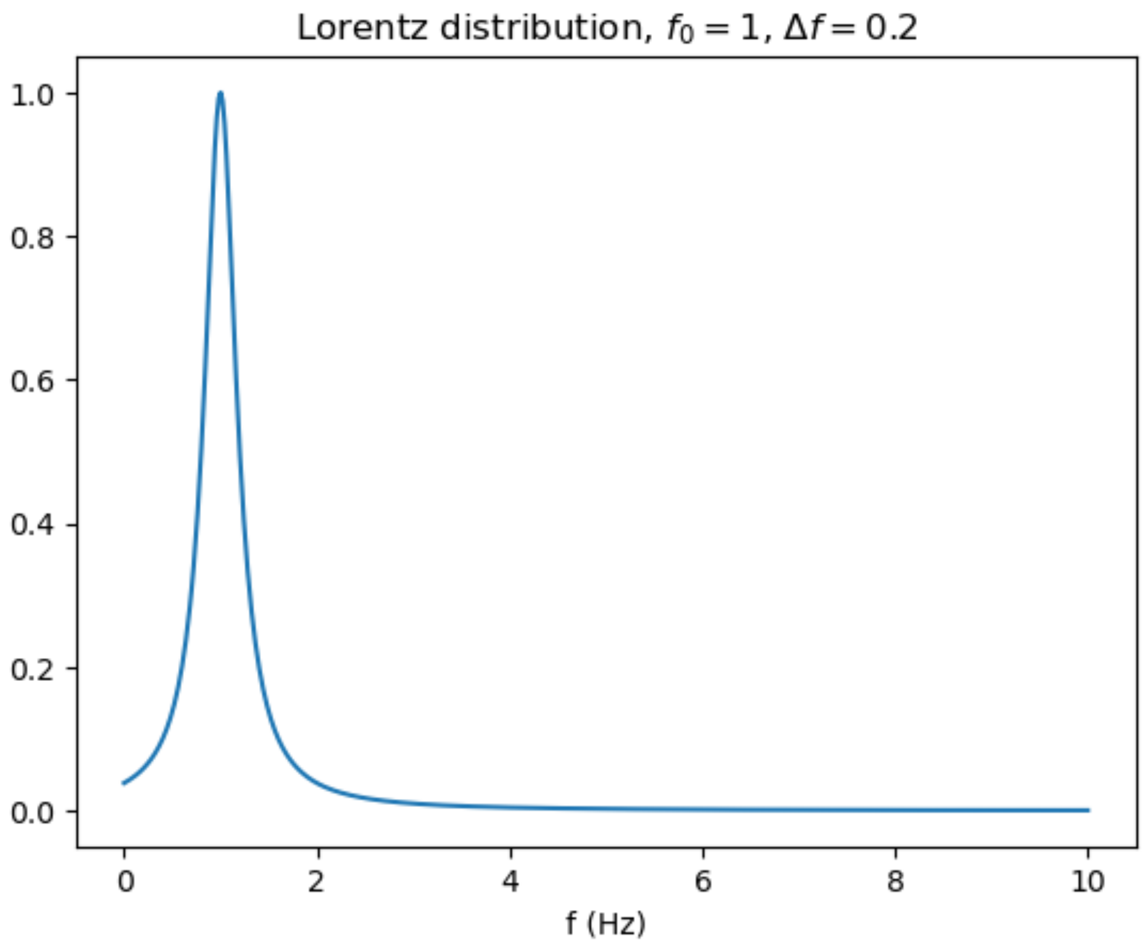


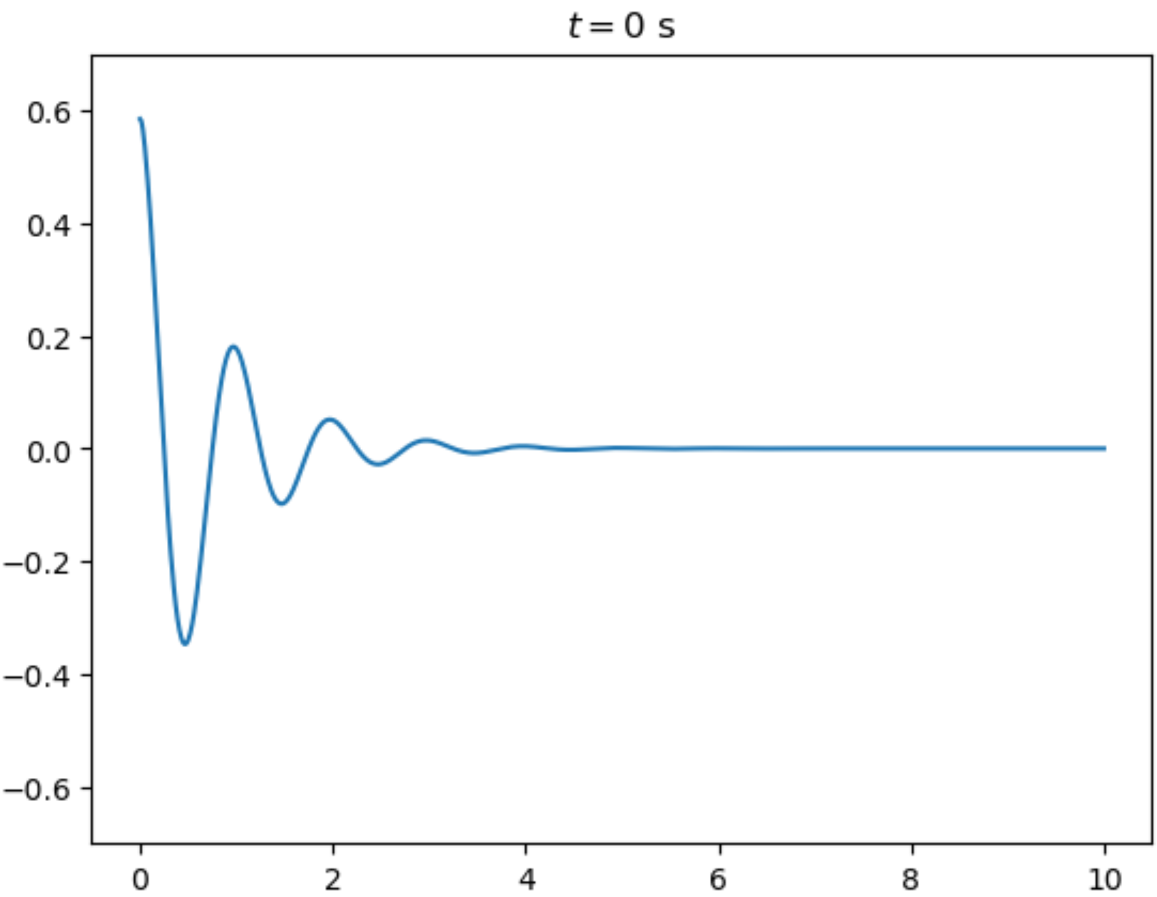

$t = 13$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

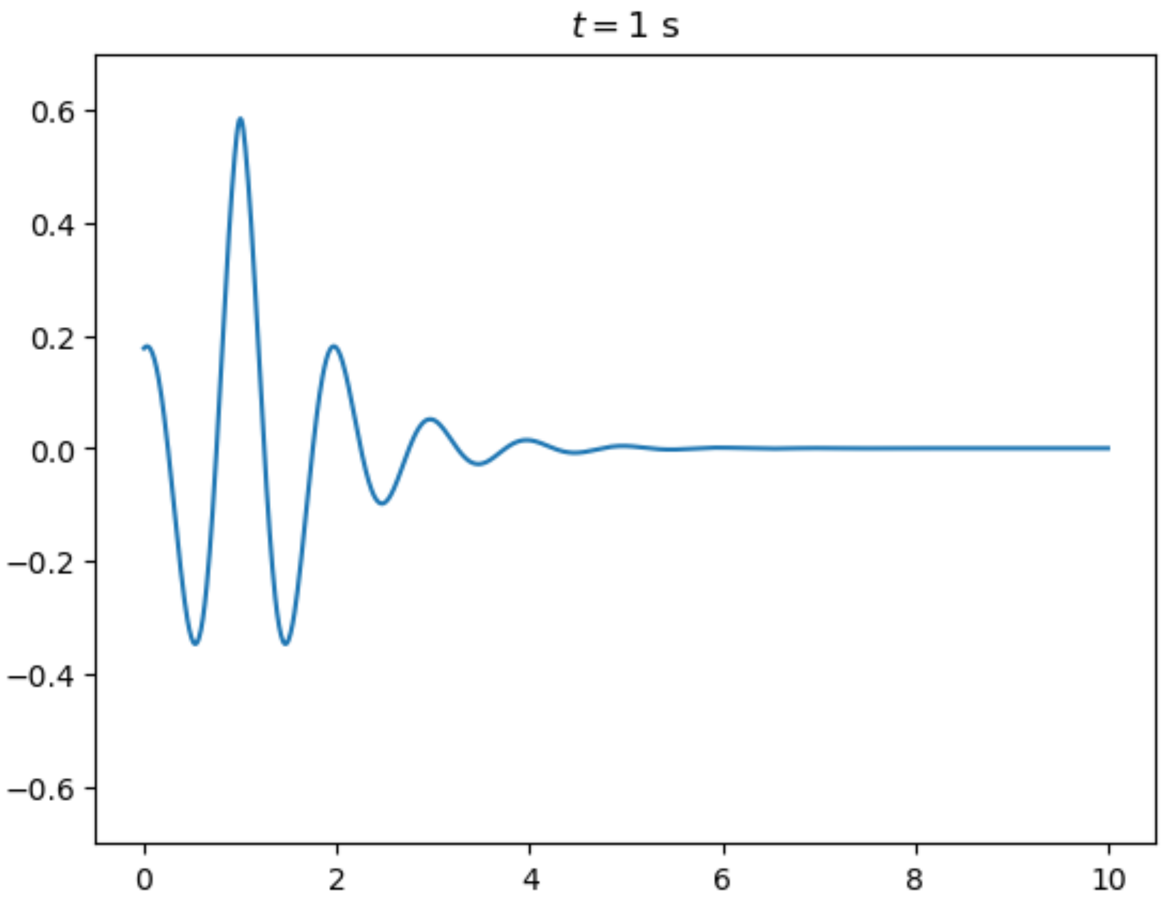

$t = 14$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

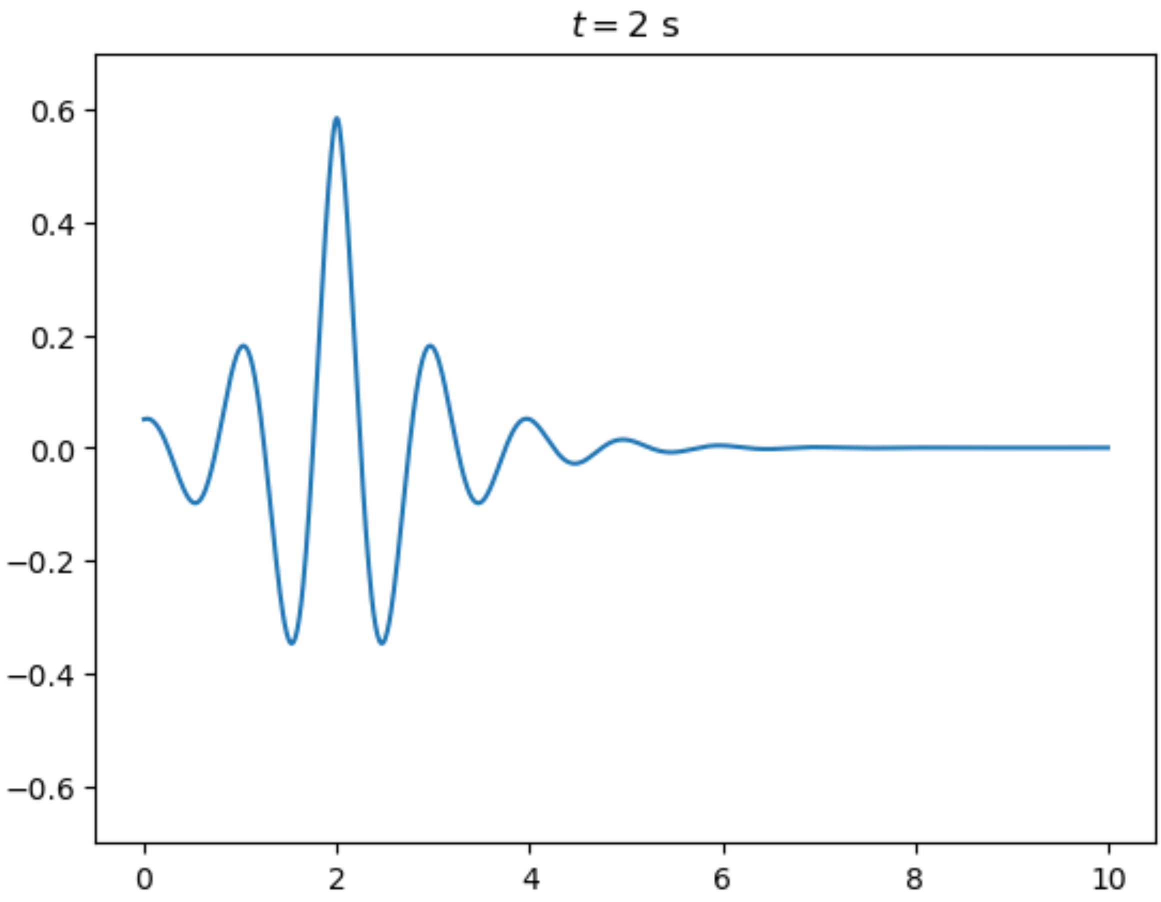
N = 500

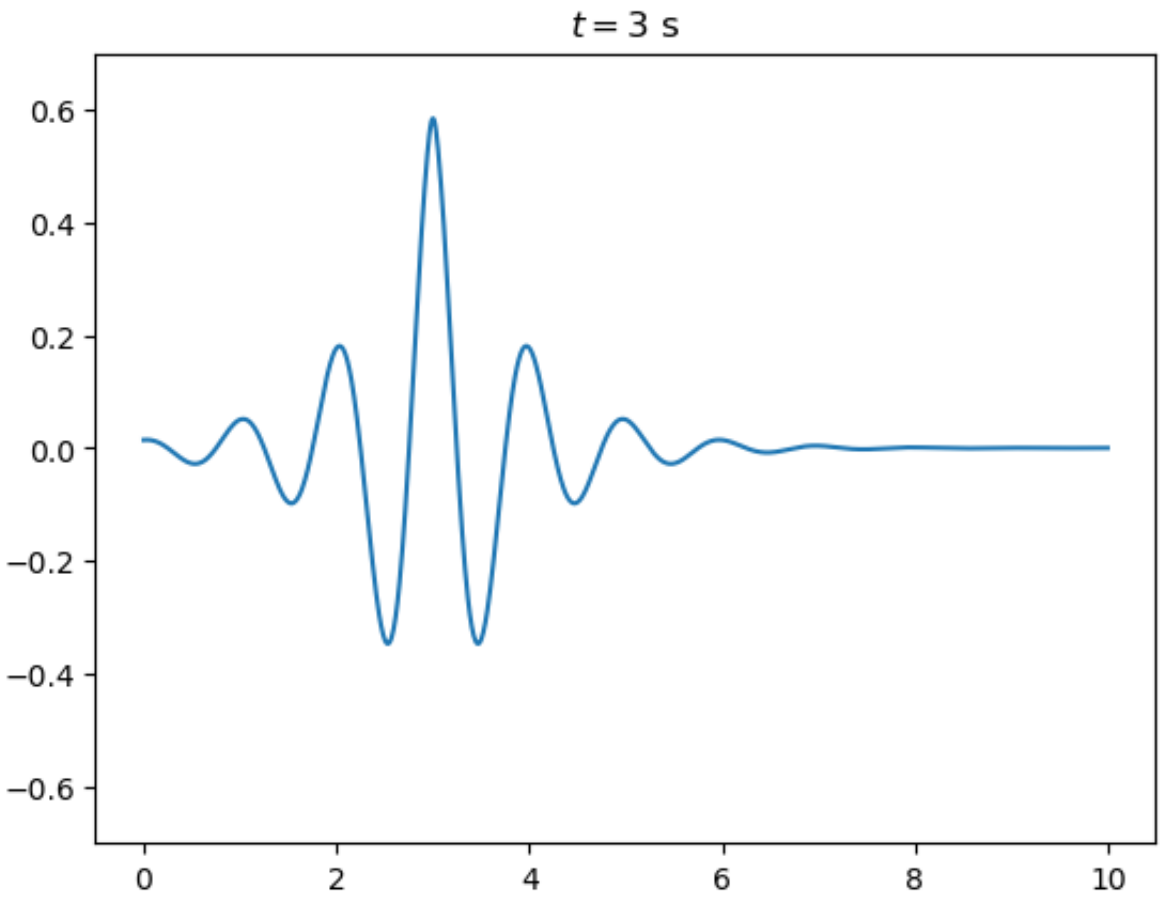
In [ ]:
```
N = 500

f_domain = np.linspace(0, 10*f_0, N)
lorentz_distrib = g(f_domain, f_0, f_width)

plt.plot(f_domain, lorentz_distrib)
plt.title(f"Lorentz distribution, $f_0={f_0}$, $\Delta f={f_width}$")
plt.xlabel("f (Hz)")
#plt.ylabel("")
plt.show()

df = f_domain[1]-f_domain[0]
c = 1 # phase speed
# fixed t

t_domain = np.arange(0, 12, 1)
for t in t_domain:
#t_1 = 0
    x_domain = np.linspace(0, 10, 1000)

    phi_1 = np.zeros(len(x_domain))
    for i in range(len(x_domain)):
        x = x_domain[i]
        for j in range(len(f_domain)):
            f = f_domain[j]
            phi_1[i] += lorentz_distrib[j]*np.cos(2*np.pi*f*(t - x/c))*df


    plt.plot(x_domain, phi_1)
    plt.title(f"$t={t}$ s")
    plt.ylim((-0.7, 0.7))
    plt.show()
```


Lorentz distribution, $f_0 = 1$, $\Delta f = 0.2$
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
f (Hz)

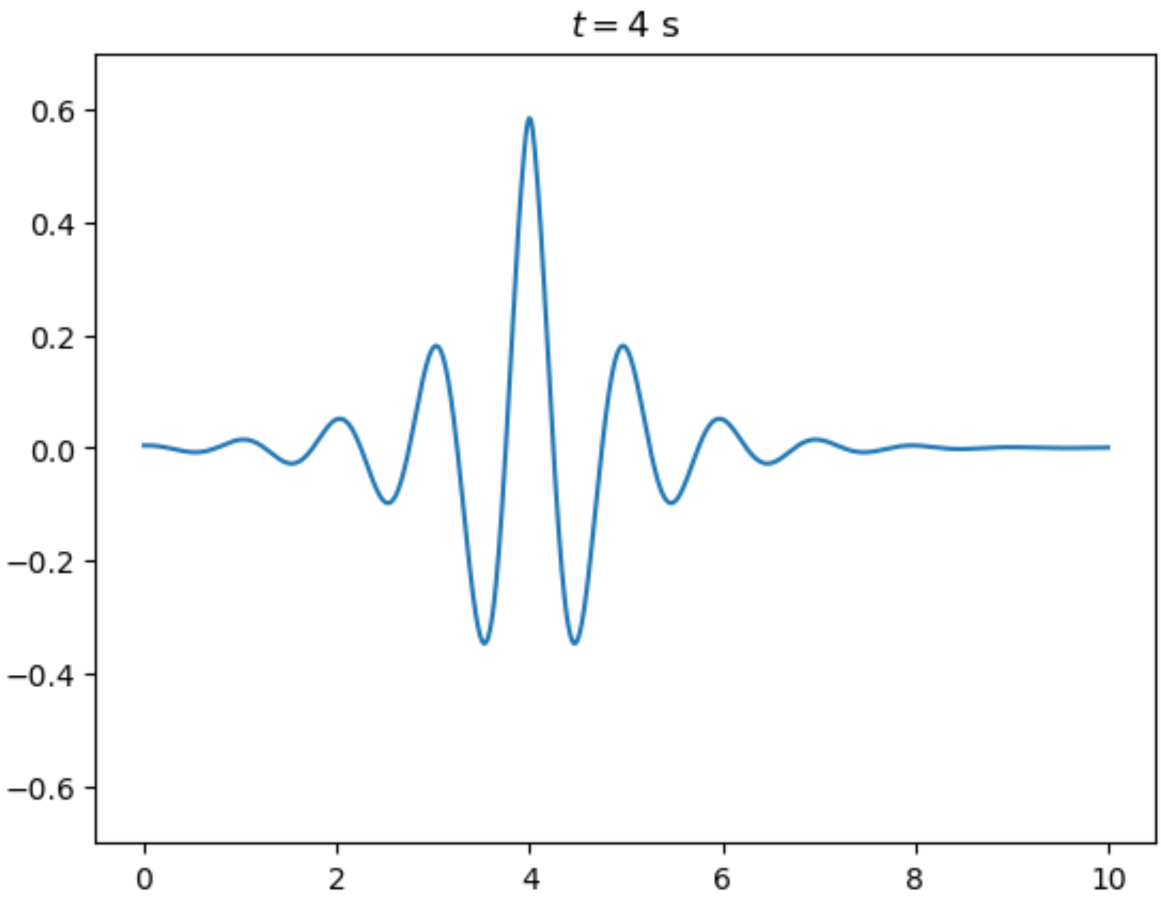

$t = 0$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 1$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

