MC Estimations 1: Area calculation

1-

$$F(x)=3x^3-5x^2+1 \ rac{xmin=-1}{xmax=2}$$

Double-click (or enter) to edit

```
import numpy as np
 2 from scipy.integrate import quad
    from pylab import *
4
5
    1=[]
    def func1(x):
        f=1-(5*x**2)+(3*x**2)
10
        v=0
11
        if f>=0:
12
          1.append(x)
13
          return f
14
15
        else.
```

```
16
          return v
17
18
    def function(x):
        f=1-(5*x**2)+(3*x**2)
19
        v=0
20
        if f>=0:
21
22
23
           return f
24
        else:
           return (v*x)
25
26
27
    def mc integrate(func, a, b, n = 1000):
        # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
        vals = np.random.uniform(a, b, n)
30
        y = [abs(func(val)) for val in vals]
31
32
        y mean = np.sum(y)/n
33
        integ = (b-a) * y mean
34
35
        return integ
36
    print(f"Monte Carlo solution: {mc integrate(func1, -1, 2, 1000): .4f}")
37
    res, =quad(function, -1, 2)
38
    print(f"Correct solution: {res: .4f}")
39
  Monte Carlo solution: 0.9945
   Correct solution: 0.9428
1 mx=math.floor(max(1))
 2 mn=math.floor(min(1))
```

CIJC.

```
1 print(f'Calculation error: {np.sqrt((mx-mn)/12)/np.sqrt(1000)}')
```

Calculation error: 0.009128709291752768

```
1 import numpy as np
2 from scipy.integrate import quad
 3 from pylab import *
6 1=[]
8 def func1(x):
9
      f=1-(5*x**2)+(3*x**2)
10
     v=0
11
      if f>=0:
12
        1.append(x)
13
        return f
14
15
      else:
16
        return v
17
18 def function(x):
      f=1-(5*x**2)+(3*x**2)
19
      v=0
20
      if f>=0:
21
22
23
          return f
24
      else:
```

```
return (v*x)
25
26
27 def mc integrate(func, a, b, n = 10000):
      # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
30
      vals = np.random.uniform(a, b, n)
      y = [abs(func(val)) for val in vals]
31
32
33
      y mean = np.sum(y)/n
      integ = (b-a) * y mean
34
35
      return integ
36
37 print(f"Monte Carlo solution: {mc integrate(func1, -1, 2, 10000): .4f}")
38 res, =quad(function, -1, 2)
39 print(f"Correct solution: {res: .4f}")
   Monte Carlo solution: 0.9492
   Correct solution: 0.9428
1 mx=math.floor(max(1))
 2 mn=math.floor(min(1))
1 print(f'Calculation error: {np.sqrt((mx-mn)/12)/np.sqrt(10000)}')
   Calculation error: 0.0028867513459481286
```

2-

$$F(x) = e^{-x}$$
 $xmin = -0.5$
 $xmax = 1$

```
1 import numpy as np
 2 from scipy.integrate import quad
 3 from pylab import *
 5
 6 1=[]
 8 def func1(x):
10
      f = e^{**}(-x)
11
   V=0
   if f>=0:
12
     1.append(x)
13
14
   return f
15
      else:
16
        return v
17
18 def function(x):
      f = e^{**}(-x)
19
20
   v=0
      if f>=0:
21
22
23
         return f
24
      else:
         return (v*x)
25
```

```
26
27 def mc integrate(func, a, b, n = 1000):
      # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
30
      vals = np.random.uniform(a, b, n)
      y = [abs(func(val)) for val in vals]
31
32
33
      y mean = np.sum(y)/n
      integ = (b-a) * y mean
34
35
      return integ
36
37 print(f"Monte Carlo solution: {mc integrate(func1, -0.5, 1, 1000): .4f}")
38 res, =quad(function, -0.5,1)
39 print(f"Correct solution: {res: .4f}")
   Monte Carlo solution: 1.3110
   Correct solution: 1.2808
1 mx=(max(1))
2 \text{ mn}=(\min(1))
1 print(f'Calculation error: {np.sqrt((mx-mn)/12)/np.sqrt(1000)}')
   Calculation error: 0.011174739552668402
```

