mathhw2

October 14, 2021

[ ]:
```
#Zack Wang HW 2
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
import math
```

[ ]:
```
points = []
lengths = []
def approximate(foo, a, b, tolerance, iterations = 0, computations = 0):
    midpoint = (a + b) / 2
    #print("A is {}, mid is {}, b is {}".format(a, midpoint, b))
    coarse = calcInt(foo, a, b)
    fine = calcInt(foo, a, midpoint) +  calcInt(foo, midpoint, b)
    err = (fine - coarse) / 3
    if iterations >= 30 or (err < tolerance and iterations > 3):
        points.append(a)
        points.append(b)
        lengths.append(b-a)
        lengths.append(b-a)
        return fine, err, iterations, computations
    else:
        lval, lerr, liter, lcomp = approximate(foo, a, midpoint, tolerance/2,␣
 ↪iterations + 1, computations + 3)
        rval, rerr, riter, rcomp = approximate(foo, midpoint, b, tolerance/2,␣
 ↪iterations + 1, computations + 3)
        return lval + rval, lerr + rerr, max(liter, riter), lcomp + rcomp

t1 = 10 ** -3
t2 = 10 ** -5
t3 = 10 ** -7
t4 = 10 ** -9
```

[ ]:
```
def calcInt(bar, a, b):
    return (b - a) * bar((a + b)/2)
```

extrapolated integral val = approx val + approx error

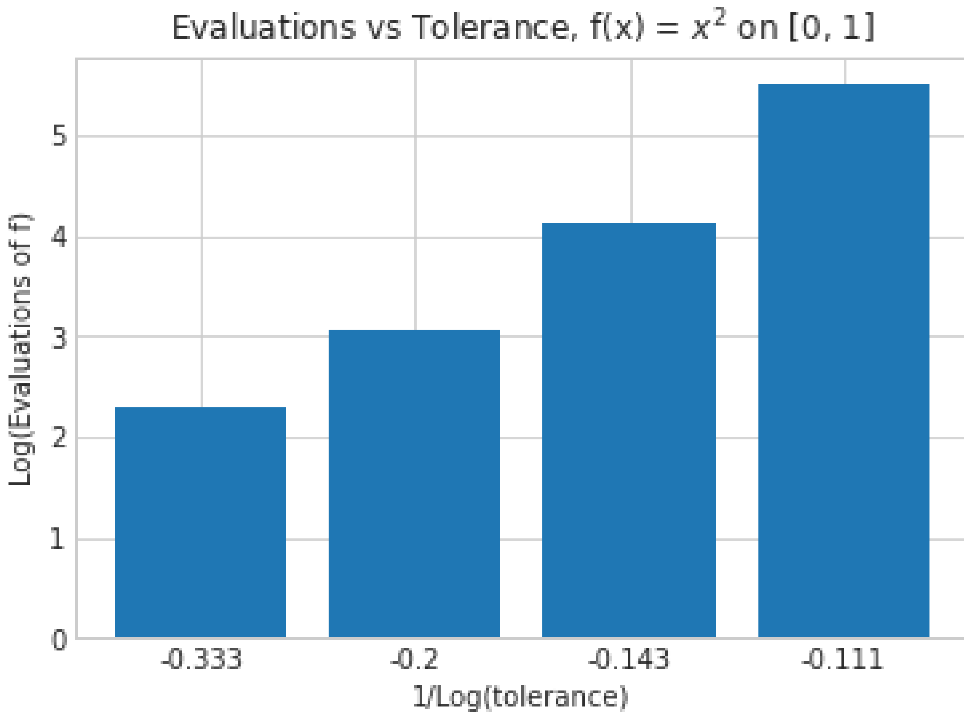
actual err = er4

actual extrapol err = extrapolated integral val - v4

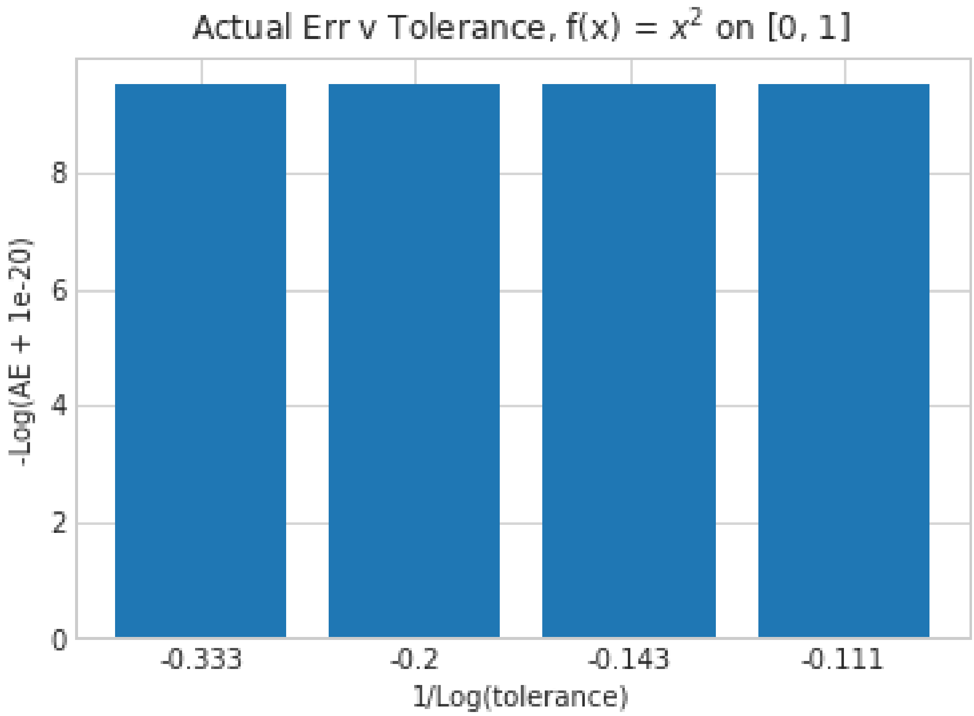
1 SECTION 1 X^2

[ ]:
```
v1, er1, d1, i1 =  approximate(lambda x: x**2, 0, 1, t1)
v2, er2, d2, i2 =  approximate(lambda x: x**2, 0, 1, t2)
v3, er3, d3, i3 =  approximate(lambda x: x**2, 0, 1, t3)
v4, er4, d4, i4 =  approximate(lambda x: x**2, 0, 1, t4)
eiv1 = v1 + er1
eiv2 = v2 + er2
eiv3 = v3 + er3
eiv4 = v4 + er4
aee1 = eiv1 - v4
aee2 = eiv2 - v4
aee3 = eiv3 - v4
aee4 = eiv4 - v4
```

[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Evaluations vs Tolerance, f(x) = $x^2$ on [0, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("Log(Evaluations of f)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
 ↪list(map(lambda y: math.log(y, 10), evals)))
plt.show()
```



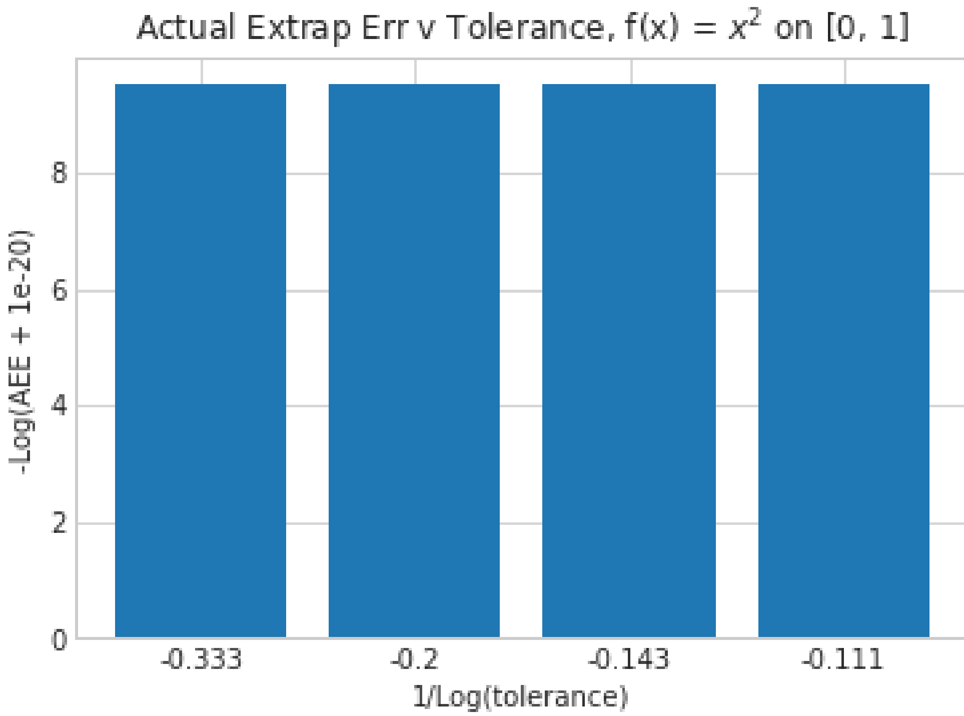
[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Actual Err v Tolerance, f(x) = $x^2$ on [0, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AE + 1e-20)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
 ↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), [er4] * 4)))
plt.show()
```



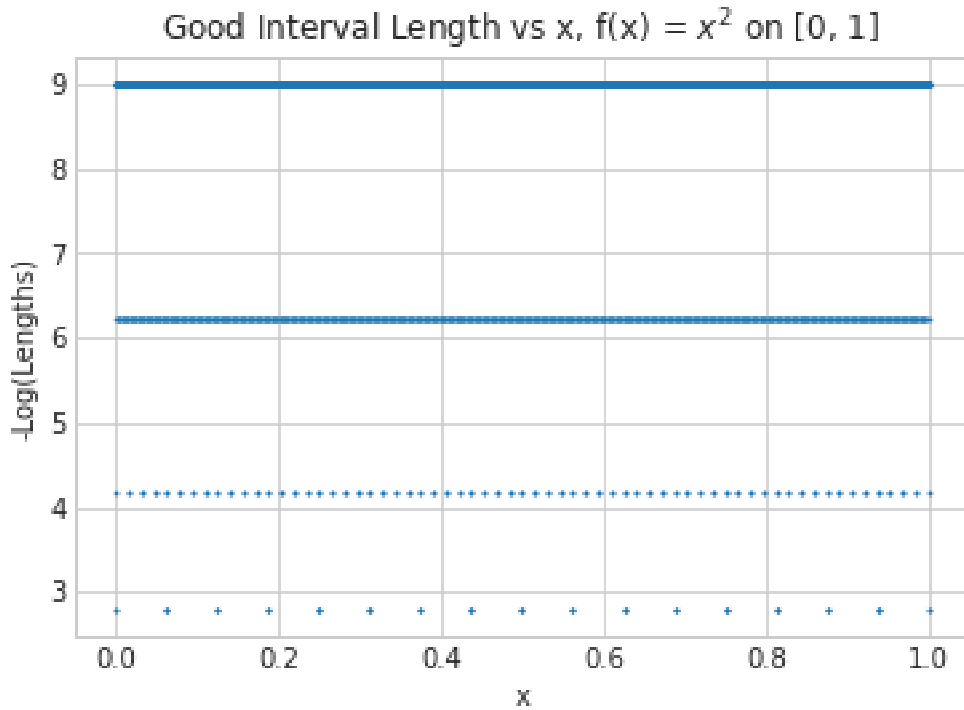
[ ]:
```
-math.log(er4, 10)
```

[ ]: 9.508021124639097

[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Actual Extrap Err v Tolerance, f(x) = $x^2$ on [0, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AEE + 1e-20)")
tols = [t1, t2, t3, t4]
aees = [aee1, aee2, aee3, aee4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),
 ↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), aees)))
plt.show()
```



[ ]:
```
fig = plt.figure()
ax = plt.axes()

v1, er1, d1, i1 =  approximate(lambda x: x**2, 0, 1, t1)

plt.title("Good Interval Length vs x, f(x) = $x^2$ on [0, 1]")
plt.xlabel("x")
plt.ylabel("-Log(Lengths)")
ax.scatter(points, list(map(lambda y: -1 * math.log(y), lengths)), s = .5)
plt.show()
```




```
[ ]: eiv1, eiv2, eiv3, eiv4
     er1, er2, er3, er4
```

```
[ ]: (8.138020833333333e-05,
      5.086263020833333e-06,
      7.947285970052083e-08,
      3.104408582051595e-10)
```

Plot 1: As tolerance gets larger, so too does the number of evaluations of f. (the log of both, that is)

Plot 2: Actual error is a constant that is nearly equal to the actual extrapolated error.

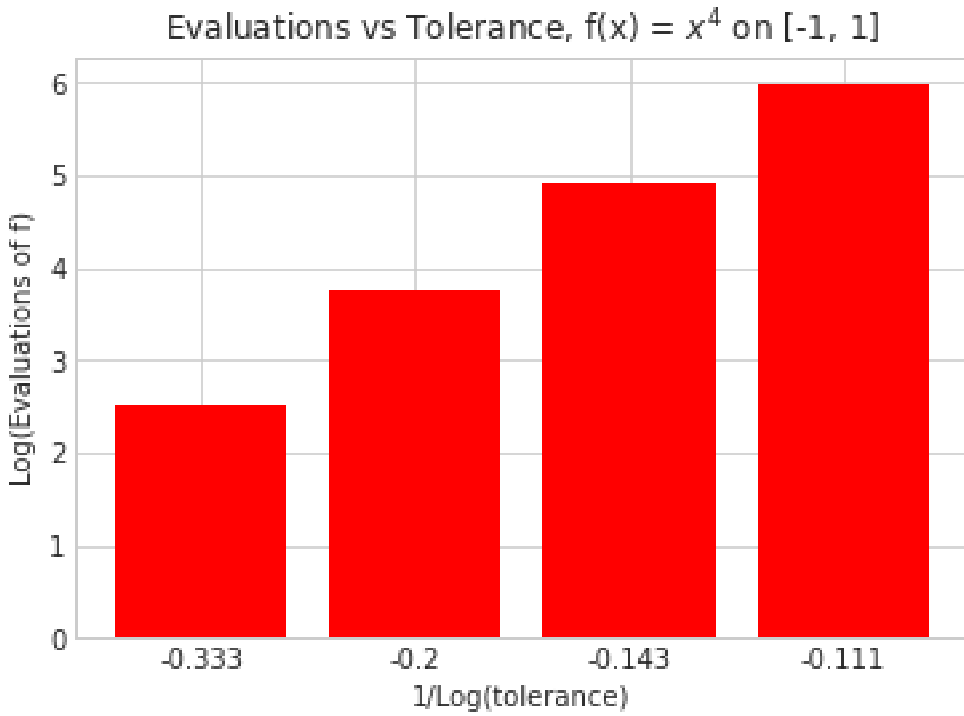
Plot 3: We pretty much cannot tell the difference between the actual extrapolated error terms for this function. Each call, regardless of tolerance, is very close to the actual value, meaning the difference between each is negigible.

Plot 4: We see that the good interval points where the intervals are large (f is easy to approximate) seem almost periodic. If we look at the scatterplot the points make multiple parabolas.

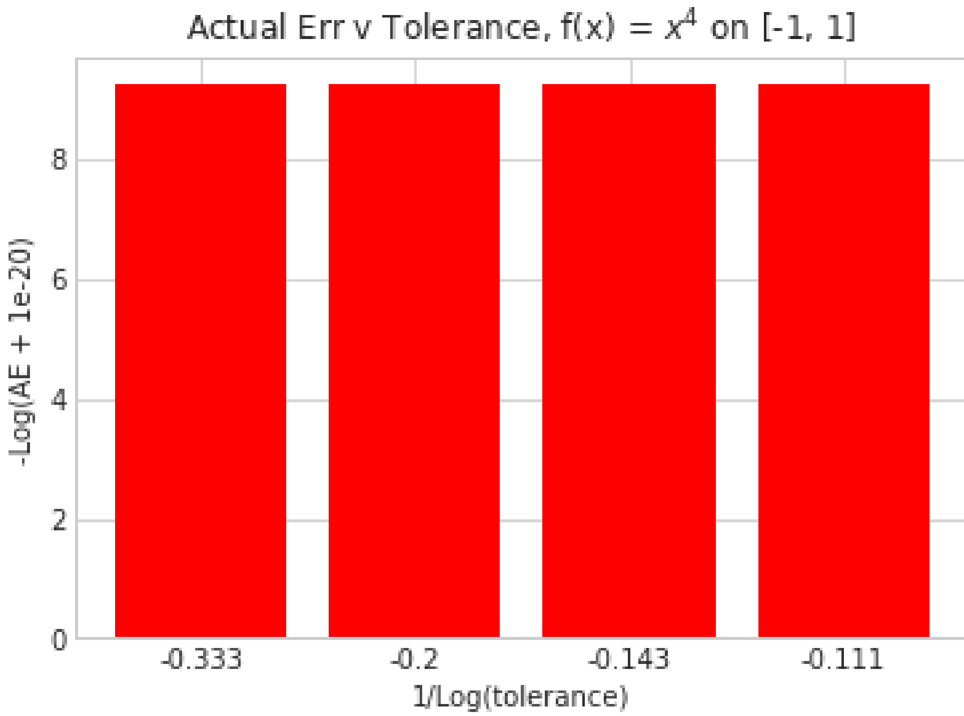
2 SECTION 2 x^4

[ ]:
```
v1, er1, d1, i1 =  approximate(lambda x: x**4, -1, 1, t1)
v2, er2, d2, i2 =  approximate(lambda x: x**4, -1, 1, t2)
v3, er3, d3, i3 =  approximate(lambda x: x**4, -1, 1, t3)
v4, er4, d4, i4 =  approximate(lambda x: x**4, -1, 1, t4)
eiv1 = v1 + er1
eiv2 = v2 + er2
eiv3 = v3 + er3
eiv4 = v4 + er4
aee1 = eiv1 - v4
aee2 = eiv2 - v4
aee3 = eiv3 - v4
aee4 = eiv4 - v4
```

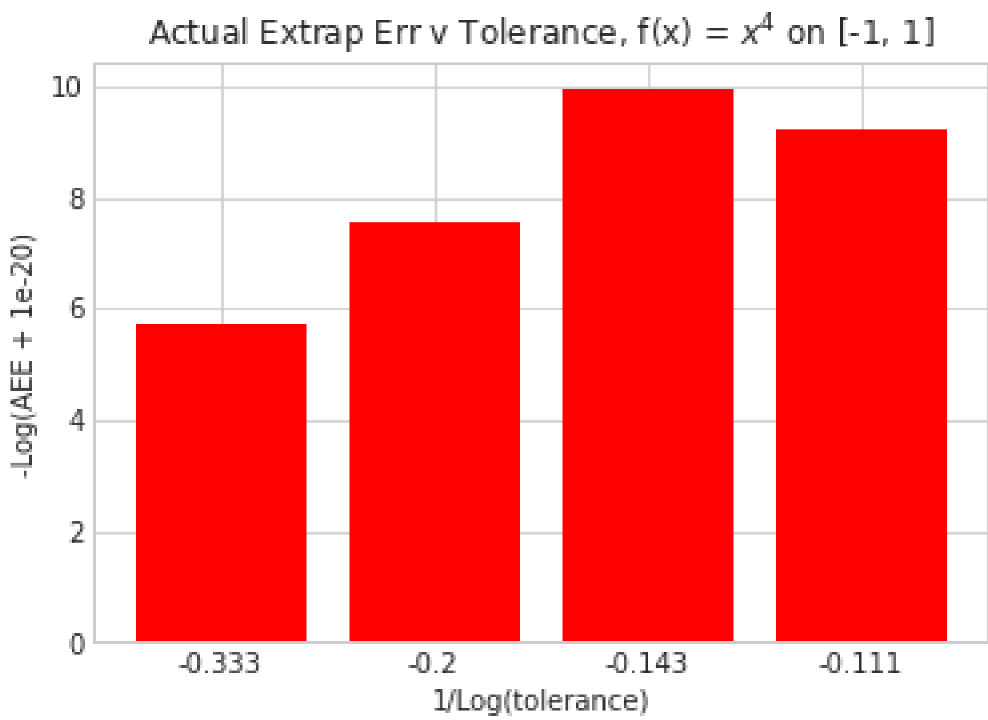
[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Evaluations vs Tolerance, f(x) = $x^4$ on [-1, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("Log(Evaluations of f)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
 ↪list(map(lambda y: math.log(y, 10), evals)), color = 'red')
plt.show()
```



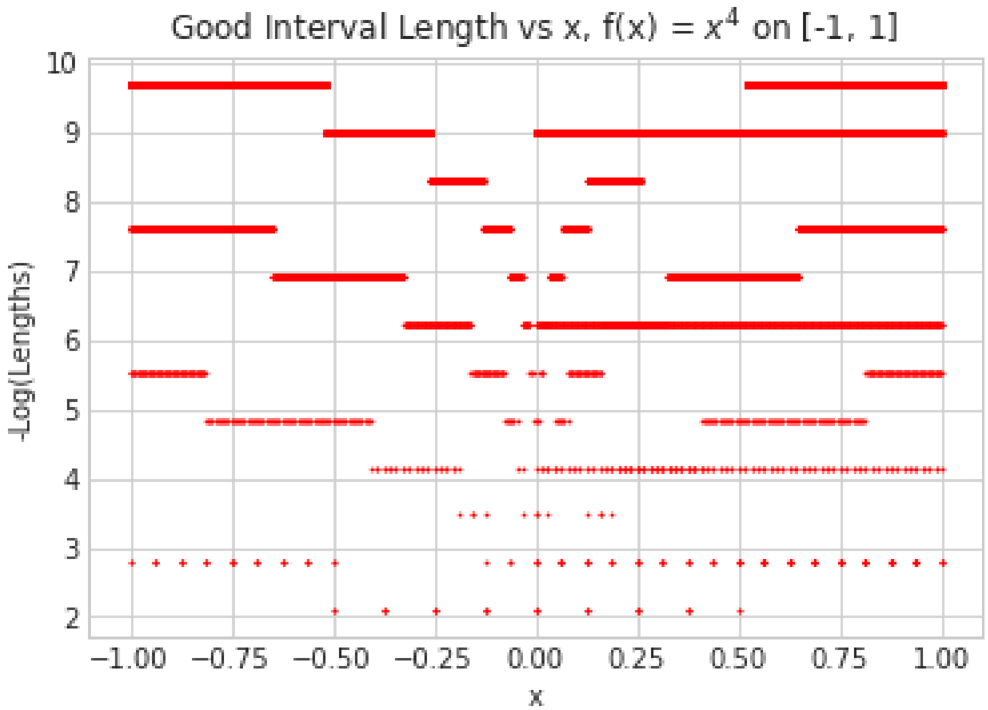

```
[ ]: fig = plt.figure()
ax = plt.axes()

plt.title("Actual Err v Tolerance, f(x) = $x^4$ on [-1, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AE + 1e-20)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), [er4] * 4)),␣
↪color = 'red')
plt.show()
```



