Week 9 Newton

November 21, 2024

[3]:
```
import matplotlib.pyplot as plt
import numpy as np
import math
import random
```

This code implements Newton Method for find roots to the equation

$$f(x) = \frac{x^2}{4} + \frac{x}{4} - 5$$

[36]:
```
def newton (start, numsteps):
    x = start
    print("n  x")
    for n in range (numsteps):
        print (n, x)
        y = (x**2)/4 + x/4 -5
        yprime = x/2 + 1/4
        if yprime == 0:
            print ("Found f'(x) = 0, function could not complete")
            return
        x = x - y/yprime
```

Running the code with random generated inputs between -10 and +10 resulted in finding both roots in 2-4 tries the few times I did it. A starting point of -1/2 would result in a divide by zero error and because f'(x) = 0 and therefore would never reintercept the x axis. I added code to handle this.

[37]:
```
newton(random.randrange(-10, 10,1), 8)
newton(random.randrange(-10, 10,1), 8)
newton(random.randrange(-10, 10,1), 8)
newton(random.randrange(-10, 10,1), 8)
newton(-1/2, 8)
```

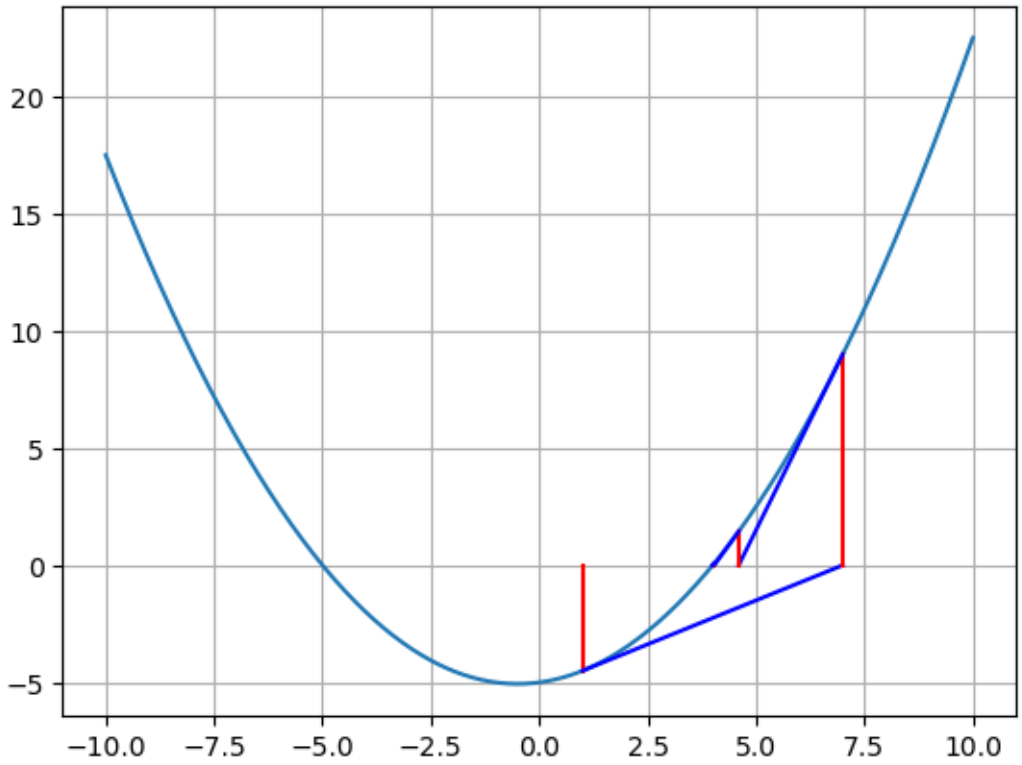
```
n  x
0 1
1 7.0
2 4.6
3 4.035294117647059
4 4.00013733119707
```

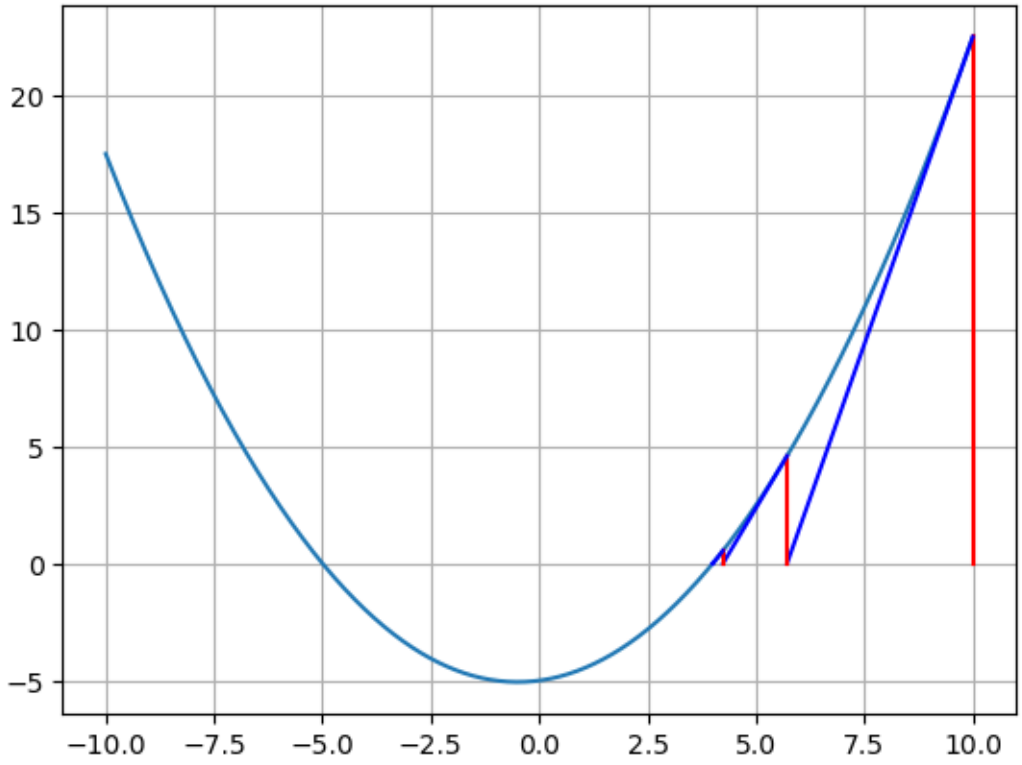
```
5 4.000000002095476
6 4.0
7 4.0
n  x
0 2
1 4.8
2 4.060377358490566
3 4.000399684623852
4 4.000000017748179
5 4.0
6 4.0
7 4.0
n  x
0 2
1 4.8
2 4.060377358490566
3 4.000399684623852
4 4.000000017748179
5 4.0
6 4.0
7 4.0
n  x
0 0
1 20.0
2 10.24390243902439
3 5.814346225187563
4 4.260664533386002
5 4.007136188497304
6 4.000005649395108
7 4.0000000000035465
n  x
0 -0.5
Found f'(x) = 0, function could not complete
```