→ 10000 Samples

1 import numpy as np

```
2 from scipy.integrate import quad
 3 from pylab import *
 5
6 1=[]
8 def func1(x):
9
10
      f = e^{**}(-x)
11
      v=0
12
      if f>=0:
        1.append(x)
13
        return f
14
      else:
15
         return v
16
17
18 def function(x):
      f = e^{**}(-x)
19
20
      v=0
      if f>=0:
21
22
          return f
23
      else:
24
25
          return (v*x)
26
27 def mc integrate(func, a, b, n = 10000):
      # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
      vals = np.random.uniform(a, b, n)
30
      y = [abs(func(val)) for val in vals]
31
32
```

```
33
     y mean = np.sum(y)/n
      integ = (b-a) * y_mean
34
       return integ
35
36
37 print(f"Monte Carlo solution: {mc integrate(func1, -0.5, 1, 10000): .4f}")
38 res, =quad(function, -0.5,1)
39 print(f"Correct solution: {res: .4f}")
   Monte Carlo solution: 1.2807
   Correct solution: 1.2808
 1 \text{ mx} = (\text{max}(1))
 2 mn=(min(1))
 1 print(f'Calculation error: {np.sqrt((mx-mn)/12)/np.sqrt(10000)}')
   Calculation error: 0.003535084187420184
```

$$F(x) = sin(3x) \ _{\substack{xmin = -\pi \ xmax = 0}}$$

Double-click (or enter) to edit

```
1 import numpy as np
2 from scipy.integrate import quad
```

```
3 from pylab import *
4
 5
6 1=[]
7
8 def func1(x):
9
      f=math.sin(3*x)
10
11
      v=0
      if f>=0:
12
        1.append(x)
13
        return f
14
15
      else:
16
        return v
17
18 def function(x):
      f=math.sin(3*x)
19
      v=0
20
      if f>=0:
21
22
23
          return f
24
      else:
          return (v*x)
25
26
27 def mc integrate(func, a, b, n = 1000):
      # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
      vals = np.random.uniform(a, b, n)
30
      y = [abs(func(val)) for val in vals]
31
32
33
      y mean = np.sum(y)/n
```

```
integ = (b-a) * y mean
34
       return integ
35
36
37 print(f"Monte Carlo solution: {mc integrate(func1, -math.pi, 0, 1000): .4f}")
38 res, =quad(function, -math.pi, 0)
39 print(f"Correct solution: {res: .4f}")
   Monte Carlo solution: 0.6808
   Correct solution: 0.6667
 1 mx=(max(1))
 2 \text{ mn}=(\min(1))
    print(f'Calculation error: {np.sqrt((mx-mn)/12)/np.sqrt(1000)}')
   Calculation error: 0.009325006456078368
Double-click (or enter) to edit
```

```
1 import numpy as np
2 from scipy.integrate import quad
3 from pylab import *
4
5
6 l=[]
7
8 def func1(x):
```

```
10
      f=math.sin(3*x)
11
      v=0
      if f>=0:
12
        1.append(x)
13
        return f
14
15
      else:
16
        return v
17
18 def function(x):
      f=math.sin(3*x)
19
      v=0
20
      if f>=0:
21
22
          return f
23
24
      else:
          return (v*x)
25
26
27 def mc integrate(func, a, b, n = 10000):
      # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
      vals = np.random.uniform(a, b, n)
30
      y = [abs(func(val)) for val in vals]
31
32
33
      y mean = np.sum(y)/n
      integ = (b-a) * y mean
34
35
       return integ
36
37 print(f"Monte Carlo solution: {mc integrate(func1, -math.pi, 0, 10000): .4f}")
38 res, =quad(function, -math.pi, 0)
20 ppint/f"Copport colution: [poc. 4f]")
   Monte Carlo solution: 0.6735
```

```
Correct solution: 0.6667 1 \max(\max(1)) 2 \min(\min(1)) 1 \operatorname{print}(f'\operatorname{Calculation error}: \{\operatorname{np.sqrt}((\operatorname{mx-mn})/12)/\operatorname{np.sqrt}(10000)\}') \operatorname{Calculation error}: 0.002951649105715342 F(x) = \cos(4x)
```