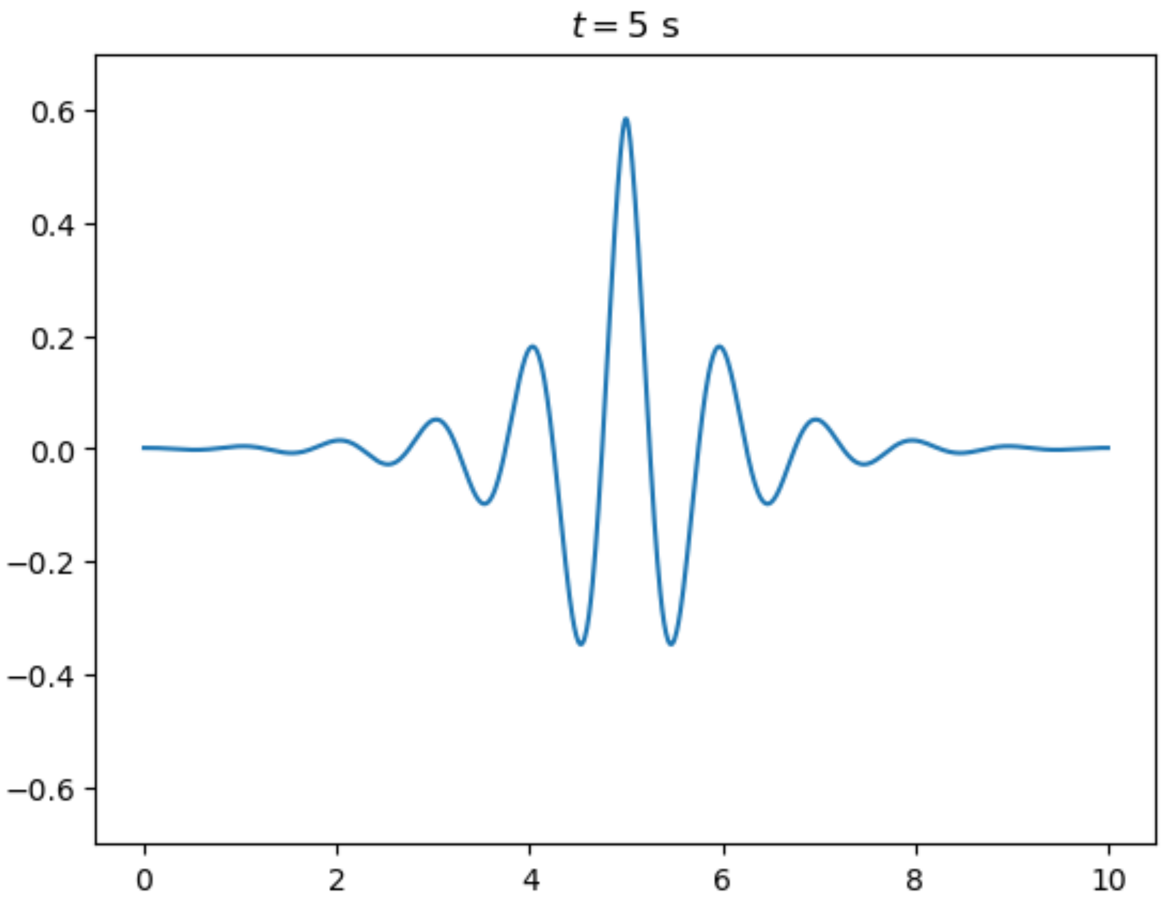

$t = 2$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

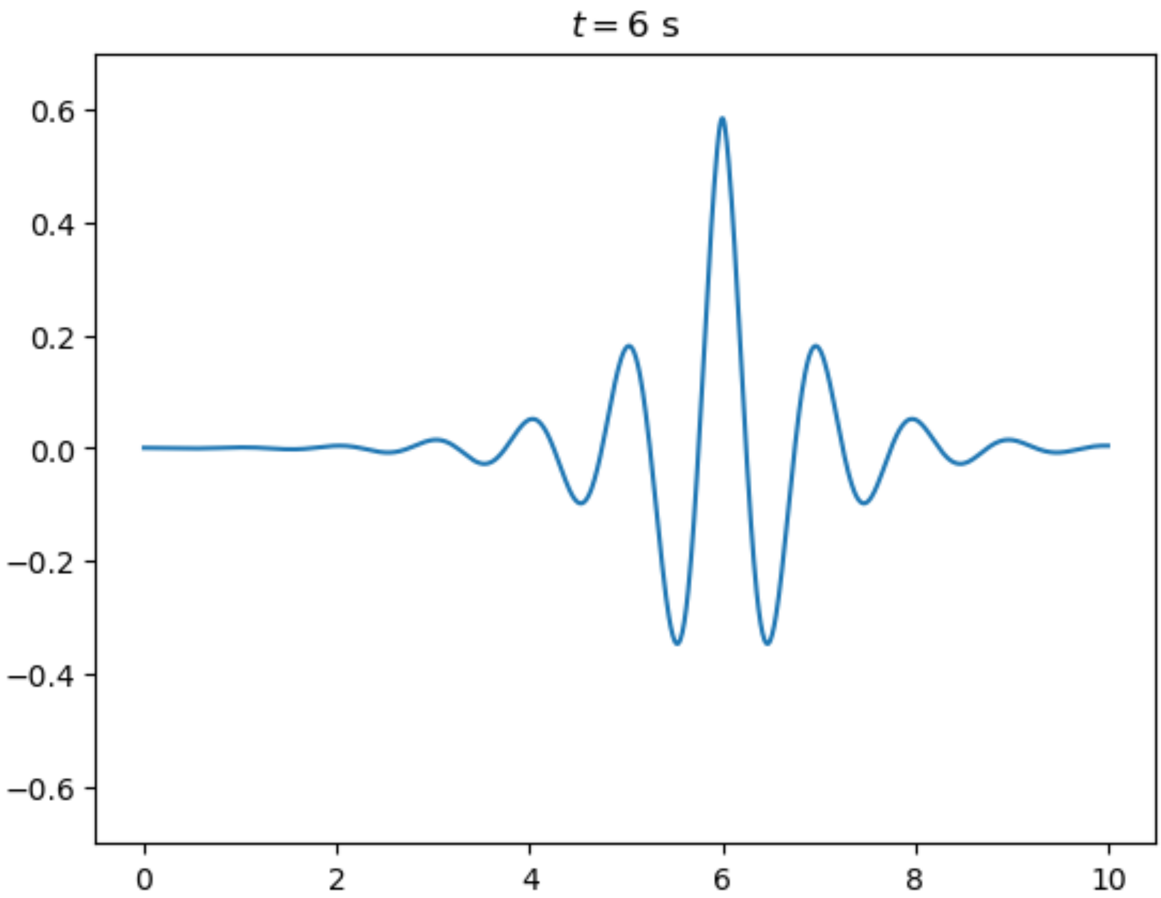


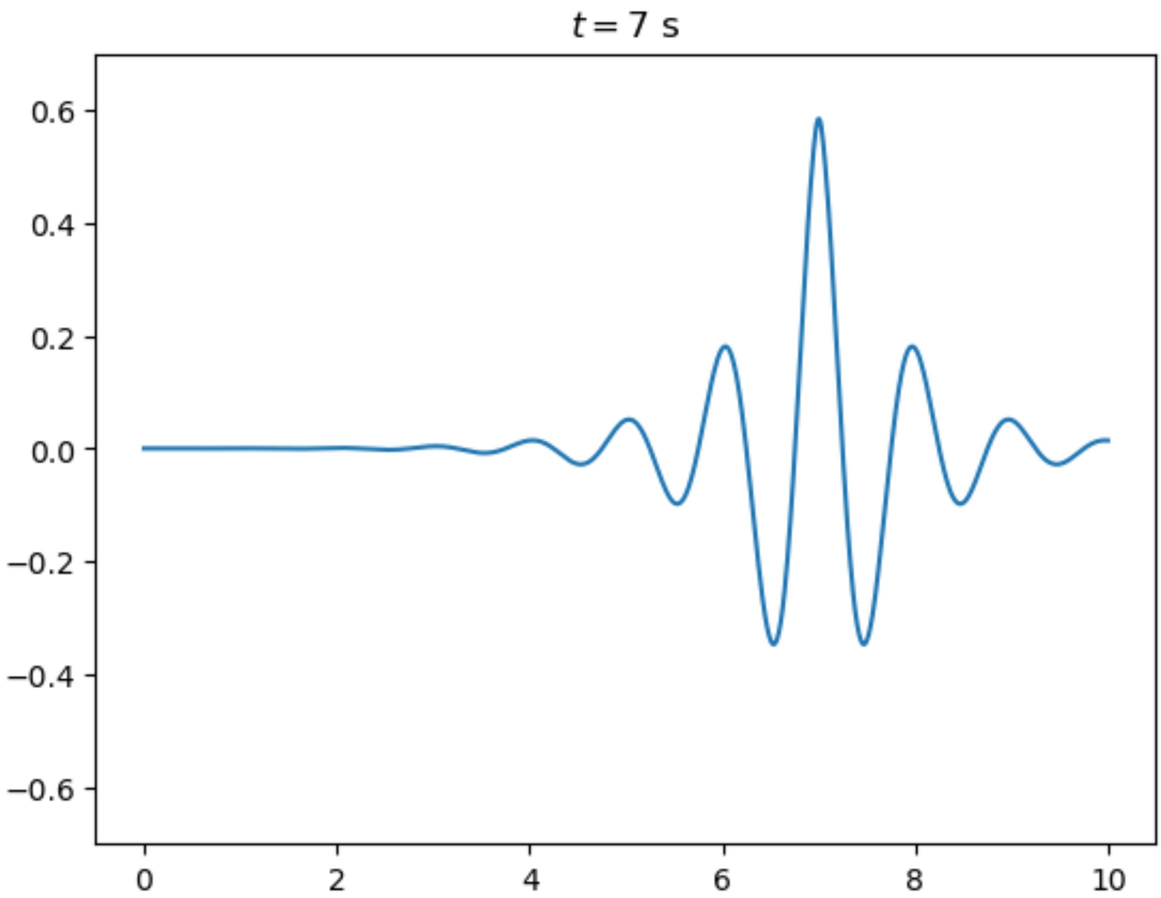
$t = 3$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