[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Actual Extrap Err v Tolerance, f(x) = $x^4$ on [-1, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AEE + 1e-20)")
tols = [t1, t2, t3, t4]
aees = [aee1, aee2, aee3, aee4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),
 ↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), aees)), color =
 ↪'red')
plt.show()
```



[ ]:
```
fig = plt.figure()
ax = plt.axes()

v1, er1, d1, i1 =  approximate(lambda x: x**4, -1, 1, t1)

plt.title("Good Interval Length vs x, f(x) = $x^4$ on [-1, 1]")
plt.xlabel("x")
plt.ylabel("-Log(Lengths)")
ax.scatter(points, list(map(lambda y: -1 * math.log(y), lengths)), s = .5,␣
 ↪color = "red")
plt.show()
```



Plot 1: As tolerance gets larger, so too does the number of evaluations of f. (the log of both, that is). Same as example 1.

Plot 2: Actual error is a constant, looks the same magnitude as in example 1.

Plot 3: Actual extrapolated error differs much more in this example. Each call, depending on tolerance, produced a noticably different estimation of the integral.

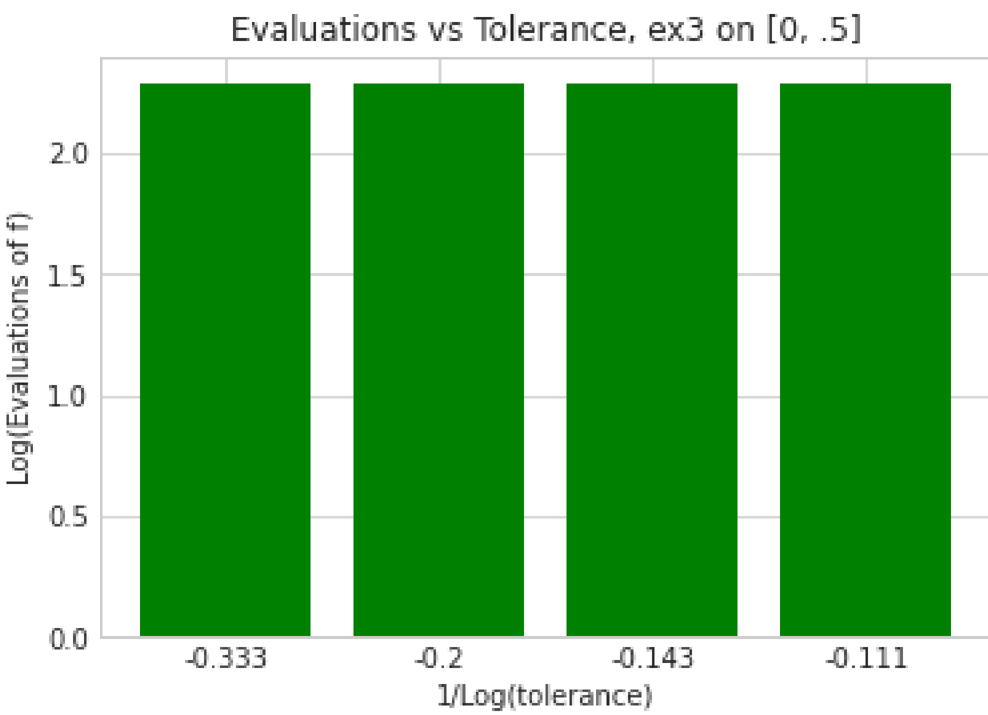
Plot 4: We see that the good interval points form a loose U shape. The points at the ends, near x = -1 and 1, indicate that intervals are shorter. This makes sense as x^4 is changing at a faster rate at those regions.

3 SECTION 3 $SQRT(sin(pi * x))$

[ ]:
```
v1, er1, d1, i1 =  approximate(lambda x: math.sqrt(math.sin(math.pi * x)), 0, .
 ↪5, t1)
v2, er2, d2, i2 =  approximate(lambda x: math.sqrt(math.sin(math.pi * x)), 0, .
 ↪5, t2)
v3, er3, d3, i3 =  approximate(lambda x: math.sqrt(math.sin(math.pi * x)), 0, .
 ↪5, t3)
v4, er4, d4, i4 =  approximate(lambda x: math.sqrt(math.sin(math.pi * x)), 0, .
 ↪5, t4)
eiv1 = v1 + er1
```

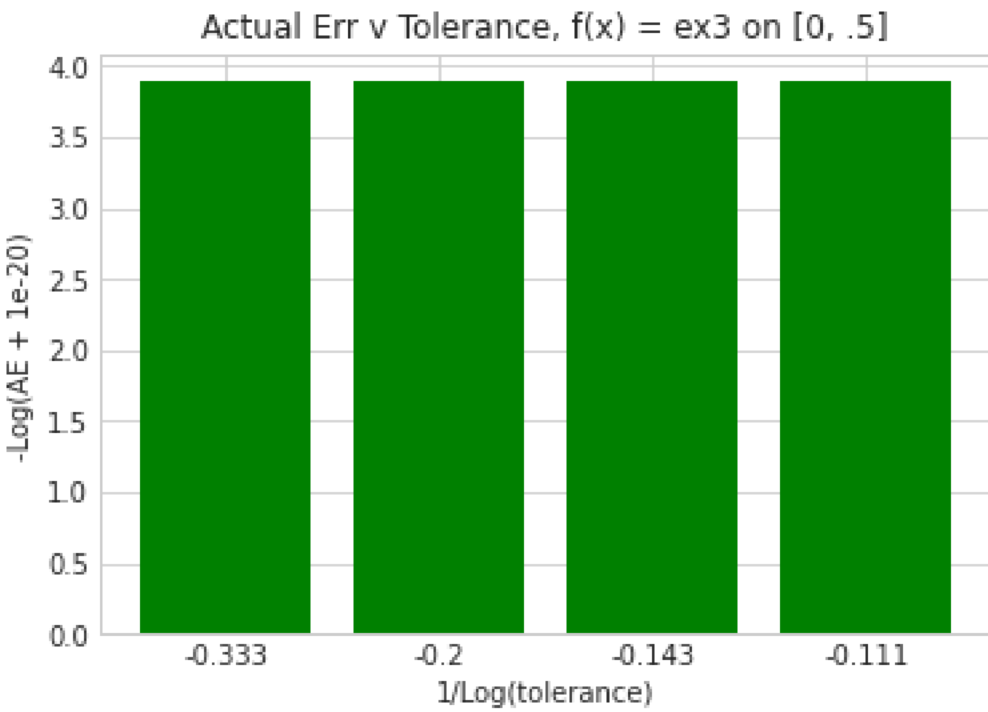
```
eiv2 = v2 + er2
eiv3 = v3 + er3
eiv4 = v4 + er4
aee1 = eiv1 - v4
aee2 = eiv2 - v4
aee3 = eiv3 - v4
aee4 = eiv4 - v4
```

[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Evaluations vs Tolerance, ex3 on [0, .5]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("Log(Evaluations of f)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
 ↪list(map(lambda y: math.log(y, 10), evals)), color = 'green')
plt.show()
```

