Homework4 sensor fusion

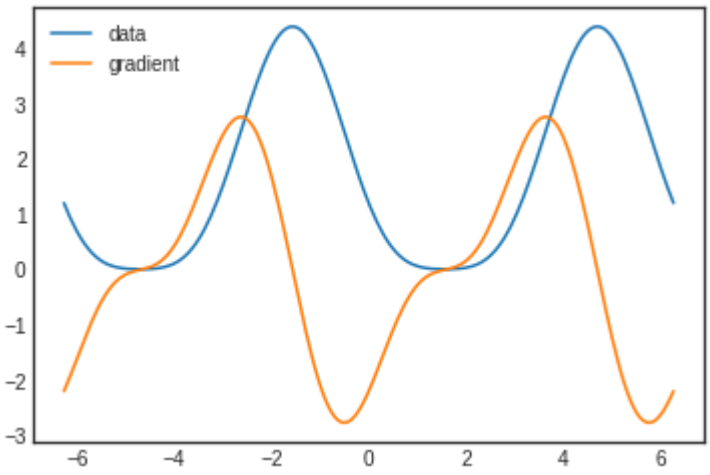
Student Vezeteu Eugeniu - 886240

In [79]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import animation
from IPython.display import HTML, Image
```

In [77]:

```
x = np.linspace(-2*np.pi,2*np.pi,200)

def Jx(x): #our function
    return (1.1 - np.sin(x))**2

def dJx(x): # function derivative
    return  -2.2*np.cos(x) + 2.*np.sin(x)*np.cos(x)

plt.plot(x,Jx(x), label='data')
plt.plot(x,dJx(x), label='gradient')
plt.legend()
plt.show()
```

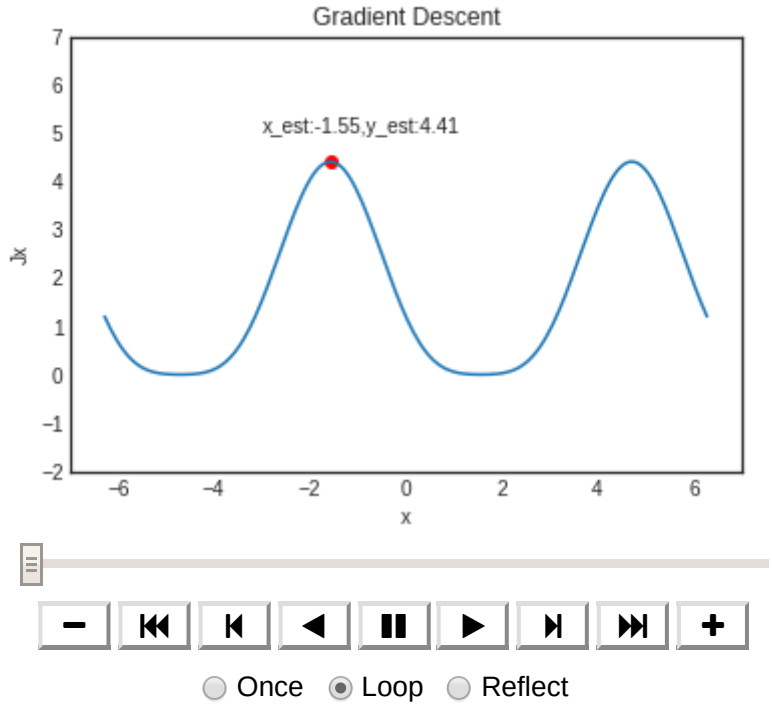


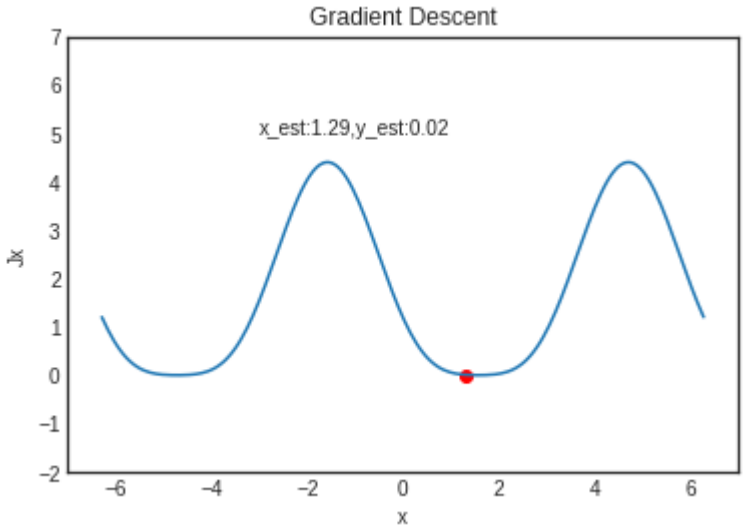
Generally the bigger step size results in a faster convergence, however, Gradient Descent with too big learning rate may overshoot the minima. To solve this issue we can start with a big step size, and during training apply decay, so we make the learning rate very small value till the end of training.

In [78]:

```
y = Jx(x)
gamma = 0.03 # 0.5# 0.1 # # learning rate
x_est = -1.55 # initial point
y_est = Jx(x_est)

iters = 150 #number of iterations

def animate(i):
    global x_est
    global y_est

    # Gradient Descent
    x_est = x_est - gamma*dJx(x_est)
    y_est = f(x_est)

    # plot
    scat.set_offsets([[x_est,y_est]])
    text.set_text("x_est:{},y_est:{}".format(round(x_est,2),round(y_est,2)))
    line.set_data(x, y)
    return line, scat, text

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

fig, ax = plt.subplots()
ax.set_xlim([-7, 7])
ax.set_ylim([-2, 7])
ax.set_xlabel("x")
ax.set_ylabel("Jx")
plt.title("Gradient Descent")
line, = ax.plot([], [])
scat = ax.scatter([], [], c="red")
text = ax.text(-3,5,"")

myAnim = animation.FuncAnimation(fig, animate, iters,
    init_func=init, interval=iters, blit=True)

HTML(myAnim.to_jshtml())
```

Out[78]:


Gradient Descent
x_est:-1.55,y_est:4.41
Jx
x
Once
Loop
Reflect


Gradient Descent
x_est:1.29,y_est:0.02
Jx
x

In [ ]: