

```
1: #!/usr/bin/python3
2:
3: from pylab import *
4: from math import *
5: from numpy import *
6:
7: def simpson(func, a, b, n=1000):
8:     """Approximates integral using simpson method"""
9:     n -= 1 if n % 2 == 1 else 0
10:    h = abs(b - a) / n
11:    return (h / 3.0) * (
12:        func(a) + func(b) +
13:        (4.0 * sum([func(a + (k * h)) for k in range(1, n, 2)])) +
14:        (2.0 * sum([func(a + (k * h)) for k in range(2, n - 1, 2)])))
15:
16:
17: def p1():
18:     """5.4: The diffraction limit of a telescope"""
19:
20:     def J(m, x, n=1000):
21:         return (1 / pi) * simpson(lambda theta: cos(m * theta - x * sin(theta)),
22:                                   0, pi, n)
23:
24:     def a():
25:         plot(linspace(0, 20), [J(0, x) for x in linspace(0, 20)])
26:         plot(linspace(0, 20), [J(1, x) for x in linspace(0, 20)])
27:         plot(linspace(0, 20), [J(2, x) for x in linspace(0, 20)])
28:         show()
29:
30:     def b():
31:         print("This takes a little while to run...")
32:         max_r = 1e-6
33:         resolution = 50
34:         scale = max_r / resolution
35:         k = 2 * pi / (5e-7)
36:         I = lambda r: pow(J(1, k * r, 100) / (k * r), 2) if r != 0 else pow(J(1, k * 1e
-9, 100) / (k * 1e-9), 2)
37:         D = lambda x, y: sqrt(x**2 + y**2)
38:         data = [[
39:             I(D(x - resolution, y - resolution) * scale)
40:             for x in range(resolution * 2 + 1)
41:         ]
42:             for y in range(resolution * 2 + 1)]
43:         imshow(data, vmax=0.01)
44:         # imshow(data)
45:         show()
46:
47:     a()
48:     b()
49:
50: if __name__ == "__main__":
51:     p1()
```