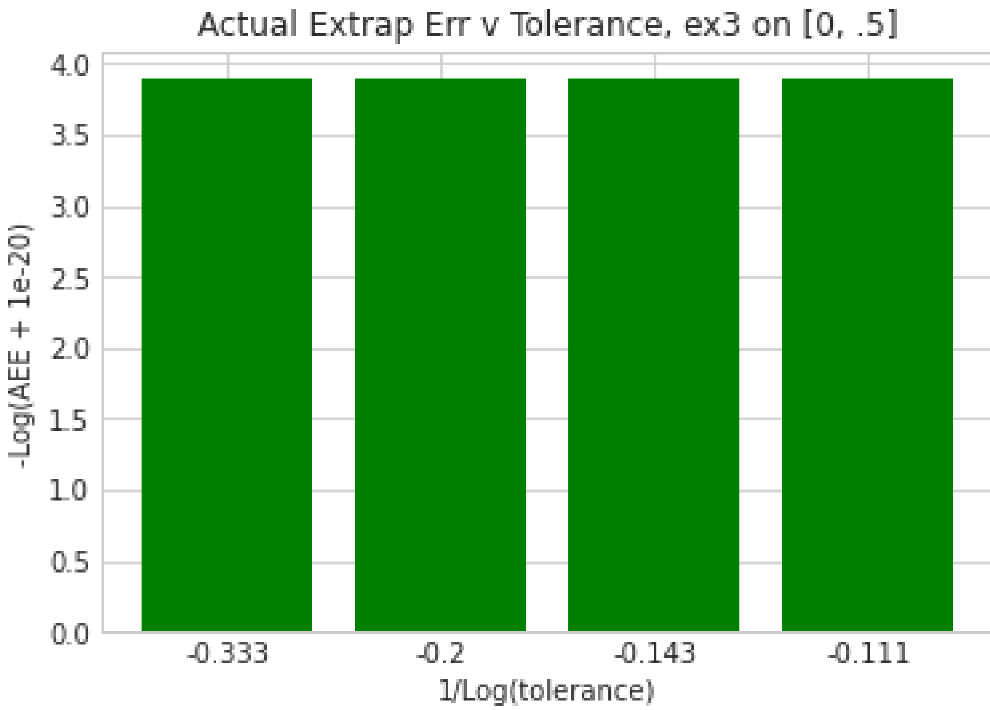


[ ]:
```
fig = plt.figure()
ax = plt.axes()
```

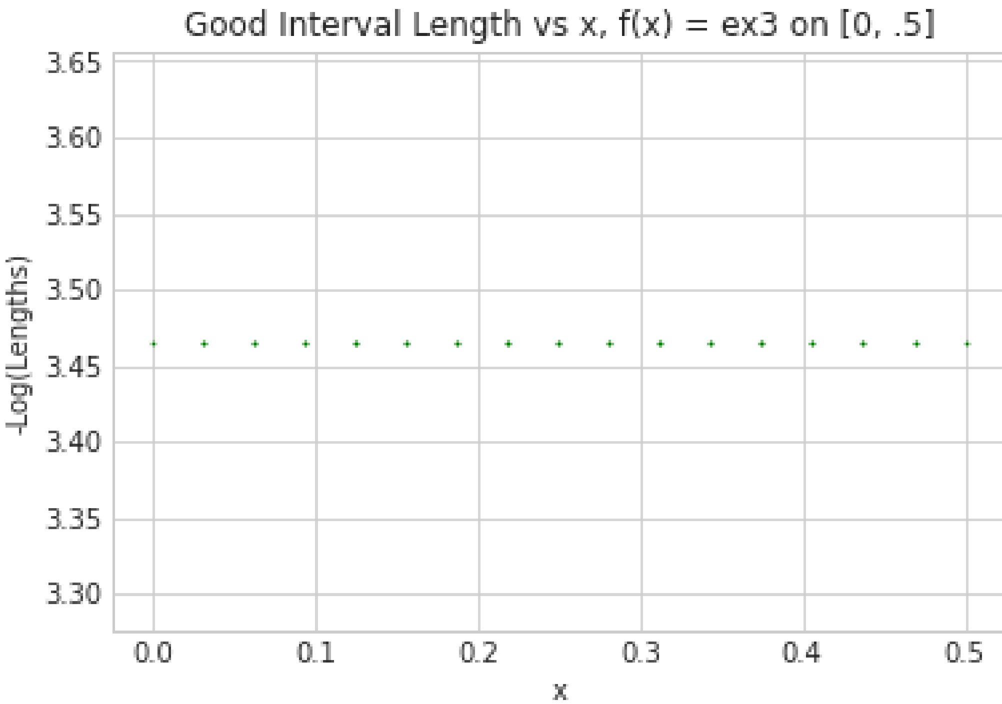
```
plt.title("Actual Err v Tolerance, f(x) = ex3 on [0, .5]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AE + 1e-20)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), [er4] * 4)),␣
↪color = 'green')
plt.show()
```




```
[ ]: fig = plt.figure()
ax = plt.axes()

plt.title("Actual Extrap Err v Tolerance, ex3 on [0, .5]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AEE + 1e-20)")
tols = [t1, t2, t3, t4]
aees = [aee1, aee2, aee3, aee4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), aees)), color =␣
↪'green')
plt.show()
```



[ ]:
```
fig = plt.figure()
ax = plt.axes()

points = []
lengths = []
v1, er1, d1, i1 = approximate(lambda x: math.sqrt(math.sin(math.pi * x)), 0, .
 ↪5, t1)

plt.title("Good Interval Length vs x, f(x) = ex3 on [0, .5]")
plt.xlabel("x")
plt.ylabel("-Log(Lengths)")
ax.scatter(points, list(map(lambda y: -1 * math.log(y), lengths)), s = .5,␣
 ↪color = "green")
plt.show()
```



Plot 1: Tolerance does not seem to affect number of evaluations. Every function attempt is running the maximum number of times.

Plot 2: Actual error is a constant, same as the actual extrapolated error. -Log Actual error is less than prior examples at around 3, meaning it was harder to estimate this function.

Plot 3: Because each tolerance level is running the same number of times until it maxes out, the approximate extrapolated error is the same for every tolerance.

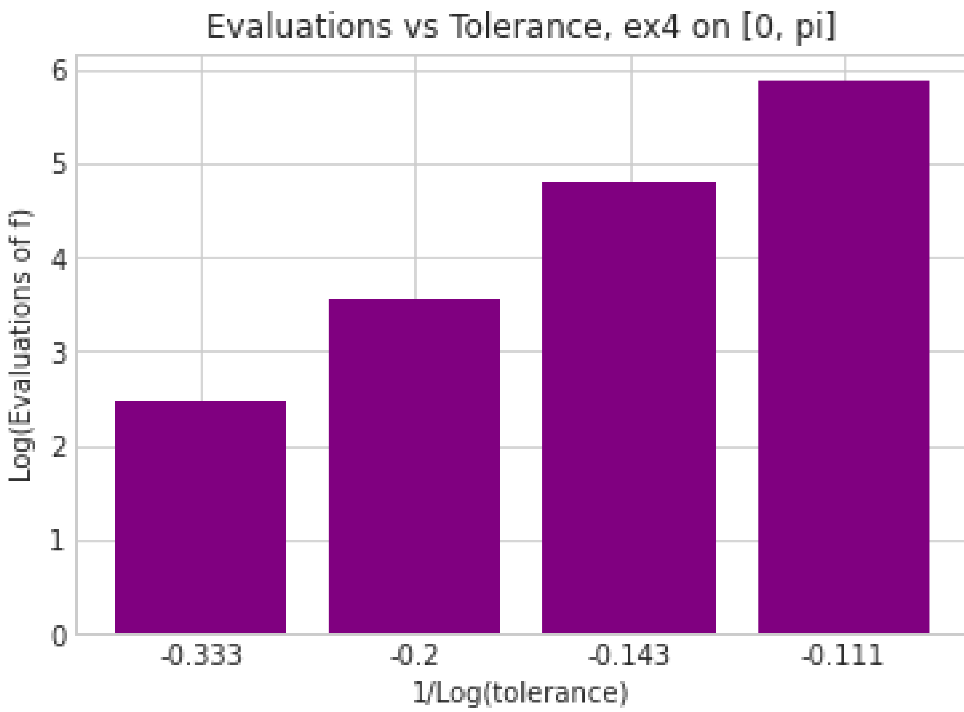
Plot 4: We see that the good interval points form a uniform distribution. Each interval ends up as the same distance, reaching its maxDepth call.

