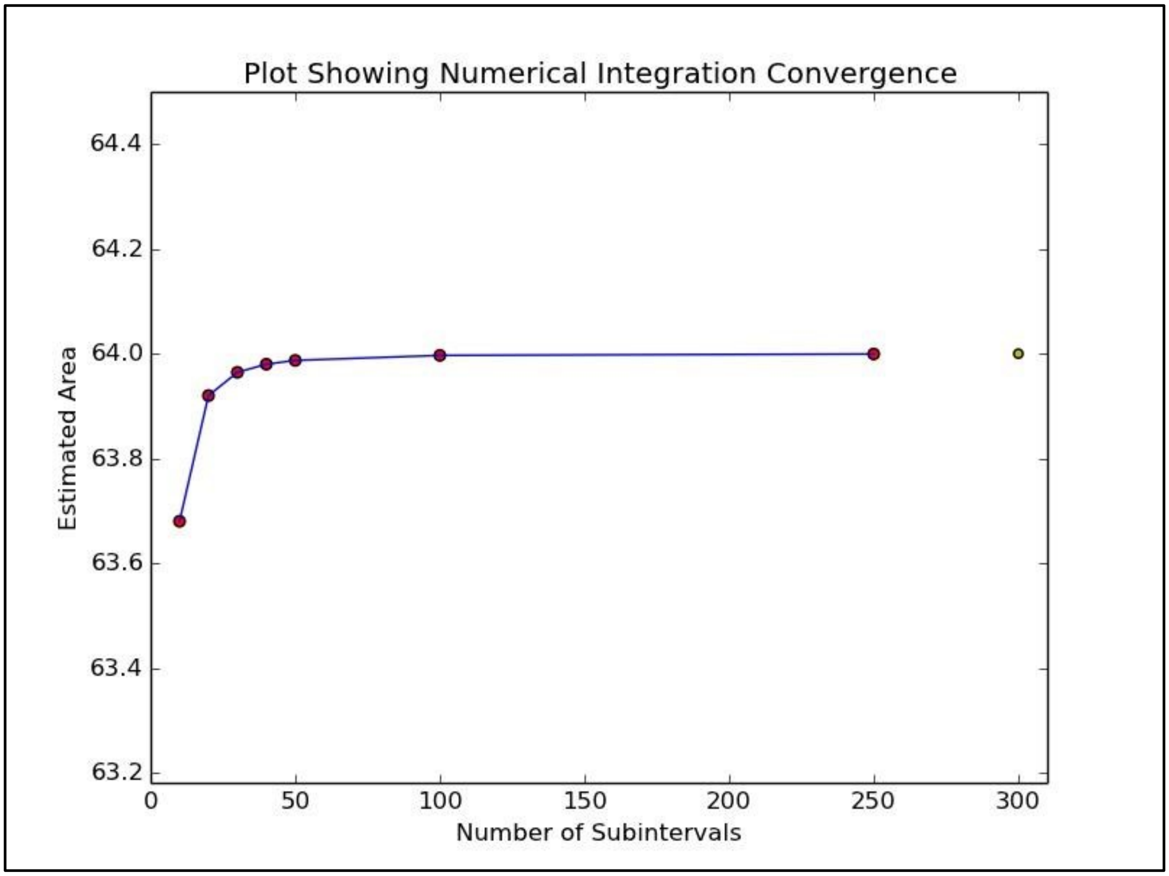
MSPA 400 Session 8 Python Solutions

Module 1

Exercise: Instead of using the trapezoidal rule for integration, substitute midpoint rule in the function integrate() and run the rest of the code without modification. Note the difference in how convergence occurs. Compare to the answer sheet.

The modifications to the integrate function are shown below with the plot.

```
def integrate(a,b,n):
    sum = 0.0
    delta = (b-a)/n
    i = 0
    while i < n:
        sum = sum + delta*(f(a+delta*(i+0.5)))
        i = i+1
    return sum
```