 $egin{aligned} xmin = -\pi \ xmax = 0 \end{aligned}$

Double-click (or enter) to edit

```
1 import numpy as np
2 from scipy.integrate import quad
3 from pylab import *
4
5
6 l=[]
7
8 def func1(x):
9
10 f=math.cos(4*x)
```

```
11
      v=0
      if f>=0:
12
13
        1.append(x)
        return f
14
      else:
15
16
         return v
17
18 def function(x):
19
      f=math.cos(4*x)
20
      v=0
      if f>=0:
21
22
23
          return f
24
      else:
          return (v*x)
25
26
27 def mc integrate(func, a, b, n = 1000):
      # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
      vals = np.random.uniform(a, b, n)
30
      y = [abs(func(val)) for val in vals]
31
32
33
      y mean = np.sum(y)/n
      integ = (b-a) * y mean
34
      return integ
35
36
37 print(f"Monte Carlo solution: {mc integrate(func1, -math.pi, 0, 1000): .4f}")
38 res, =quad(function, -math.pi, 0)
39 print(f"Correct solution: {res: .4f}")
   Monte Carlo solution: 1.0190
   Correct solution: 1.0000
```

```
1 import numpy as np
 2 from scipy.integrate import quad
 3 from pylab import *
6 1=[]
8 def func1(x):
 9
10
      f=math.cos(4*x)
    v=0
11
12
      if f>=0:
13
        1.append(x)
        return f
14
15
      else:
16
        return v
17
18 def function(x):
```

```
f=math.cos(4*x)
19
      v=0
20
      if f>=0:
21
22
23
          return f
24
       else:
          return (v*x)
25
26
27 def mc integrate(func, a, b, n = 10000):
      # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
      vals = np.random.uniform(a, b, n)
30
      y = [abs(func(val)) for val in vals]
31
32
33
      y mean = np.sum(y)/n
       integ = (b-a) * y mean
34
       return integ
35
36
37 print(f"Monte Carlo solution: {mc integrate(func1, -math.pi, 0, 10000): .4f}")
38 res, =quad(function, -math.pi, 0)
39 print(f"Correct solution: {res: .4f}")
   Monte Carlo solution: 0.9959
   Correct solution: 1.0000
1 \text{ mx} = (\text{max}(1))
2 mn = (min(1))
1 print(f'Calculation error: {np.sqrt((mx-mn)/12)/np.sqrt(10000)}')
```

Calculation error: 0.005116468044522063

$$F(x)=sin(e^{x/2})$$
 $x=0$ $x=\pi$

```
1
 1 import numpy as np
 2 from scipy.integrate import quad
 3 from pylab import *
 6 1=[]
 8 def func1(x):
 9
      f=math.sin(e^{**}(x/2))
10
     v=0
11
      if f>=0:
12
        1.append(x)
13
14
       return f
15
      else:
16
        return v
17
18 def function(x):
```

```
f=math.sin(e^{**}(x/2))
19
      v=0
20
      if f>=0:
21
22
23
          return f
24
       else:
          return (v*x)
25
26
27 def mc integrate(func, a, b, n = 1000):
       # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
      vals = np.random.uniform(a, b, n)
30
       y = [abs(func(val)) for val in vals]
31
32
33
      y mean = np.sum(y)/n
       integ = (b-a) * y mean
34
       return integ
35
36
37 print(f"Monte Carlo solution: {mc integrate(func1, 0,math.pi, 1000): .4f}")
38 res, =quad(function, 0, math.pi)
39 print(f"Correct solution: {res: .4f}")
   Monte Carlo solution: 1.8693
   Correct solution: 1.8117
 1 \text{ mx} = (\text{max}(1))
 2 mn = (min(1))
 1 print(f'Calculation error: {np.sqrt((mx-mn)/12)/np.sqrt(1000)}')
```

Calculation error: 0.013797480057624496

```
1
```

Double-click (or enter) to edit

1

```
1 import numpy as np
 2 from scipy.integrate import quad
 3 from pylab import *
 4
 5
 6 1=[]
 8 def func1(x):
      f=math.sin(e^{**}(x/2))
10
      v=0
11
      if f>=0:
12
         1.append(x)
13
        return f
14
15
      else:
16
         return v
17
18 def function(x):
      f=math.sin(e^{**}(x/2))
19
      v=0
20
      if f>=0:
21
```

```
22
          return f
23
24
       else:
25
          return (v*x)
26
27 def mc integrate(func, a, b, n = 10000):
       # Monte Carlo integration between x1 and x2 of given function from a to b
28
29
30
      vals = np.random.uniform(a, b, n)
       y = [abs(func(val)) for val in vals]
31
32
      y mean = np.sum(y)/n
33
34
       integ = (b-a) * y mean
35
       return integ
36
37 print(f"Monte Carlo solution: {mc integrate(func1, 0, math.pi, 10000): .4f}")
38 res, =quad(function, 0,math.pi)
39 print(f"Correct solution: {res: .4f}")
   Monte Carlo solution: 1.8117
   Correct solution: 1.8117
 1 \text{ mx} = (\text{max}(1))
 2 mn = (min(1))
 1 print(f'Calculation error: {np.sqrt((mx-mn)/12)/100}')
   Calculation error: 0.004366816008795486
```