4 SECTION 4 $sin^3(x)$

[ ]:
```
v1, er1, d1, i1 =  approximate(lambda x: math.sin(x) ** 3, 0, math.pi, t1)
v2, er2, d2, i2 =  approximate(lambda x: math.sin(x) ** 3, 0, math.pi, t2)
v3, er3, d3, i3 =  approximate(lambda x: math.sin(x) ** 3, 0, math.pi, t3)
v4, er4, d4, i4 =  approximate(lambda x: math.sin(x) ** 3, 0, math.pi, t4)
eiv1 = v1 + er1
eiv2 = v2 + er2
eiv3 = v3 + er3
eiv4 = v4 + er4
aee1 = eiv1 - v4
aee2 = eiv2 - v4
```

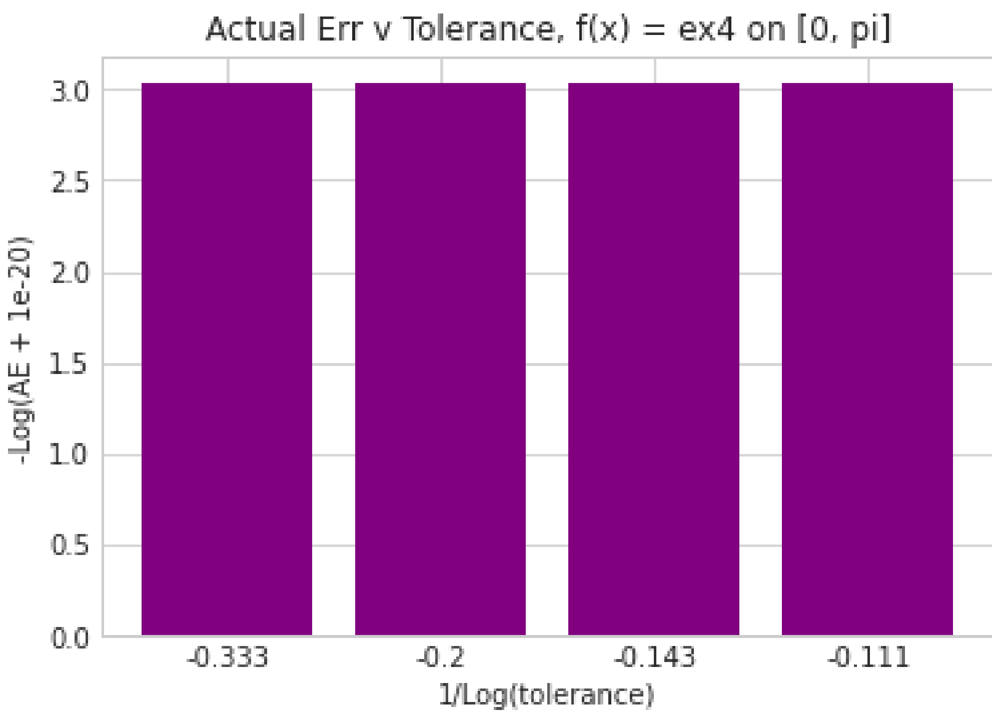
```
aee3 = eiv3 - v4
aee4 = eiv4 - v4
```

[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Evaluations vs Tolerance, ex4 on [0, pi]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("Log(Evaluations of f)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
 ↪list(map(lambda y: math.log(y, 10), evals)), color = 'purple')
plt.show()
```

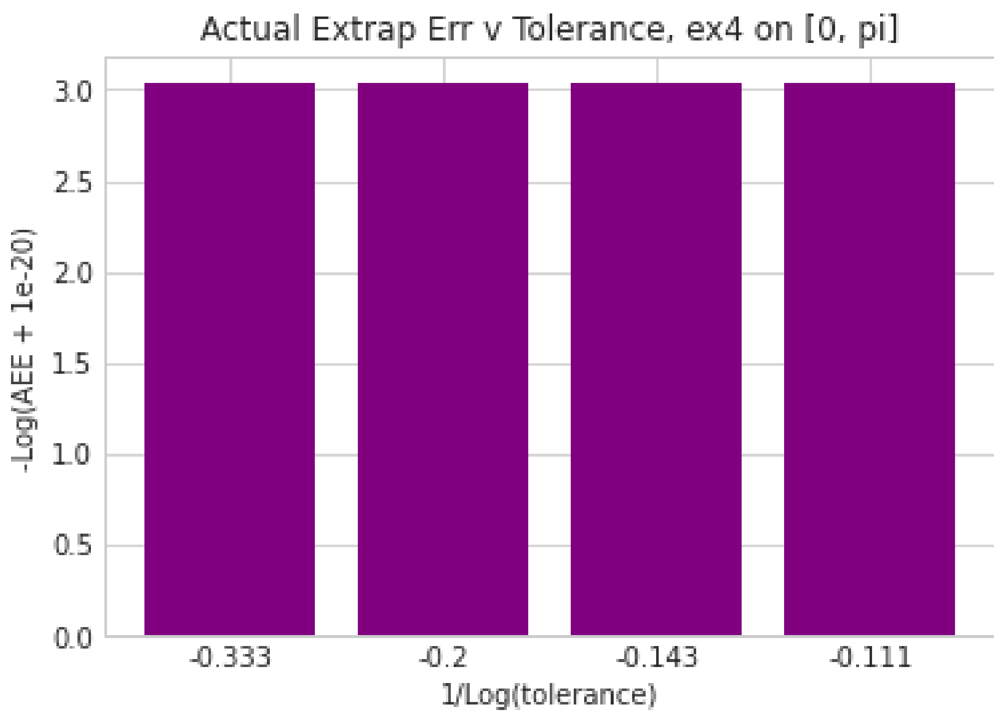


[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Actual Err v Tolerance, f(x) = ex4 on [0, pi]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AE + 1e-20)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
```