Final Estimate of Area with 250 subdivisions = 63.999

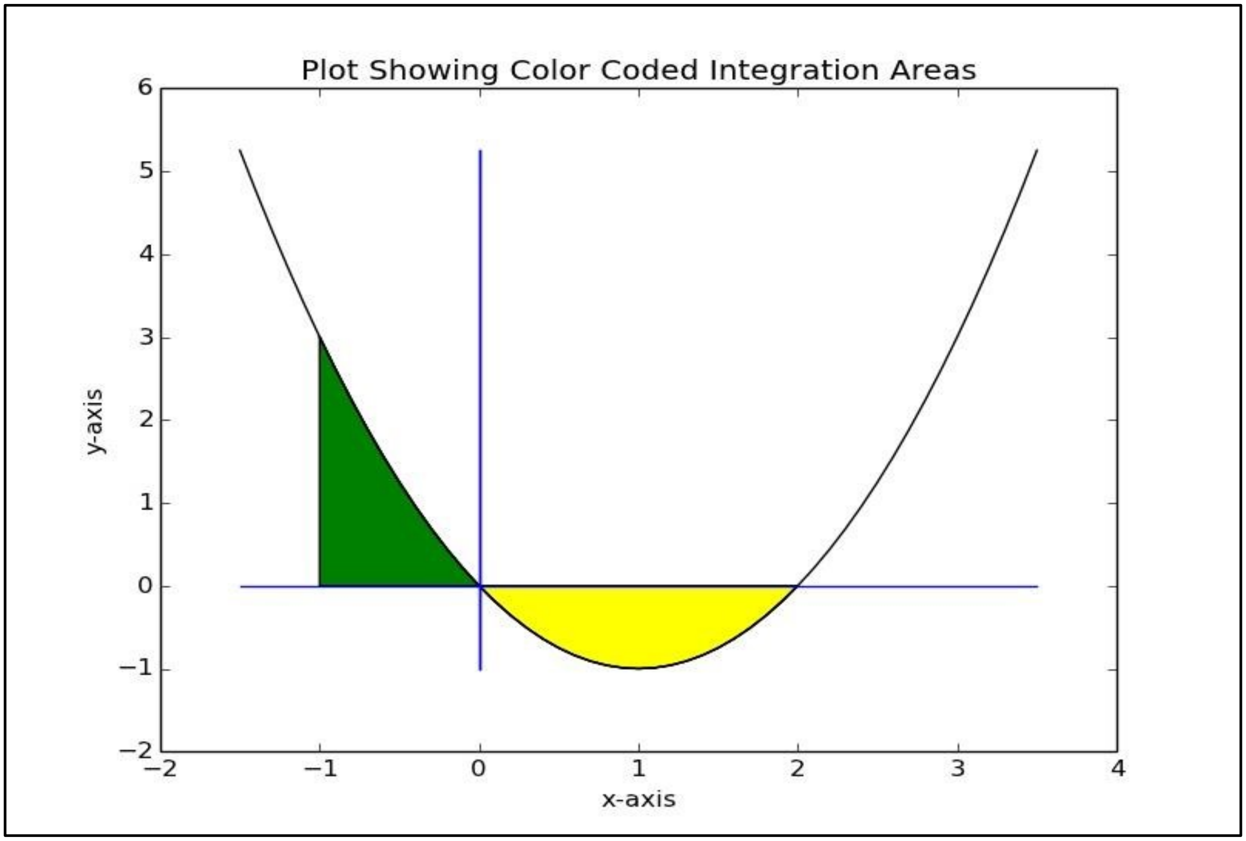
MSPA 400 Session 8 Python Solutions

Module 2

Exercise: Refer to Lial Section 15.4 Exercise 42. Modify the code to reproduce the plot shown in the exercise. Compare to the answer sheet.

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
   f = x*x-2.0*x
   return f

def integrate(a,b,n):
   sum = 0.0
   delta = (b-a)/n
   i = 0
   while i < n:
      sum = sum +
delta*(f(a+delta*(i+1))+f(a+del
ta*i))/2
      i = i+1
   return sum

c = 2.0
b = 0.0
a = -1.0
n=100

area1 = integrate(a,b,n)
area2 = integrate(b,c,n)
area =
np.abs(area1)+np.abs(area2)
print ('Final Estimate of Area=
%r' %area)

plt.figure()
shaded_x = np.arange(-
1.0,2.1,0.1)
y = f(shaded_x)
plt.plot(shaded_x,y,c='k')
ymin=min(y)-0.5
ymax=max(y)+0.5
```

```
plt.fill_between(shaded_x,0.0,y,
where= y < 0.0, facecolor =
'y',interpolate=True)
plt.fill_between(shaded_x,0.0,y,
where= y > 0.0, facecolor = 'g',
interpolate=True)
plt.xlim(-2.0,4.0)
plt.ylim(ymin-0.5,ymax+2.5)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Plot Showing Color
Coded Integration Areas')

black_x=np.arange(-1.5,3.6,.1)
y=f(black_x)
z=0.0*black_x
u=0.0*y
plt.plot(black_x,y,c='k')
plt.plot(black_x,z,u,y,c='b')
plt.show()
```



Final Estimate of Area= 2.66655