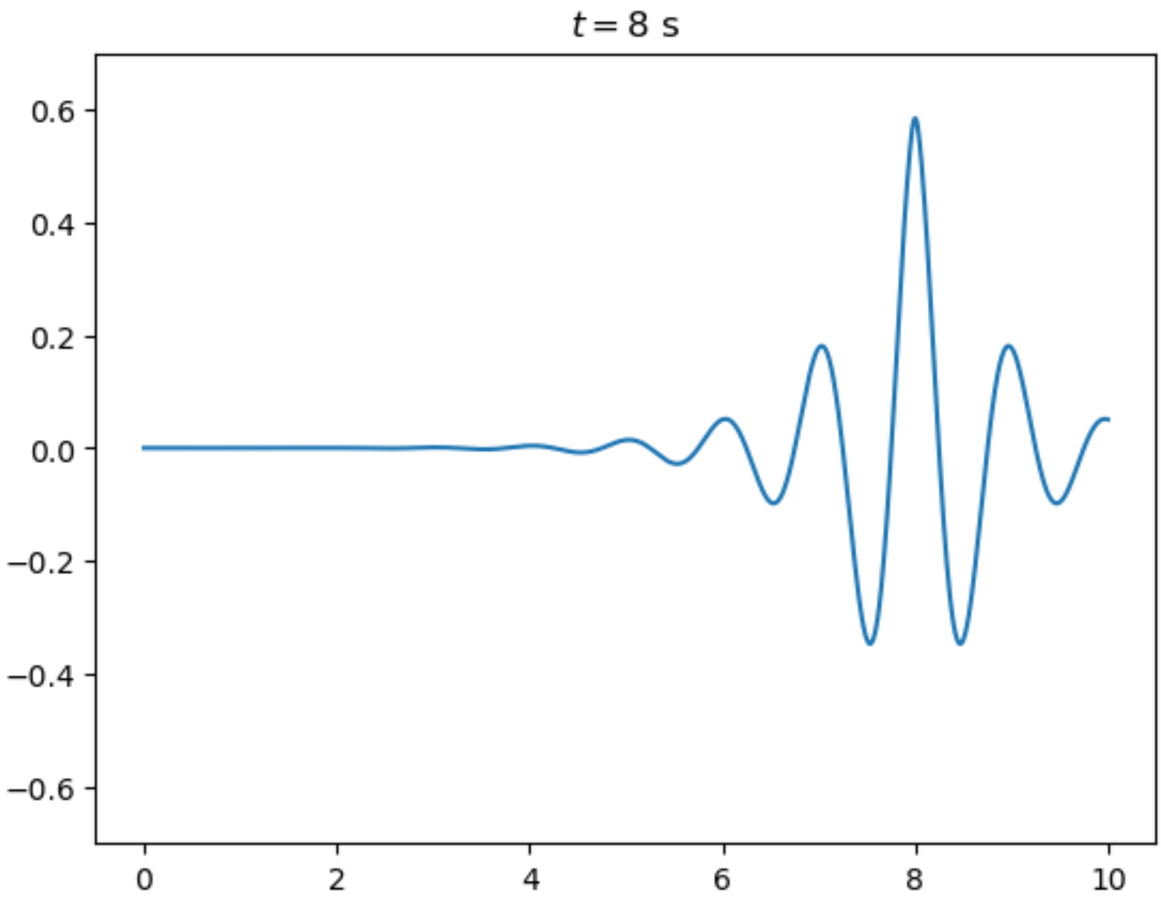


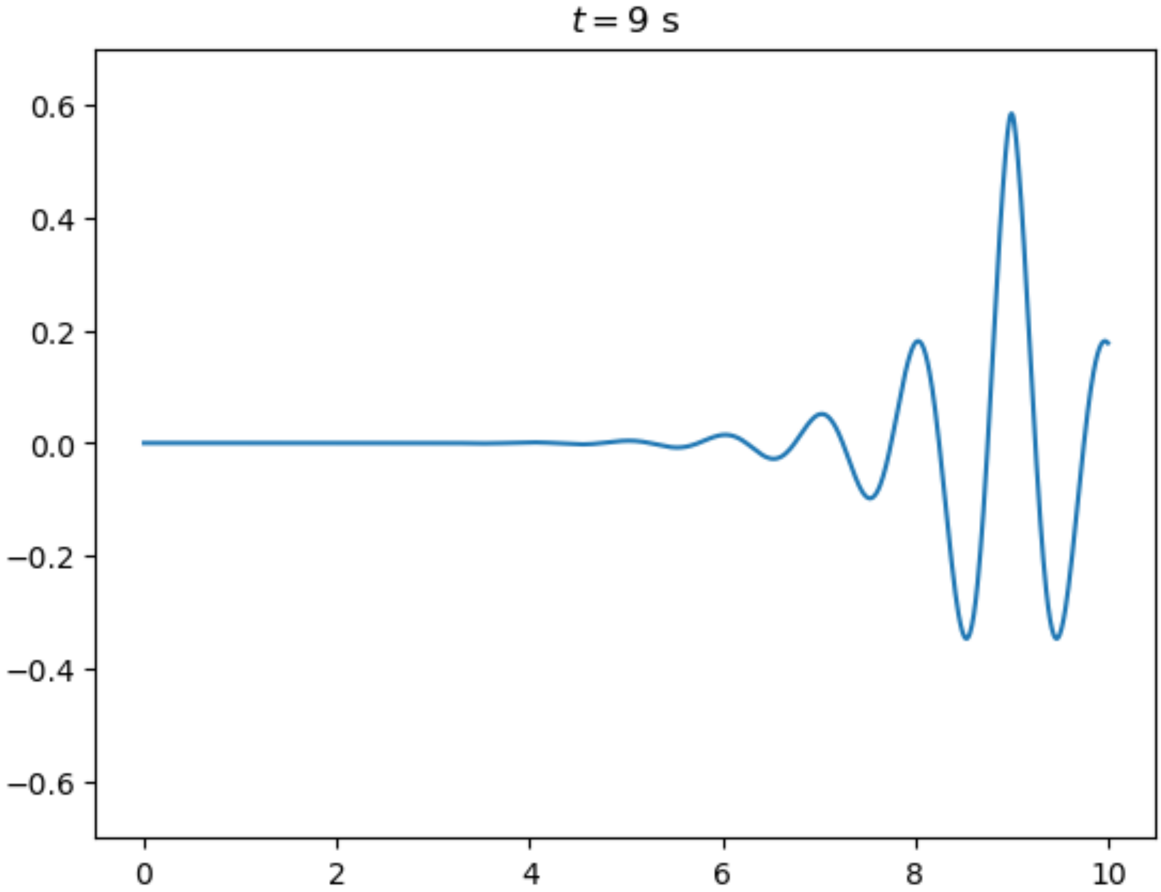
$t = 4$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

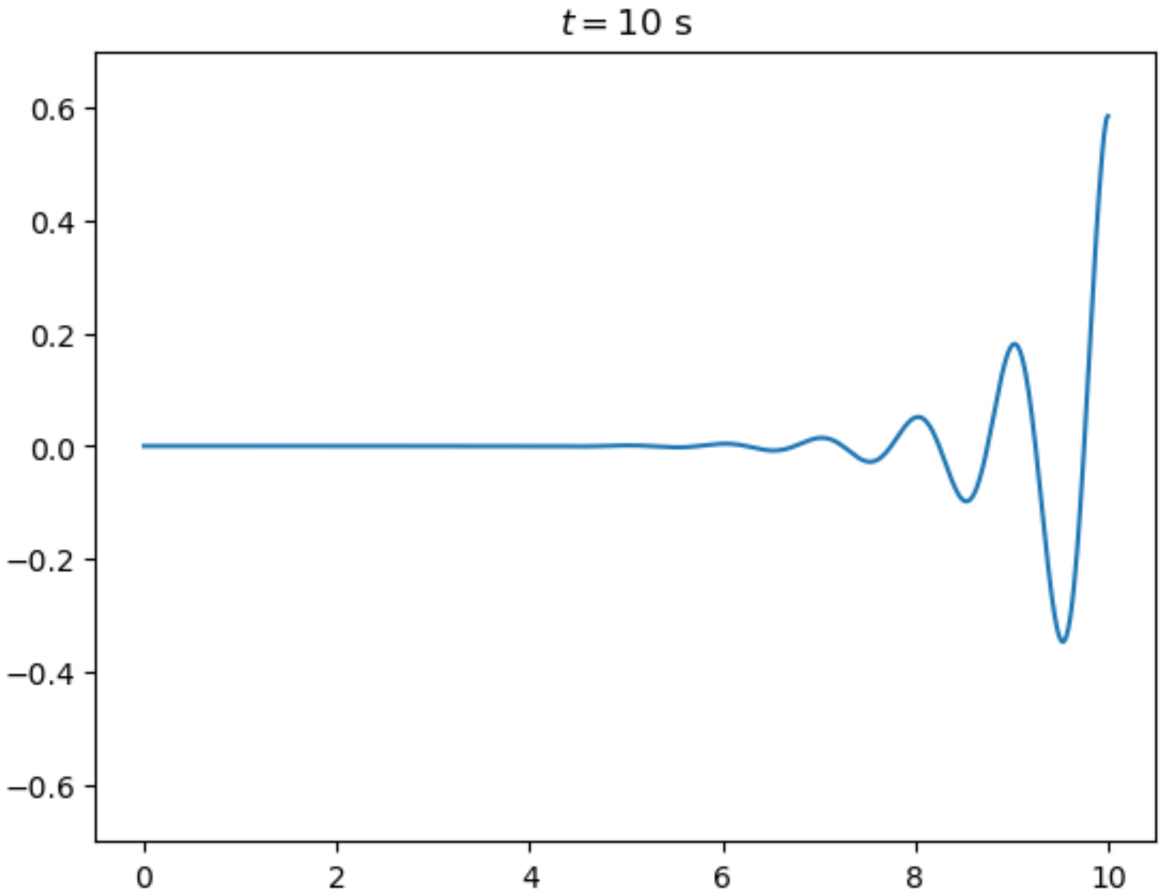

$t = 5$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

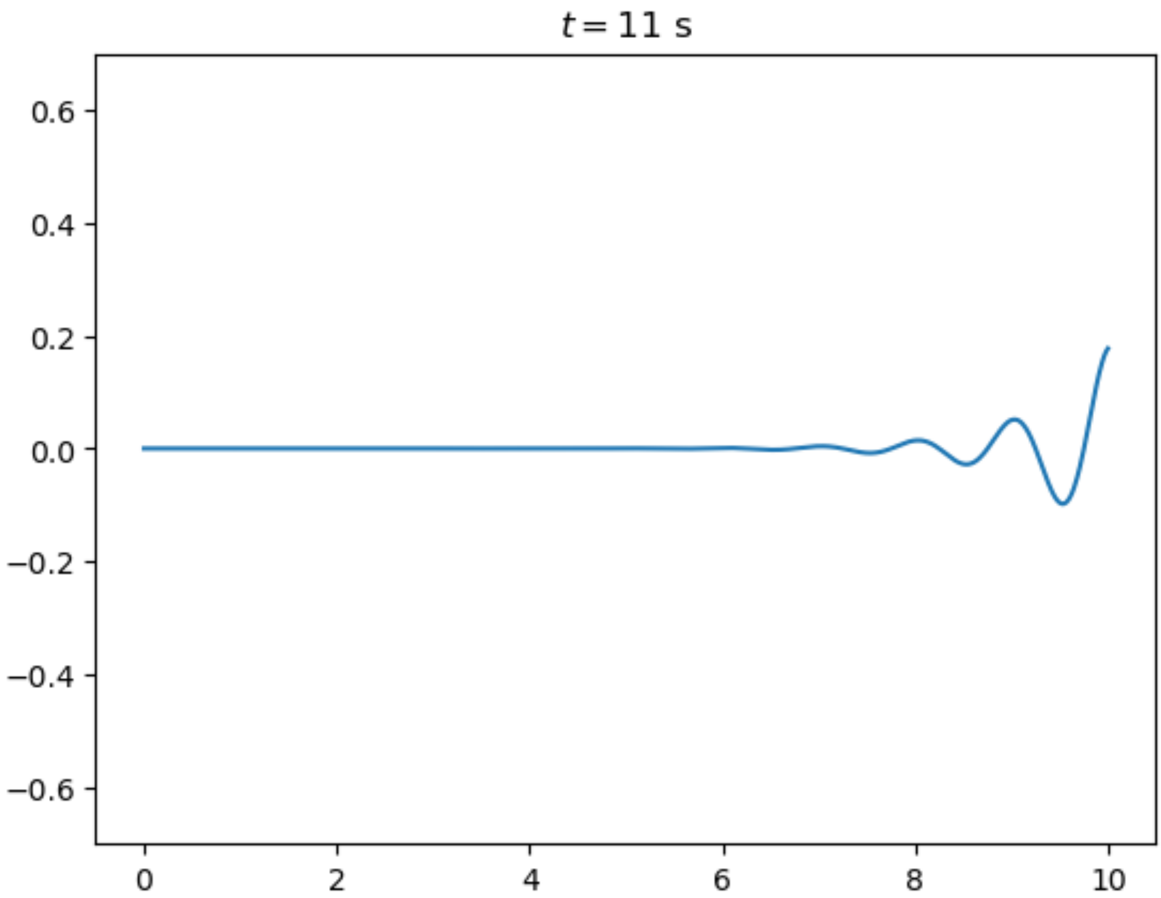

$t = 6$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 7$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 8$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

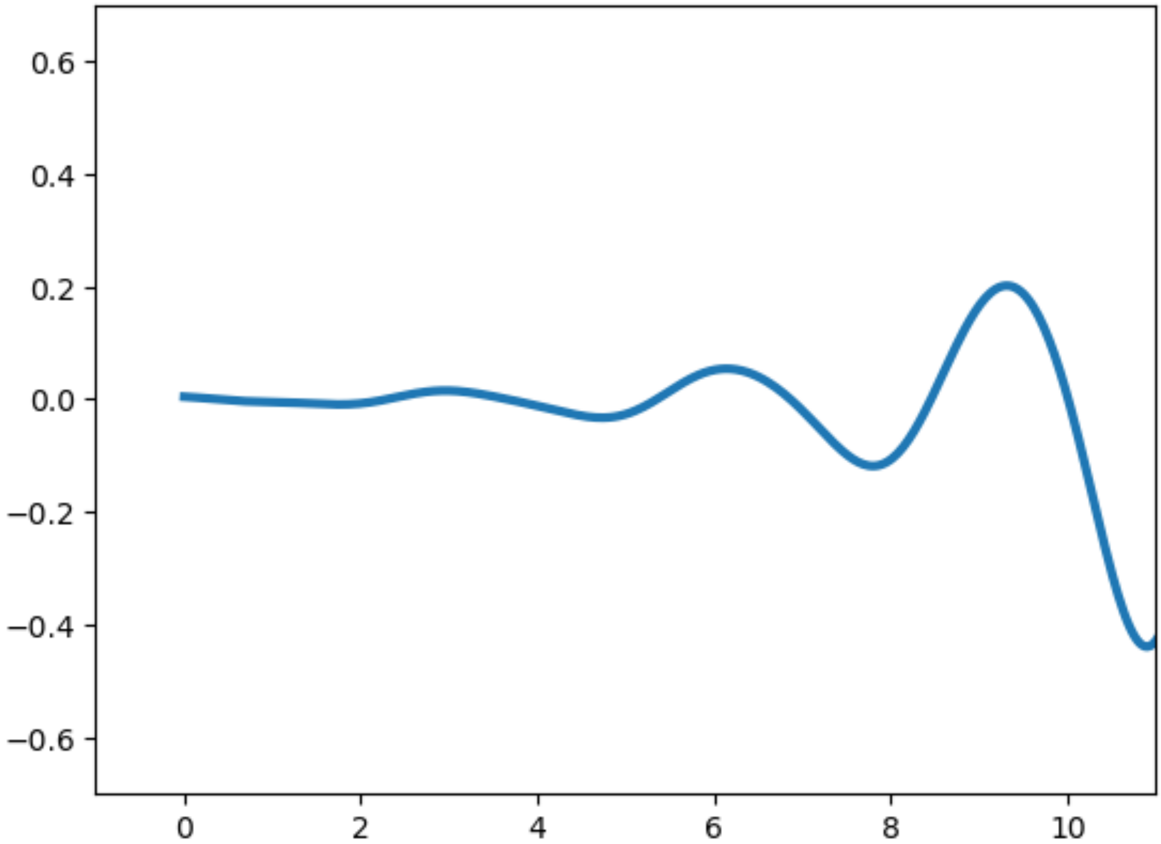

$t = 9$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 10$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 11$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