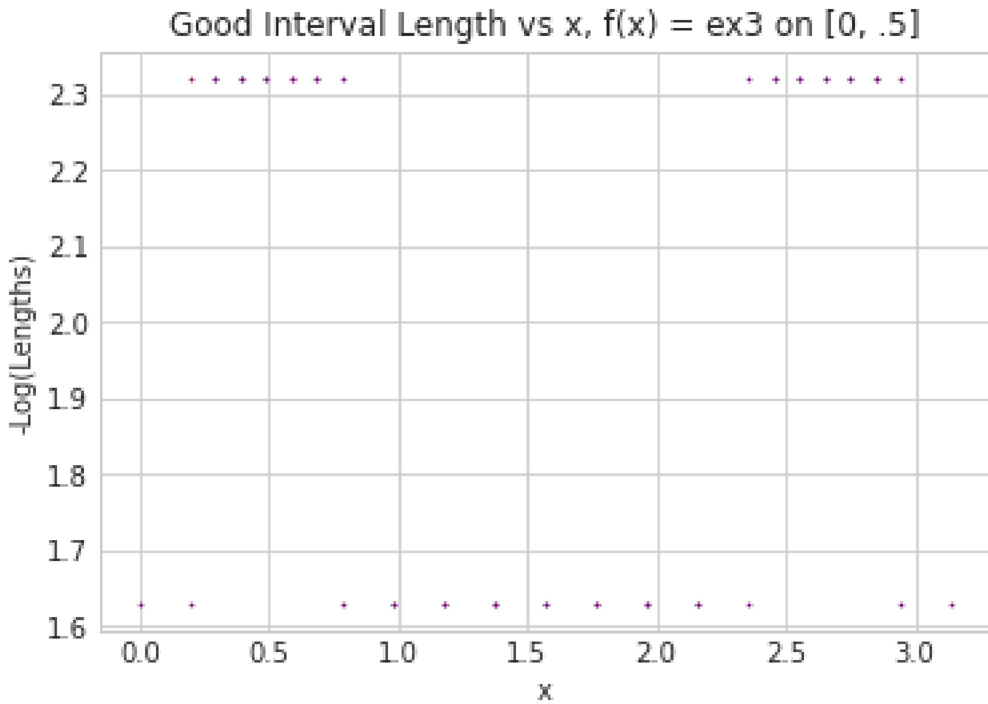
```
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), [er4] * 4)),␣
↪color = 'purple')
plt.show()
```




```
[ ]: fig = plt.figure()
ax = plt.axes()

plt.title("Actual Extrap Err v Tolerance, ex4 on [0, pi]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AEE + 1e-20)")
tols = [t1, t2, t3, t4]
aees = [aee1, aee2, aee3, aee4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), aees)), color =␣
↪'purple')
plt.show()
```




```
[ ]: fig = plt.figure()
     ax = plt.axes()

     points = []
     lengths = []
     v1, er1, d1, i1 = approximate(lambda x: math.sin(x) ** 3, 0, math.pi, t1)

     plt.title("Good Interval Length vs x, f(x) = ex3 on [0, .5]")
     plt.xlabel("x")
     plt.ylabel("-Log(Lengths)")
     ax.scatter(points, list(map(lambda y: -1 * math.log(y), lengths)), s = .5,␣
      ↪color = "purple")
     plt.show()
```



Plot 1: As tolerance increases, so too does the number of evaluations.

Plot 2: Actual error is a constant, same as the actual extrapolated error. -Log Actual error is less than prior examples at around 3, meaning it was harder to estimate this function.

Plot 3: The AEEs are actually different depending on the tolerance for this function, but the differences are so slight it appears the approximate extrapolated error is the same for every tolerance.

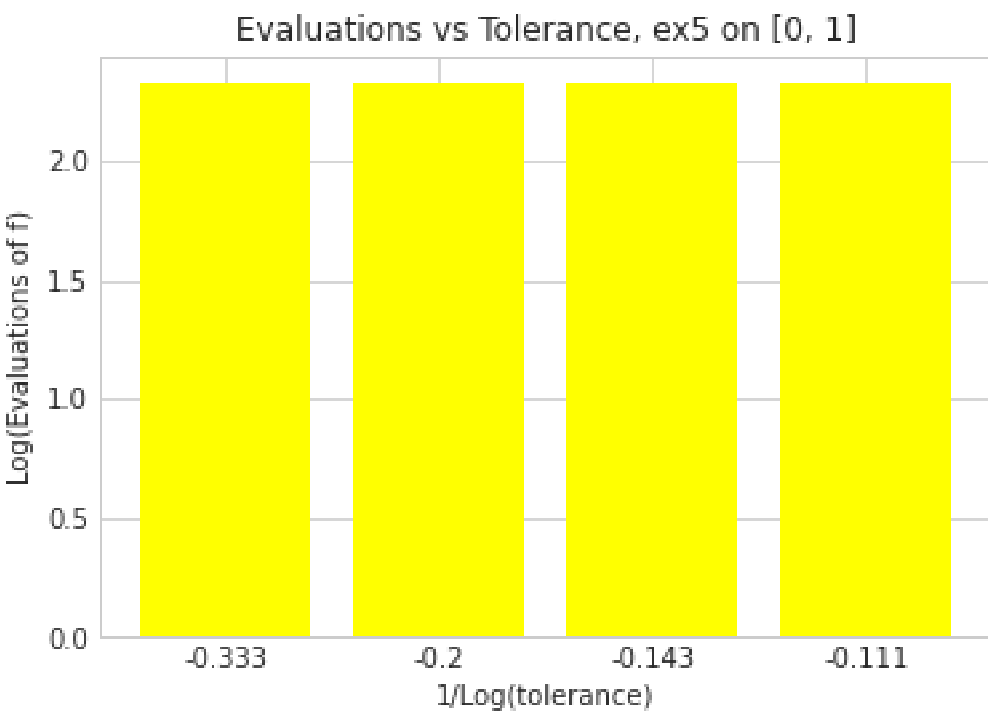
Plot 4: We see that the good interval points form two strata. Near zero and pi/2, we see easy estimations (large intervals) and near pi/4 and 3pi/4 we see small intervals (harder estimation).

5 EX 5 piecewise

[ ]:
```
def f (x):
    if 2 * (x ** 2) < 1:
        return 1
    else:
        return 0

v1, er1, d1, i1 =  approximate(f, 0, 1, t1)
v2, er2, d2, i2 =  approximate(f, 0, 1, t2)
v3, er3, d3, i3 =  approximate(f, 0, 1, t3)
v4, er4, d4, i4 =  approximate(f, 0, 1, t4)
eiv1 = v1 + er1
```

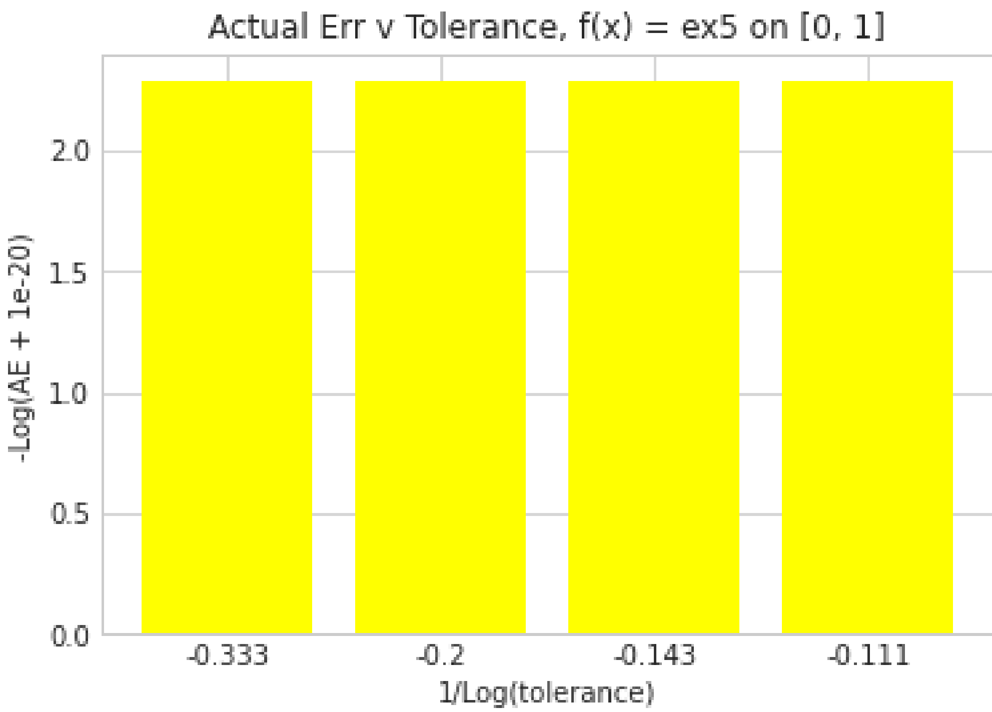
```
eiv2 = v2 + er2
eiv3 = v3 + er3
eiv4 = v4 + er4
aee1 = eiv1 - v4
aee2 = eiv2 - v4
aee3 = eiv3 - v4
aee4 = eiv4 - v4
```

[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Evaluations vs Tolerance, ex5 on [0, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("Log(Evaluations of f)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),
 ↪list(map(lambda y: math.log(y, 10), evals)), color = 'yellow')
plt.show()
```

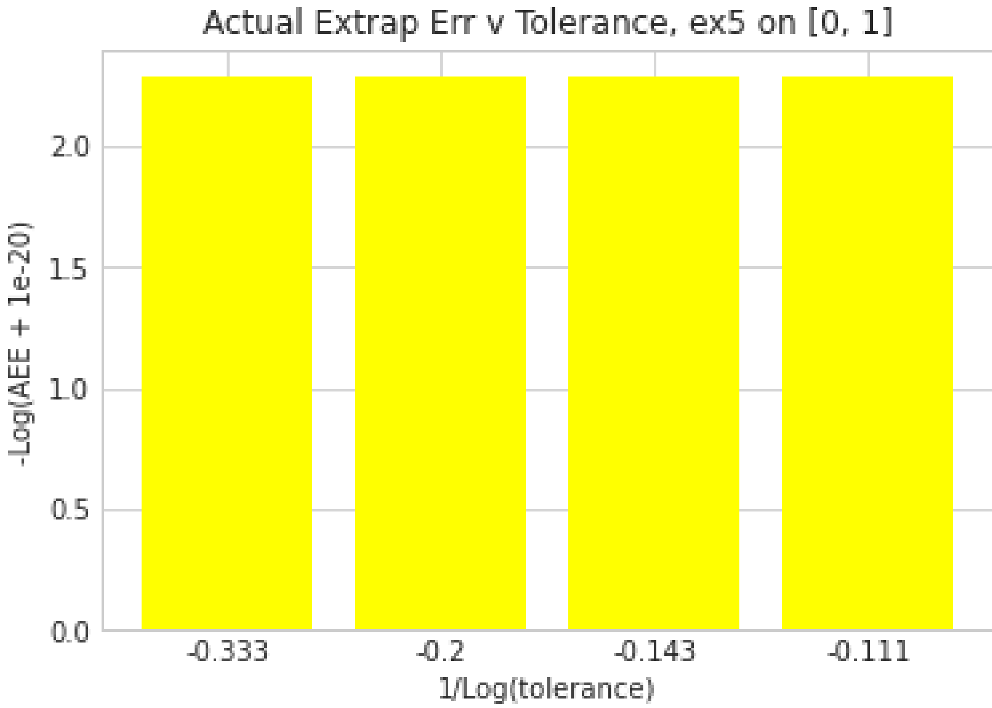


[ ]:
```
fig = plt.figure()
ax = plt.axes()
```

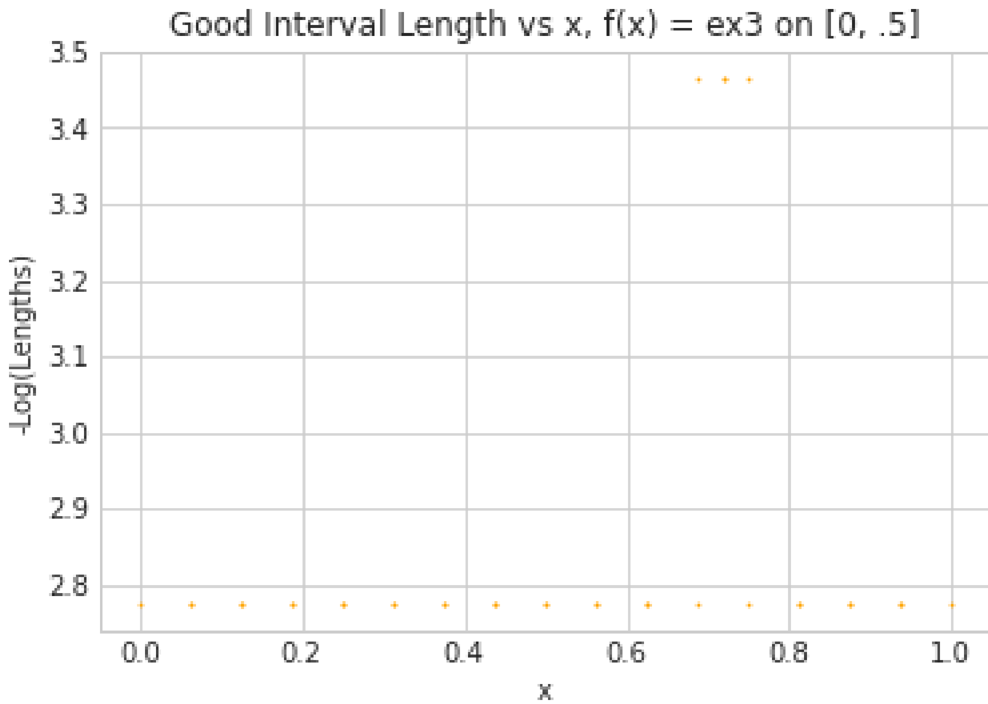
```
plt.title("Actual Err v Tolerance, f(x) = ex5 on [0, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AE + 1e-20)")
tols = [t1, t2, t3, t4]
evals = [i1, i2, i3, i4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
 ↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), [er4] * 4)),␣
 ↪color = 'yellow')
plt.show()
```



[ ]:
```
fig = plt.figure()
ax = plt.axes()

plt.title("Actual Extrap Err v Tolerance, ex5 on [0, 1]")
plt.xlabel("1/Log(tolerance)")
plt.ylabel("-Log(AEE + 1e-20)")
tols = [t1, t2, t3, t4]
aees = [aee1, aee2, aee3, aee4]
ax.bar(list(map(lambda x: str(round(1 / math.log(x, 10), 3)), tols)),␣
 ↪list(map(lambda y: -1 * math.log(abs(y) + (10 ** -20), 10), aees)), color =␣
 ↪'yellow')
plt.show()
```



[ ]:
```
fig = plt.figure()
ax = plt.axes()

points = []
lengths = []
v1, er1, d1, i1 = approximate(f, 0, 1, t1)

plt.title("Good Interval Length vs x, f(x) = ex3 on [0, .5]")
plt.xlabel("x")
plt.ylabel("-Log(Lengths)")
ax.scatter(points, list(map(lambda y: -1 * math.log(y), lengths)), s = .5,␣
 ↪color = "orange")
plt.show()
```



Plot 1: As tolerance increases, evaluations stays the same.

Plot 2: Actual error is a constant, same as the actual extrapolated error. -Log Actual error is less than prior examples at around 3, meaning it was harder to estimate this function.

Plot 3: The AEEs are identical, as each estimation is maxed in depth.

Plot 4: We see that the good interval points form a uniform distribution. This is because each estimation goes its full depth.