Length Program

November 21, 2024

Length Program

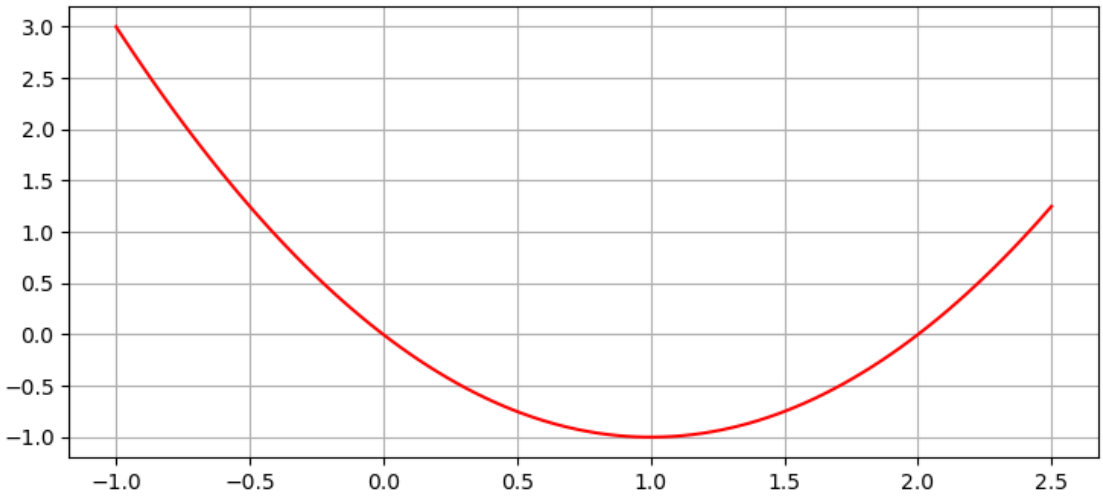
[13]:
```
import numpy as np
from matplotlib import pyplot as plt

plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True

def f(x):
   return x**2 - 2*x

x = np.linspace(-1, 2.5, 100)

plt.plot(x, f(x), color='red')
plt.grid(True)
plt.show()
```



Answers to 2.2 through 2.5 in the code comments below.

[2]:
```
import math
```

```
def fnf(x):  return x**2  # this instructs the program to work with f(x) = x^2
xinitial = 0
xfinal = 1
numberofsteps = 2 # gives the number of segments to be measured (Question 3)
deltax = (xfinal-xinitial)/numberofsteps
total = 0
for k in range(numberofsteps):
    x1 = xinitial + (k-1)*deltax # gives the left endpoint x value
    xr = xinitial + k*deltax #gives the right endpoint x value
    y1 = fnf(x1) # gives the left endpoint y value
    yr = fnf(xr) # gives the right endpoint y value
    segment = math.sqrt((xr-x1)**2 + (yr - y1)**2) #calculates the k-th segment␣
 ↪length. Base is xr-x1 and hight is yr-y1
    total = total + segment
    print(k, segment)
print (numberofsteps, total)
```

```
0 0.5590169943749475
1 0.5590169943749475
2 1.118033988749895
```

[3]:
```
import math


def fnf(x):  return x**2  # this instructs the program to work with f(x) = x^2

def length(z, xfin):
    xinitial = 0
    xfinal = xfin
    numberofsteps = z # gives the number of segments to be measured (Question 3)
    deltax = (xfinal-xinitial)/numberofsteps
    total = 0
    for k in range(numberofsteps):
        x1 = xinitial + (k-1)*deltax # gives the left endpoint x value
        xr = xinitial + k*deltax #gives the right endpoint x value
        y1 = fnf(x1) # gives the left endpoint y value
        yr = fnf(xr) # gives the right endpoint y value
        segment = math.sqrt((xr-x1)**2 + (yr - y1)**2) #calculates the k-th␣
 ↪segment length. Base is xr-x1 and hight is yr-y1
        total = total + segment
        #print(k, segment)
    print (numberofsteps, total)
    return total
```

[4]:
```
# Answer to 2.3 - 7
length(200,1)
```

```
length(2000,1)
length(20000,1)
length(200000,1)
length(2000000,1)
```

```
200 1.4727830718444779
2000 1.478325028585655
20000 1.4788810561955084
200000 1.4789366772252241
2000000 1.4789422395108391
```

[4]:
```
1.4789422395108391
```

[5]:
```
#Answer to 2.3 - 8

length(20000000,1)
length(30000000,1)
length(100000000,1)

#honestly not sure how many steps I'd have to go to to get stabilized at 8␣
 ↪steps
```

```
20000000 1.4789427957414765
30000000 1.478942816342523
100000000 1.4789428451835969
```

[5]:
```
1.4789428451835969
```

[6]:
```
xtry = 3.03
len = 0
while (len<=10):
    len = length(2000, xtry)
    print(xtry, len)
    xtry = xtry + .001

#based on this the lenght is ten around x = 3.042. I solved this by reducing␣
 ↪the inverval I increased x by
```

```
2000 9.922671485620338
3.03 9.922671485620338
2000 9.92880886621017
3.0309999999999997 9.92880886621017
2000 9.934948218131597
3.0319999999999996 9.934948218131597
2000 9.94108954140183
3.0329999999999995 9.94108954140183
2000 9.94723283603816
3.0339999999999994 9.94723283603816
```

```
2000 9.953378102057727
3.0349999999999993 9.953378102057727
2000 9.959525339477736
3.035999999999999 9.959525339477736
2000 9.965674548315391
3.036999999999999 9.965674548315391
2000 9.971825728587804
3.037999999999999 9.971825728587804
2000 9.97797888031215
3.038999999999999 9.97797888031215
2000 9.98413400350553
3.0399999999999987 9.98413400350553
2000 9.990291098185047
3.0409999999999986 9.990291098185047
2000 9.996450164367797
3.0419999999999985 9.996450164367797
2000 10.00261120207087
3.0429999999999984 10.00261120207087
```

[7]:
```
import math


def fnf(x):  return x**3  # this instructs the program to work with f(x) = x^2

def lengthcube(z, xfin):
    xinitial = 0
    xfinal = xfin
    numberofsteps = z # gives the number of segments to be measured (Question 3)
    deltax = (xfinal-xinitial)/numberofsteps
    total = 0
    for k in range(numberofsteps):
        x1 = xinitial + (k-1)*deltax # gives the left endpoint x value
        xr = xinitial + k*deltax #gives the right endpoint x value
        y1 = fnf(x1) # gives the left endpoint y value
        yr = fnf(xr) # gives the right endpoint y value
        segment = math.sqrt((xr-x1)**2 + (yr - y1)**2) #calculates the k-th␣
 ↪segment length. Base is xr-x1 and hight is yr-y1
        total = total + segment
        #print(k, segment)
    print (numberofsteps, total)
    return total
```

[8]:
```
#for 2.3 10 I modified the original program. I don't think the coding.csel has␣
 ↪the power to solve all the way to 8 decimal places so this is the furthest I␣
 ↪ran it to.
lengthcube(1000000,1)
lengthcube(2000000,1)
```

```
1000000 1.5478634924086776
2000000 1.5478645735454883
```

[8]: 1.5478645735454883