In [ ]:
```
N = 50

f_domain = np.linspace(0, 2*f_0, N)
lorentz_distrib = g(f_domain, f_0, f_width)

'''
fig = plt.figure()
ax = plt.axes(xlim=(-1, 11), ylim=(-20, 30))
line, = ax.plot([], [], lw=3)


def init():
    line.set_data([], [])
    return line,


def animate(i):
    x = np.linspace(0, 4, 1000)
    y = np.sin(2 * np.pi * (x - 0.01 * i))
    line.set_data(x, y)
    return line,

anim = FuncAnimation(fig, animate, init_func=init,
                               frames=200, interval=20, blit=True)
'''

df = f_domain[1]-f_domain[0]
c = 1 # phase speed
# fixed t
#alpha = 0.1

FPS = 20
rate = 1/FPS
t_domain = np.arange(0, 4, rate)

PHI_1 = [] # time series
for t in t_domain:
#t_1 = 0
    x_domain = np.linspace(0, 10, 1000)

    phi_x = np.zeros(len(x_domain))
    for i in range(len(x_domain)):
        x = x_domain[i]
        for j in range(len(f_domain)):
            k = 2*np.pi*f/c
            f = f_domain[j]
            phi_x[i] += lorentz_distrib[j]*np.cos(2*np.pi*f*t - k*x)*df
    PHI_1.append(phi_x)

    #plt.plot(x_domain, phi_1)
    #plt.title(f"$t={t}$ s")
    #plt.ylim((-20, 30))
    #plt.show()
```

In [ ]:
```
fig = plt.figure()
ax = plt.axes(xlim=(-1, 11), ylim=(-0.7, 0.7))
line, = ax.plot([], [], lw=3)


def init():
    line.set_data([], [])
    return line,


def animate(i):
    phi_x = PHI_1[i]
    line.set_data(x_domain, phi_x)
    return line,

anim = FuncAnimation(fig, animate, init_func=init,
                               frames=len(PHI_1), interval=FPS/2, blit=Tr


anim.save(f'figs/wave_packet_ndiff_{N=}_{c=}.gif', writer='imagemagick')
```



It does seem to tend towards an ideal wave packet

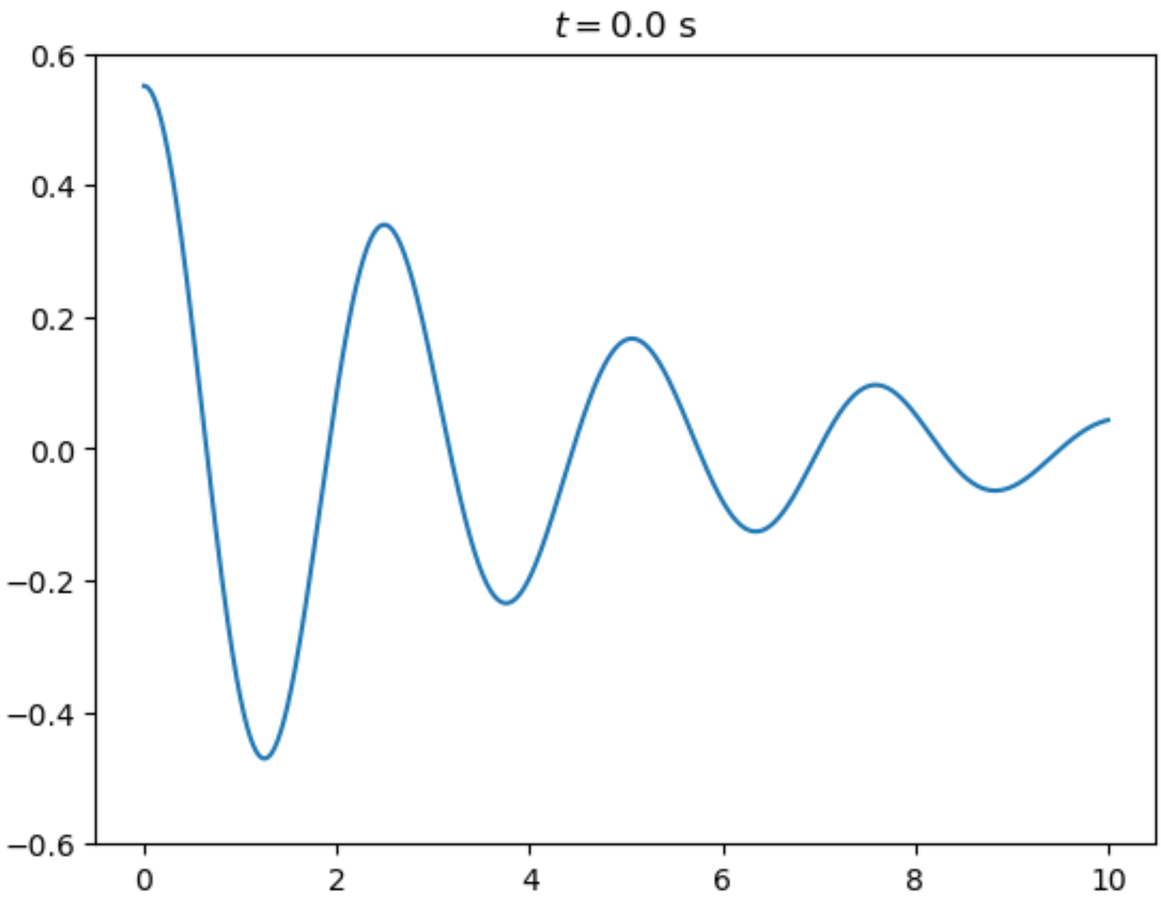
Dispersif medium

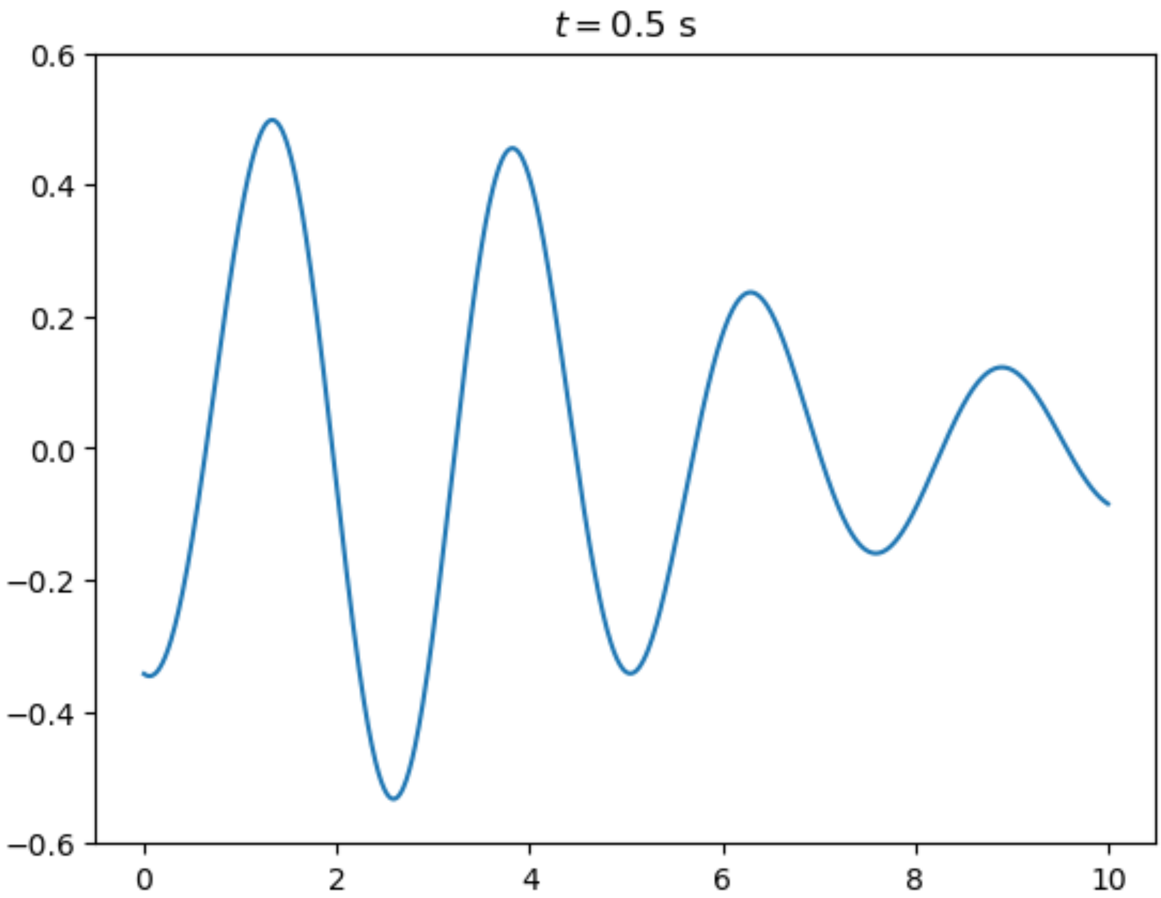
In the case of light:

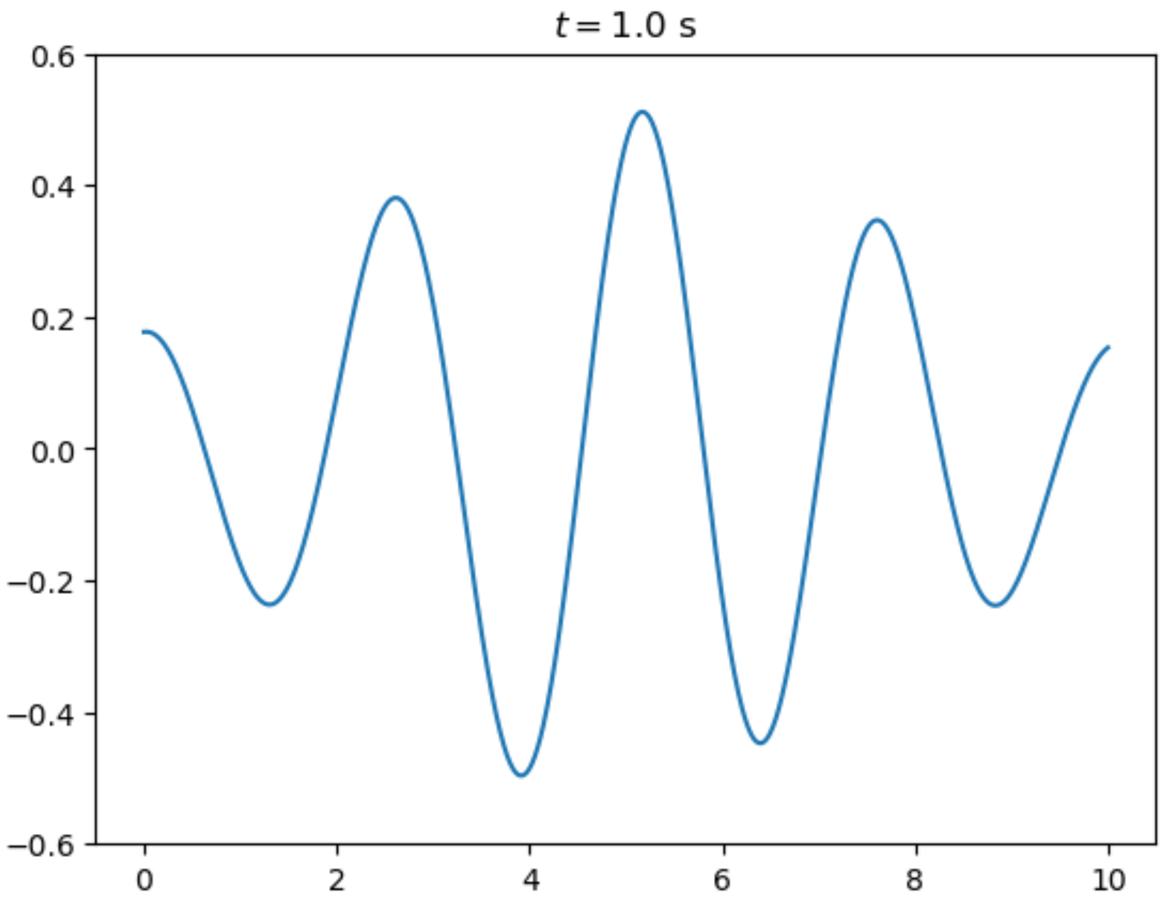
$$n(\lambda) = A + \frac{B}{\lambda^2} + o(1/\lambda^2)$$

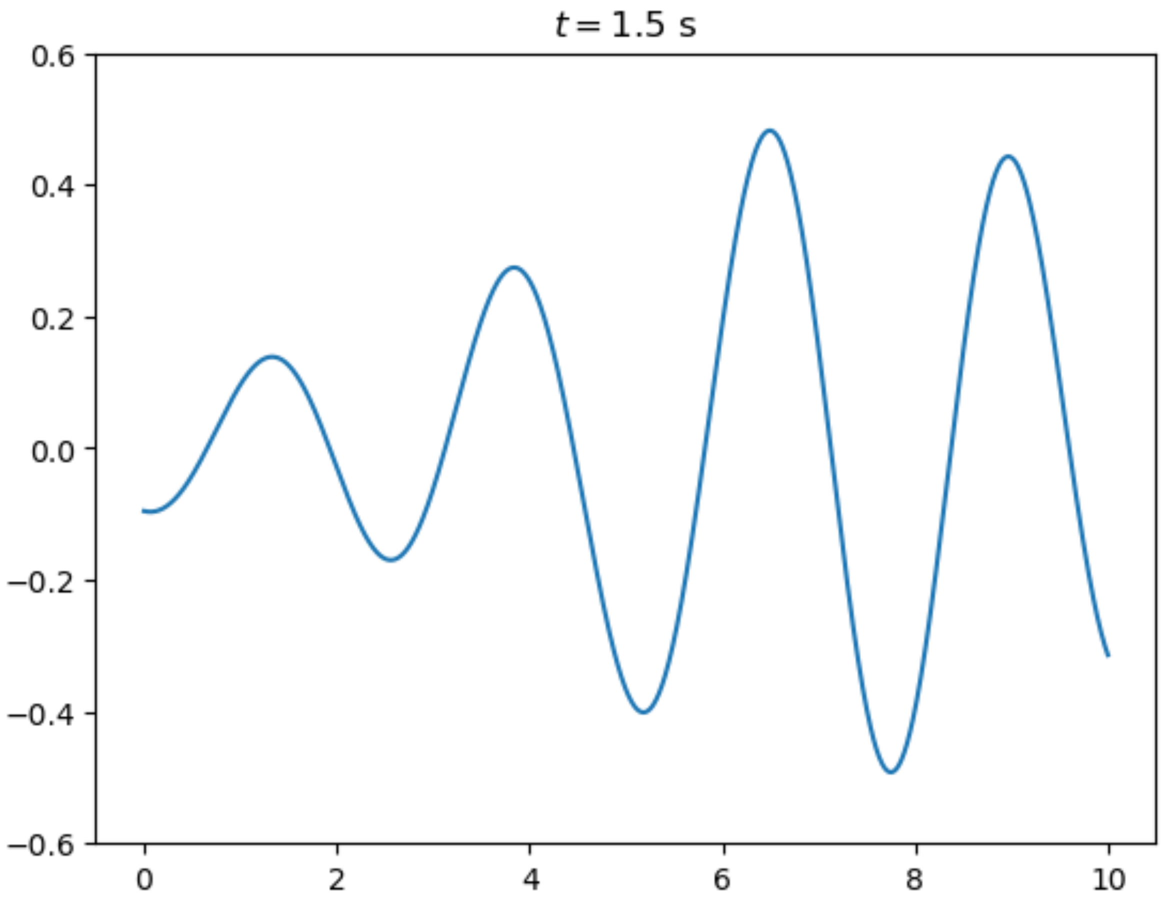
where n is the refractive index $n = \frac{c}{v_\phi}$, so there is a visible dependency between wavelength and phase velocity

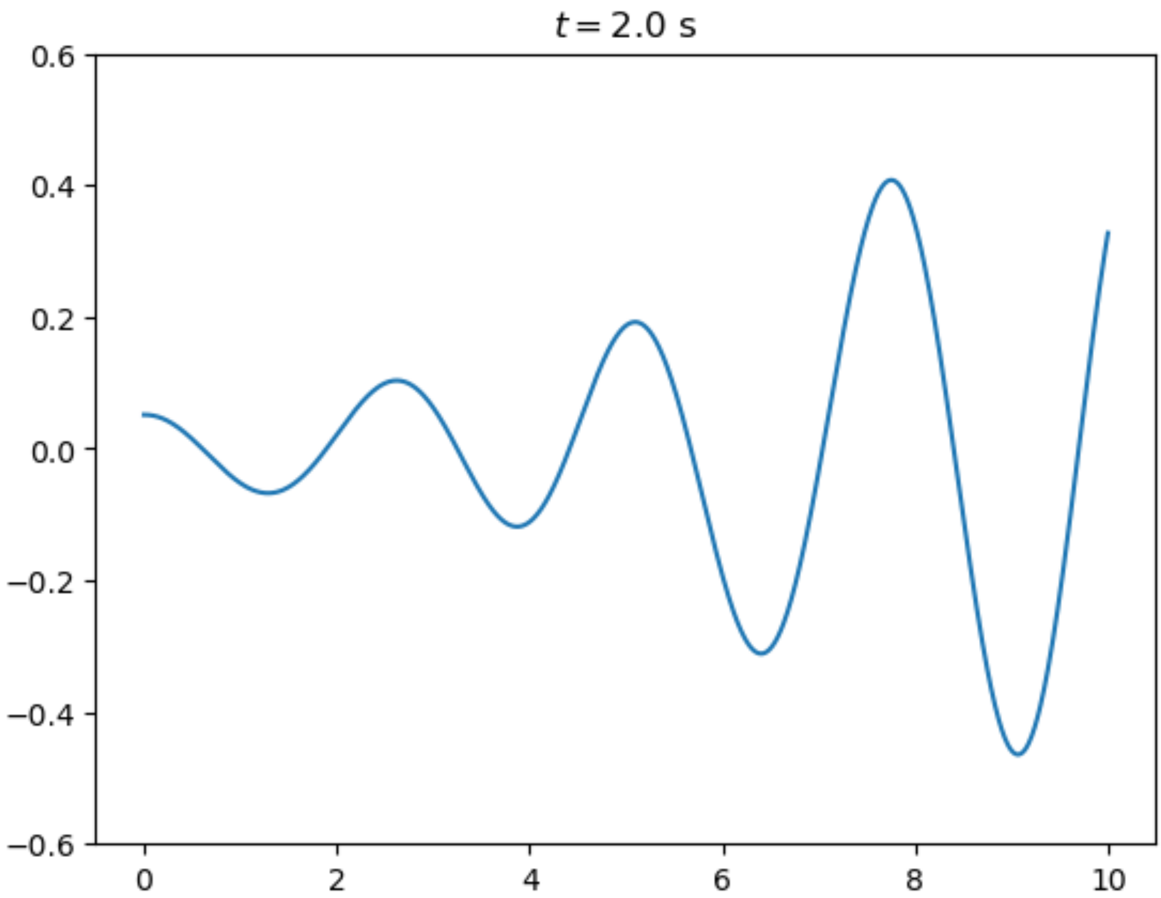
In [ ]:
```
N = 50

f_domain = np.linspace(0, 2*f_0, N)
lorentz_distrib = g(f_domain, f_0, f_width)

#plt.plot(f_domain, lorentz_distrib)
#plt.title(f"Lorentz distribution, $f_0={f_0}$, $\Delta f={f_width}$")
#plt.xlabel("f (Hz)")
#plt.ylabel("")
#plt.show()


df = f_domain[1]-f_domain[0]
#c = 1 # phase speed
# fixed t

t_domain = np.arange(0, 5, 0.5)
for t in t_domain:
#t_1 = 0
    x_domain = np.linspace(0, 10, 1000)

    phi_1 = np.zeros(len(x_domain))
    for i in range(len(x_domain)):
        x = x_domain[i]
        for j in range(len(f_domain)):
            k = np.sqrt(2*np.pi*f)
            f = f_domain[j]
            phi_1[i] += lorentz_distrib[j]*np.cos(2*np.pi*f*t - k*x)*df


    plt.plot(x_domain, phi_1)
    plt.title(f"$t={t}$ s")
    plt.ylim((-0.6, 0.6))
    plt.show()
```

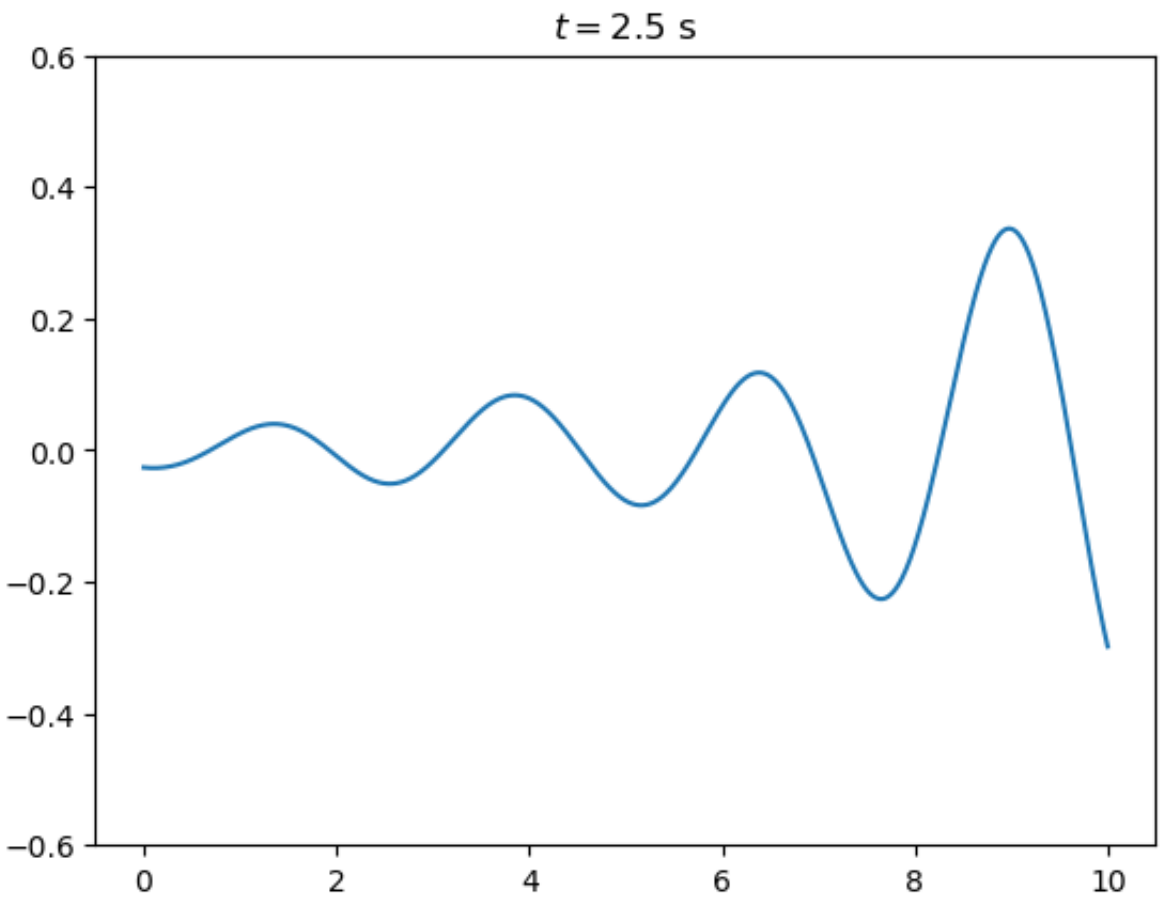

$t = 0.0$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

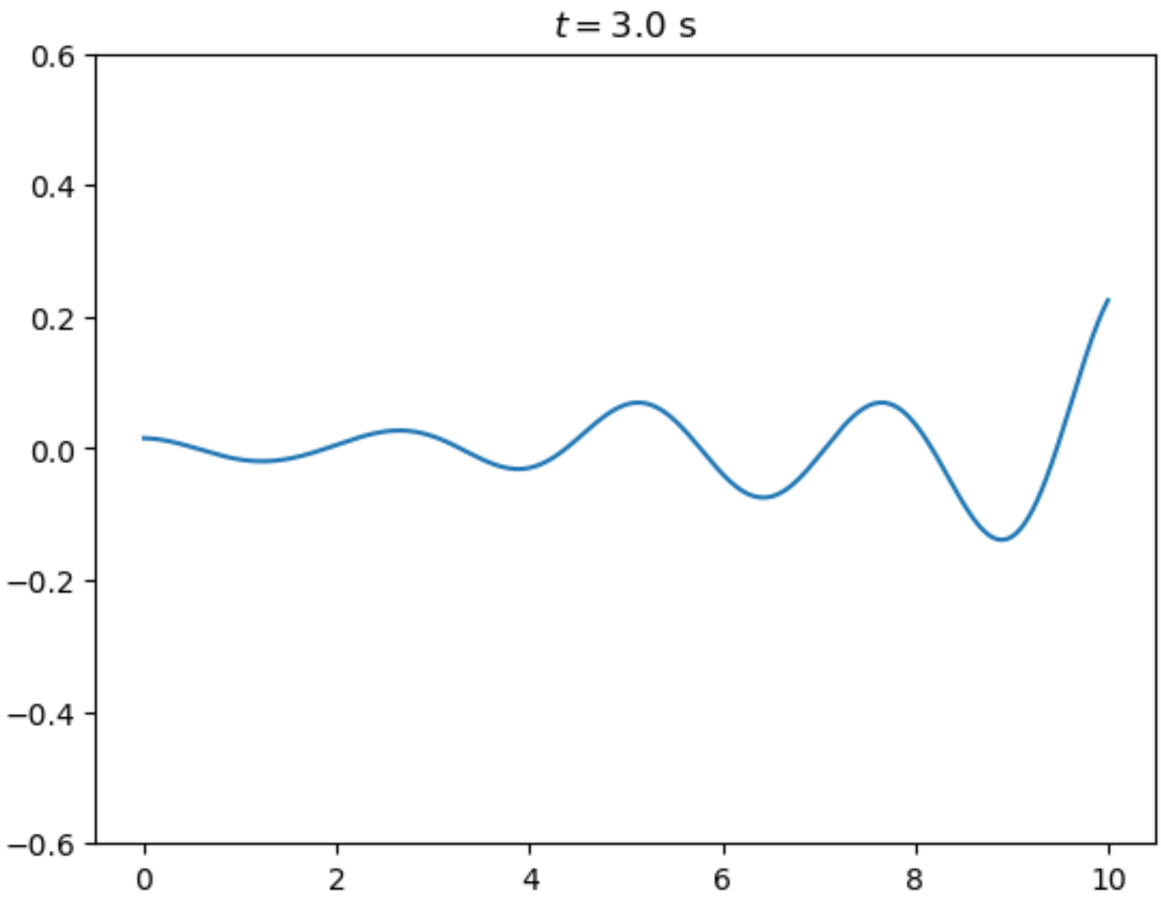

$t = 0.5$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

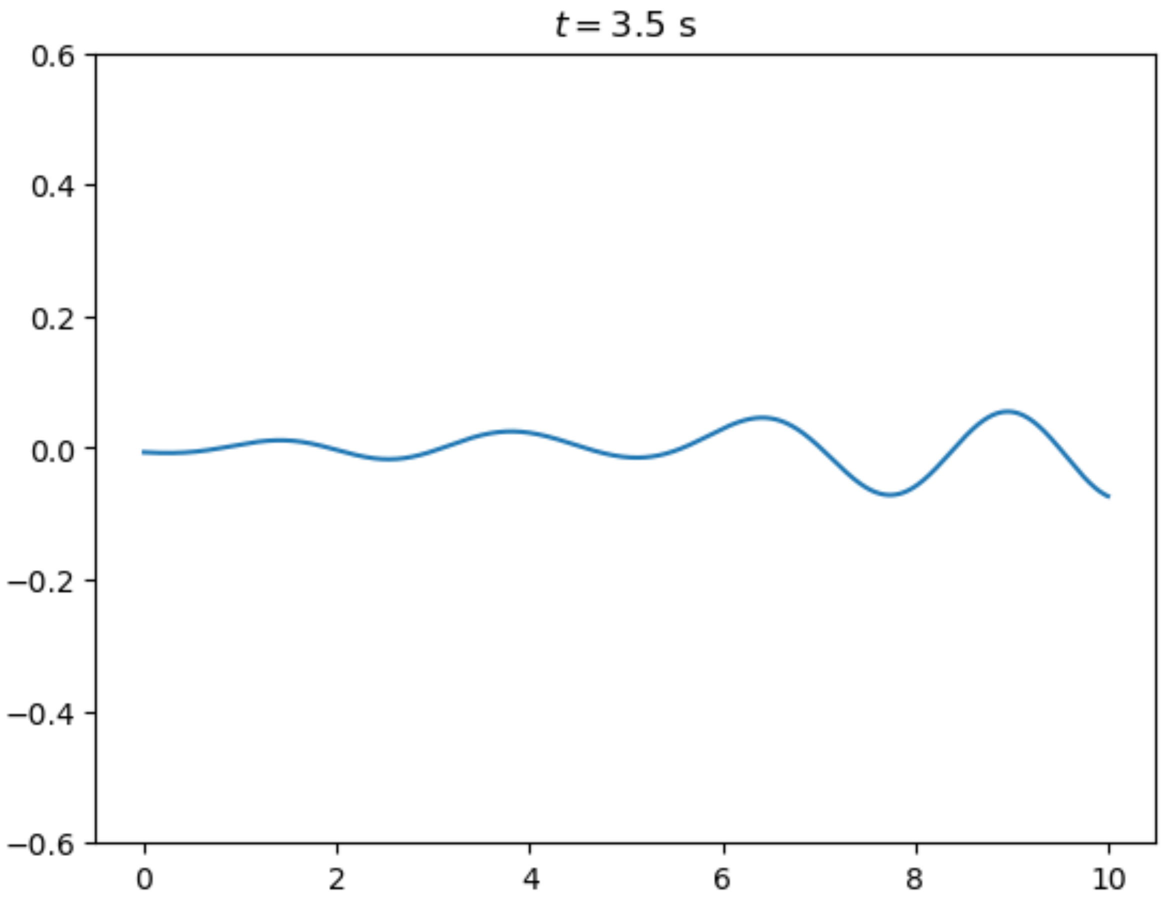

$t = 1.0$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

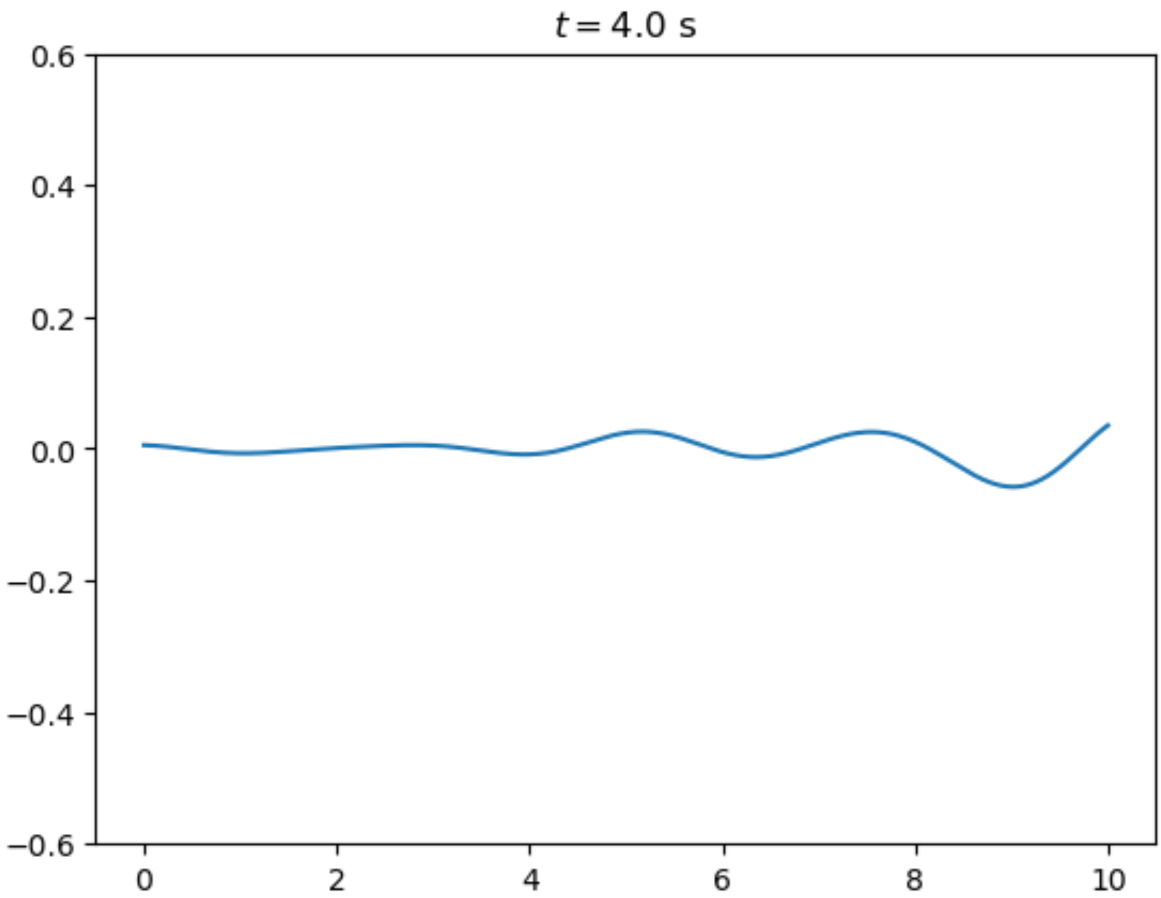

$t = 1.5$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 2.0$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

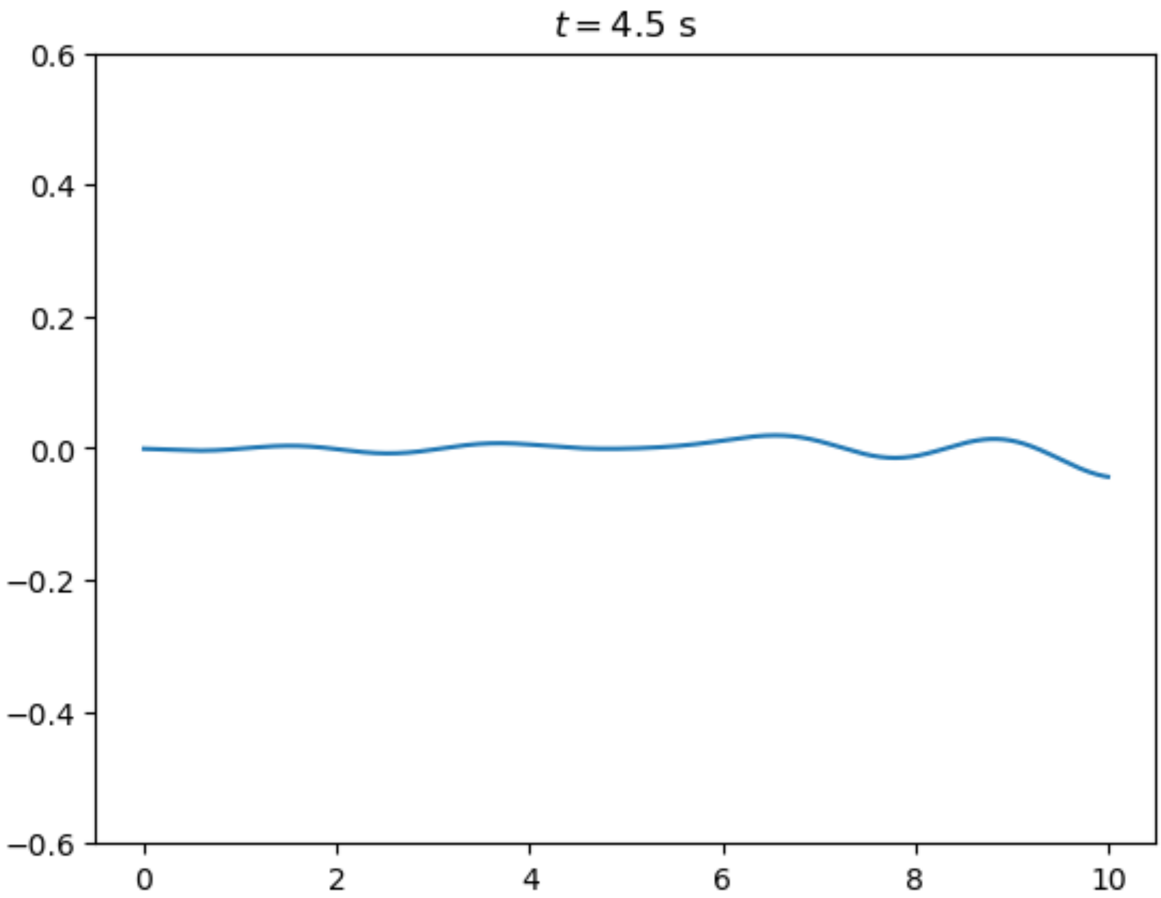

$t = 2.5$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 3.0$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10


$t = 3.5$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10



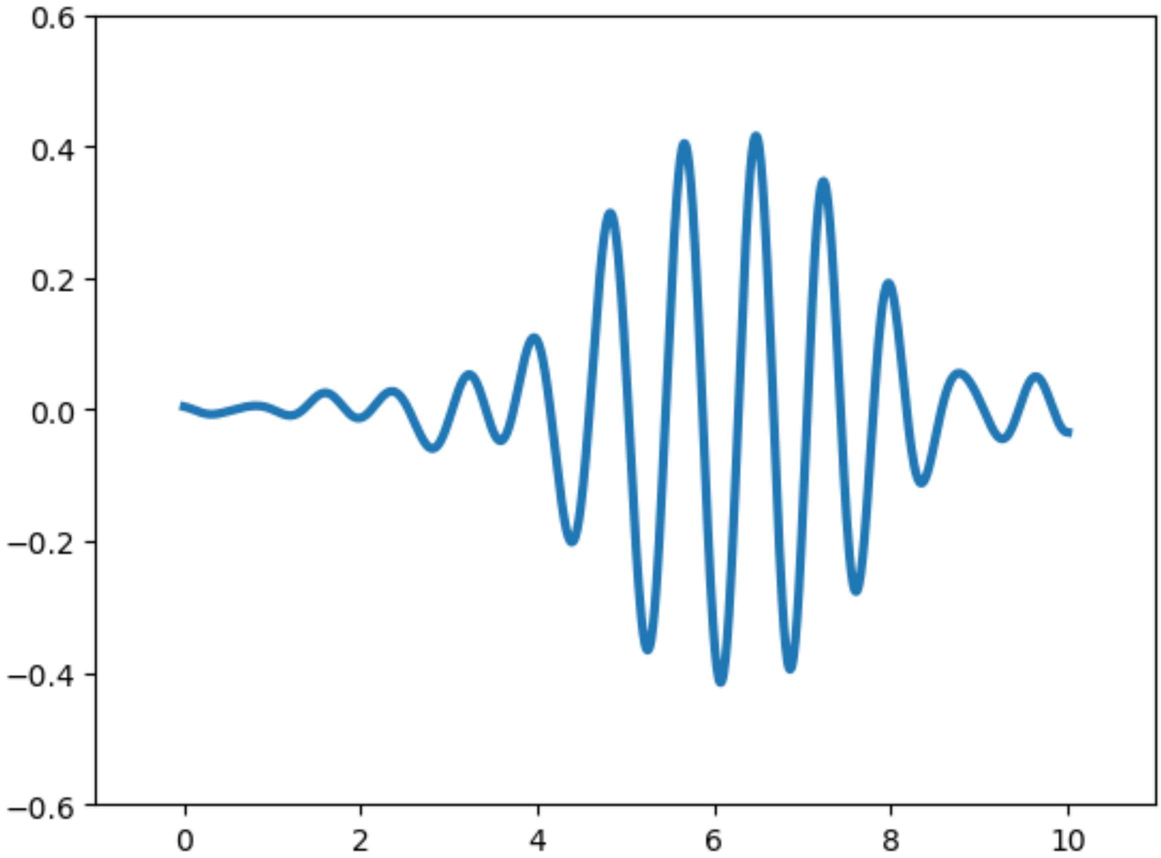
$t = 4.0$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10



$t = 4.5$ s
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
10

In [ ]:
```
N = 50

f_domain = np.linspace(0, 2*f_0, N)
lorentz_distrib = g(f_domain, f_0, f_width)

'''
fig = plt.figure()
ax = plt.axes(xlim=(-1, 11), ylim=(-20, 30))
line, = ax.plot([], [], lw=3)


def init():
    line.set_data([], [])
    return line,


def animate(i):
    x = np.linspace(0, 4, 1000)
    y = np.sin(2 * np.pi * (x - 0.01 * i))
    line.set_data(x, y)
    return line,

anim = FuncAnimation(fig, animate, init_func=init,
                               frames=200, interval=20, blit=True)
'''

df = f_domain[1]-f_domain[0]
#c = 1 # phase speed
# fixed t
alpha = 0.1

FPS = 20
rate = 1/FPS
t_domain = np.arange(0, 4, rate)

PHI = [] # time series
for t in t_domain:
#t_1 = 0
    x_domain = np.linspace(0, 10, 1000)

    phi_x = np.zeros(len(x_domain))
    for i in range(len(x_domain)):
        x = x_domain[i]
        for j in range(len(f_domain)):
            k = np.sqrt(2*np.pi*f/alpha)
            f = f_domain[j]
            phi_x[i] += lorentz_distrib[j]*np.cos(2*np.pi*f*t - k*x)*df
    PHI.append(phi_x)

    #plt.plot(x_domain, phi_1)
    #plt.title(f"$t={t}$ s")
    #plt.ylim((-20, 30))
    #plt.show()
```

In [ ]:
```
#print(len(PHI))

fig = plt.figure()
ax = plt.axes(xlim=(-1, 11), ylim=(-0.6, 0.6))
line, = ax.plot([], [], lw=3)


def init():
    line.set_data([], [])
    return line,


def animate(i):
    phi_x = PHI[i]
    line.set_data(x_domain, phi_x)
    return line,

anim = FuncAnimation(fig, animate, init_func=init,
                               frames=len(PHI), interval=FPS/2, blit=True


anim.save(f'figs/wave_packet_diff_{N=}_{alpha=}.gif', writer='imagemagick
```

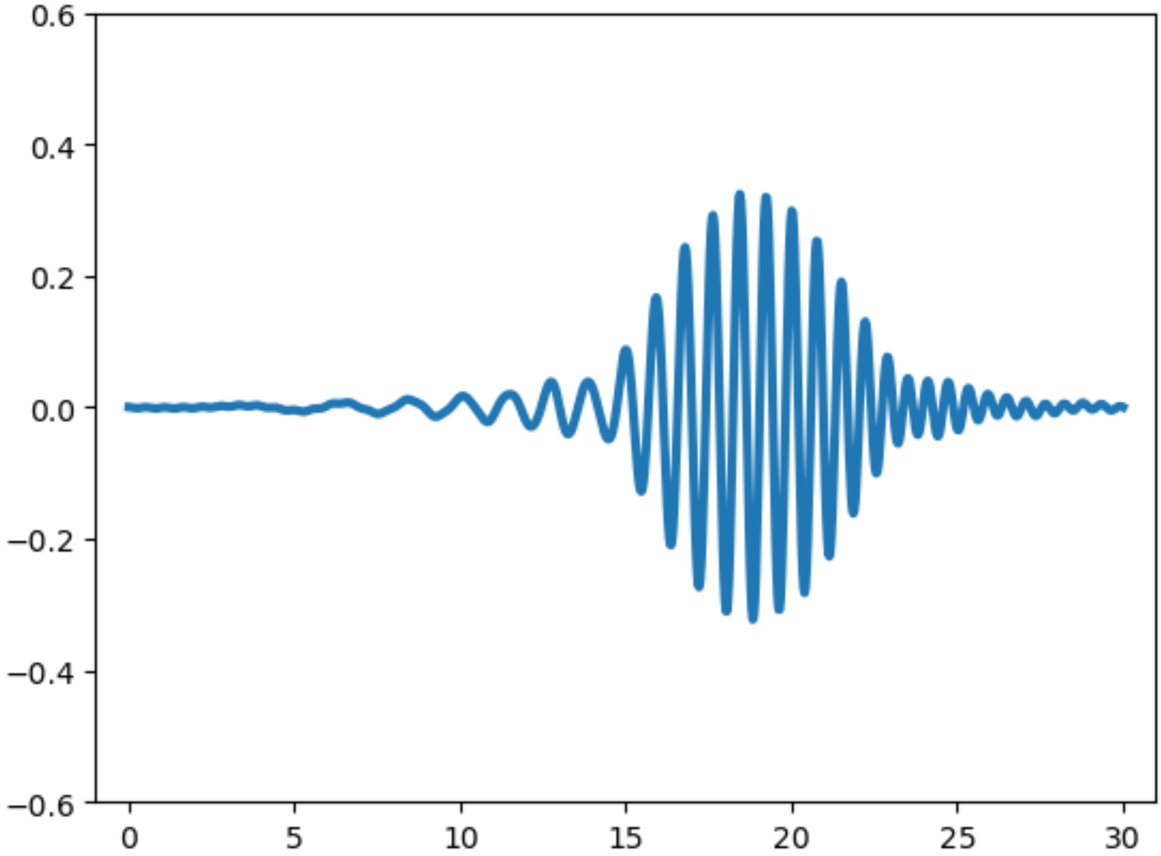


We indeed notice that the curve that modulates the plane wave moves at a different speed ; phase speed seems to be faster than group speed

Longer time interval to check wave packet spreading

In [ ]:
```
N = 100

f_domain = np.linspace(0, 2*f_0, N)
lorentz_distrib = g(f_domain, f_0, f_width)

df = f_domain[1]-f_domain[0]
#c = 1 # phase speed
# fixed t
alpha = 0.1

FPS = 20
rate = 1/FPS
t_domain = np.arange(0, 12, rate)

PHI_2 = [] # time series
for t in t_domain:
#t_1 = 0
    x_domain = np.linspace(0, 30, 1000)

    phi_x = np.zeros(len(x_domain))
    for i in range(len(x_domain)):
        x = x_domain[i]
        for j in range(len(f_domain)):
            k = np.sqrt(2*np.pi*f/alpha)
            f = f_domain[j]
            phi_x[i] += lorentz_distrib[j]*np.cos(2*np.pi*f*t - k*x)*df
    PHI_2.append(phi_x)

    #plt.plot(x_domain, phi_1)
    #plt.title(f"$t={t}$ s")
    #plt.ylim((-20, 30))
    #plt.show()
```

In [ ]:
```
#print(len(PHI))

fig = plt.figure()
ax = plt.axes(xlim=(-1, 31), ylim=(-0.6, 0.6))
line, = ax.plot([], [], lw=3)


def init():
    line.set_data([], [])
    return line,


def animate(i):
    phi_x = PHI_2[i]
    line.set_data(x_domain, phi_x)
    return line,

anim = FuncAnimation(fig, animate, init_func=init,
                               frames=len(PHI_2), interval=FPS/2, blit=Tr


anim.save(f'figs/spread_wave_packet_diff_{N=}_{alpha=}.gif', writer='imag
```



as can be seen in the gif, the wave packet does indeed become wider as it propagates through space