

```
1: """
2: The Magnetic Field from Current Distribution
3: """
4:
5: from math import *
6: from pylab import show, imshow
7: import numpy as np
8:
9:
10: def simpson(func, a, b, h=0.001):
11:     """Approximates integral using simpson method"""
12:     n = int(abs(b - a) / h)
13:     n -= 1 if n % 2 == 1 else 0
14:     return (h / 3.0) * (
15:         func(a) + func(b) +
16:         (4.0 * sum([func(a + (k * h)) for k in range(1, n, 2)])) +
17:         (2.0 * sum([func(a + (k * h)) for k in range(2, n - 1, 2)])))
18:
19: def adaptive(func, a, b, h=0.001, delta=10e-4):
20:     """Approximates integral using adaptive method"""
21:     n = int(abs(b - a) / h)
22:     i0 = h * ((0.5 * (func(a) + func(b))) + sum(
23:         [func(a + k * h) for k in range(1, int((b - a) / h)]))
24:     epsilon = delta + 10
25:     while epsilon > delta:
26:         h /= 2
27:         n *= 2
28:         i1 = (0.5 * i0) + (h * sum([func(a + k * h) for k in range(1, n, 2)]))
29:         epsilon = abs(i1 - i0) / 3
30:         i0 = i1
31:     return i1
32:
33:
34: def B(x, y, z):
35:     func_x = lambda theta: (2 * z * sin(theta)) / pow((x - 3 * cos(theta))**2 + (y - 2
36: * sin(theta))**2 + z**2, 3 / 2)
37:     func_y = lambda theta: -(3 * z * cos(theta)) / pow((x - 3 * cos(theta))**2 + (y - 2
38: * sin(theta))**2 + z**2, 3 / 2)
39:     func_z = lambda theta: (3 * y * cos(theta) - 2 * x * sin(theta)) / pow((x - 3 * cos
40: (theta))**2 + (y - 2 * sin(theta))**2 + z**2, 3 / 2)
41:     comp_x = simpson(func_x, 0, 2 * pi, 0.1)
42:     comp_y = simpson(func_y, 0, 2 * pi, 0.1)
43:     comp_z = simpson(func_z, 0, 2 * pi, 0.1)
44:     return (comp_x, comp_y, comp_z)
45:
46: def B_adaptive(x, y, z):
47:     func_x = lambda theta: (2 * z * sin(theta)) / pow((x - 3 * cos(theta))**2 + (y - 2
48: * sin(theta))**2 + z**2, 3 / 2)
49:     func_y = lambda theta: -(3 * z * cos(theta)) / pow((x - 3 * cos(theta))**2 + (y - 2
50: * sin(theta))**2 + z**2, 3 / 2)
51:     func_z = lambda theta: (3 * y * cos(theta) - 2 * x * sin(theta)) / pow((x - 3 * cos
52: (theta))**2 + (y - 2 * sin(theta))**2 + z**2, 3 / 2)
53:     comp_x = adaptive(func_x, 0, 2 * pi, 0.1, 10e-4)
54:     comp_y = adaptive(func_y, 0, 2 * pi, 0.1, 10e-4)
55:     comp_z = adaptive(func_z, 0, 2 * pi, 0.1, 10e-4)
56:     return (comp_x, comp_y, comp_z)
57:
58:
59: def main():
60:
61:     def a():
62:         print(B(1, 4, 7))
63:
64: if __name__ == '__main__':
65:     a()
```

```
58: def b():
59:     print(B_adaptive(1,2,5))
60: def c():
61:     data = []
62:     for y in np.arange(-5, 5, 0.1):
63:         row = []
64:         for x in np.arange(-5, 5, 0.1):
65:             b = B_adaptive(x, y, 1)
66:             row.append(np.sqrt(b[0]**2+b[1]**2+b[2]**2))
67:         data.append(row)
68:     imshow(data)
69:     show()
70:
71: a()
72: b()
73: c()
74:
75:
76: if __name__ == "__main__":
77:     main()
```