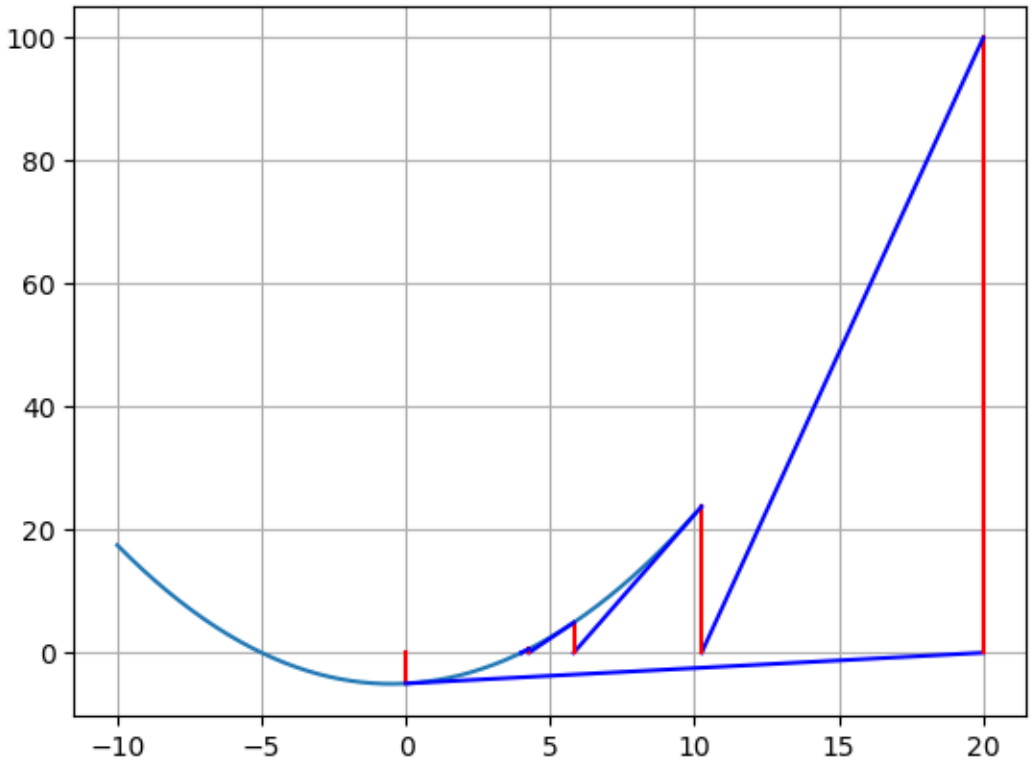
[28]:
```python
def fx(x):
    return (x**2)/4 + x/4 - 5

def fxprime(x):
    return x/2 + 1/4



def newtonplot(x, steps):
    # this portion plots the original function
    xvals = np.linspace(-10, 10, 1000)
    plt.plot(xvals, fx(xvals))
    plt.grid(True)
```

```
    for i in range(steps):

        y = fx(x)
        yprime = fxprime(x)

        # handle error if we find a yprime = 0
        if yprime == 0:
            print ("Found f'(x) = 0, function could not complete")
            return

        # Plotting (x, 0) to (x, y)
        xplot1 = [x, x]
        yplot1 = [0, y]
        plt.plot(xplot1, yplot1, color="red")

        # Calculating x1 using Newton's method
        x1 = x - y/yprime

        # Plotting tangent line from (x, y) to (x1, 0)
        xplot2 = [x, x1]
        yplot2 = [y, 0]
        plt.plot(xplot2, yplot2, color="blue")
        x = x1 #update x for next round
    plt.show()
```

[39]:
```
# Plot Newton's method for a random x for 8 steps
x= random.randrange(-10, 10,1)
steps = 8
newtonplot(x,steps)

#Plot Newton's method for x=10 to demonstrate more steps
newtonplot(10, steps)

newtonplot (0, steps)
```


20
15
10
5
0
−5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


20
15
10
5
0
−5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


100
80
60
40
20
0
−10
−5
0
5
10
15
20

[ ]: