In [73]:
```
import numpy as np
```

Problem 4

In [74]:
```
# Interpolation for the given x and y
def lagrange_interpolation(x):
    ret_val = 0

    ret_val = (((1/6325977.5) * (x-108)*(x-149.5)*(x-227)*(x-778))
              -((1/736269.4)*((x-58)*(x-149.5)*(x-227)*(x-778)))
              +((1/506321.7)*((x-58)*(x-108)*(x-227)*(x-778)))
              -((1/1250058.2)*((x-58)*(x-108)*(x-149.5)*(x-778)))
              +((1/38554773.2)*((x-58)*(x-108)*(x-149.5)*(x-227))))
    return ret_val
```

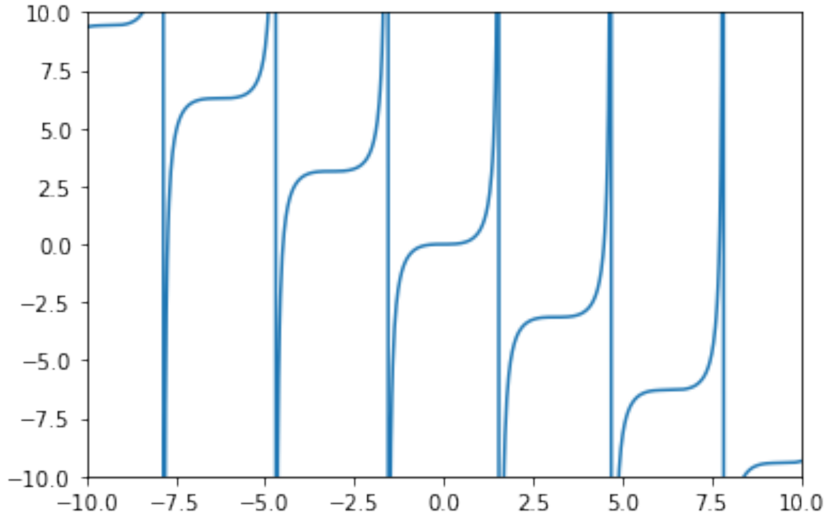
In [75]:
```
# Verification of the function for the given table of x and y
x = [58, 108, 149.5, 227, 778]
for idx, val in enumerate(x):
    y = lagrange_interpolation(val)
    y = round(y, 2)
    print(str(val) + ' = ' + str(y))
```

```
58 = 88.0
108 = 224.7
149.5 = 365.3
227 = 687.0
778 = 4332.97
```

Problem 3

In [76]:
```
#Plotting the graph for reference
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-4 * np.pi, 4 * np.pi, 1000)
plt.plot(x, np.tan(x)-x)
plt.ylim(-10, 10)
plt.xlim(-10, 10)
```

Out[76]: (-10.0, 10.0)



In [77]:

```
# Given function in the problem (tanx - x)
def tan_x(x):
    return np.tan(x) - x
```

In [78]:

```
# Derivative of (tanx - x = sec^2(x) - 1)
def tan_x_derivative(x):
    sec = 1/np.cos(x)
    return sec**2 - 1
```

In [79]:

```
def Newton_RootFinding(x):
    new_val = 0
    old_val = round(x, 5)
    for i in range(5):
        fx = round(tan_x(old_val), 5) #Value of the function
        dfx = round(tan_x_derivative(old_val), 5) #Value of the derivative
        new_val = old_val - (fx/dfx) #Updated value (xn = xn+1 * (f(xn)/f'(xn
        new_val = round(new_val, 5) #Roundign off the value to 5 decimal plac
        print(old_val, new_val)
        old_val = new_val
```

In [80]:

```
# For an intitial value of 4.4
x = 4.4
Newton_RootFinding(x)
```

```
4.4 4.53598
4.53598 4.50186
4.50186 4.49375
4.49375 4.49341
4.49341 4.49341
```

In [81]:
```
# For an intitial value of 7.7
x = 7.7
Newton_RootFinding(x)
```

```
7.7 7.73028
7.73028 7.72545
7.72545 7.72525
7.72525 7.72525
7.72525 7.72525
```

Problem 5

In [82]:
```
def alices_internet(x):
    return round((np.exp(x) - 4*x + 1), 4) #Accurate root position becomes fi
```

In [83]:
```
# Method to calculate root using bisection method
# This is a helper method added to calculate the root accurately for the give
# provided the decimal is rounded off to 4 places
# This function does not scale really well for other f(x)'s
def bisection_method(lower, upper):
    u = alices_internet(upper)
    l = alices_internet(lower)
    if u == 0 or l == 0:
        print("Root is at: ", upper if u == 0 else lower)
    if ((u * l) < 0):
        print(lower, upper)
        bisection_method((upper+lower)/ 2, upper) # Recursive call with upper
        bisection_method(lower, (upper+lower)/2)  # Recursive call with lower
    else:
        return
```

In [84]:
```
bisection_method(0, 1)
```

```
0 1
0.5 1
0.75 1
0.75 0.875
0.8125 0.875
0.8125 0.84375
0.8125 0.828125
0.8125 0.8203125
0.8125 0.81640625
0.814453125 0.81640625
0.814453125 0.8154296875
0.814453125 0.81494140625
0.814453125 0.814697265625
0.814453125 0.8145751953125
Root is at:  0.81451416015625
Root is at:  0.81451416015625
```

In [ ]: