```
1: #!/usr/bin/python3
 2:
 3: from pylab import *
 4: from numpy import *
 5:
 6:
 7: def simpson(func, a, b, h=0.1):
 8:
        """Approximates integral using simpson method"""
 9:
        n = int(abs(b - a) / h)
10:
        n -= 1 if n % 2 == 1 else 0
11:
        I1 = (h / 3.0) * (
            func(a) + func(b) +
13:
            (4.0 * sum([func(a + (k * h)) for k in range(1, n, 2)])) +
             (2.0 * sum([func(a + (k * h)) for k in range(2, n - 1, 2)])))
14:
15:
        h2 = 2 * h
16:
        n2 = int(abs(b - a) / h2)
17:
        n2 -= 1 if n2 % 2 == 1 else 0
        I2 = (h2 / 3.0) * (
18:
19:
            func(a) + func(b) +
20:
            (4.0 * sum([func(a + (k * h2)) for k in range(1, n2, 2)])) +
             (2.0 * sum([func(a + (k * h2)) for k in range(2, n2 - 1, 2)])))
21:
22:
        return (I1, abs(I1 - I2) / 3)
23:
24:
25: def adaptive(input_func, a, b, h=0.1, delta=10e-6):
        """Approximates integral using adaptive method"""
26:
27:
        if a > b:
28:
            val = adaptive(input_func, b, a, h=h, delta=delta)
29:
            return (-val[0], val[1])
30:
        if a == -inf and b == inf:
31:
            func = lambda x: (input_func(-x/(1-x))+input_func(x/(1-x))) / pow(1-x,2)
            a = 0
32:
33:
            b = 1
34:
        elif a != -inf and b == inf:
35:
            func = lambda x: input_func((x / (1 - x)) + a) / pow(1 - x, 2)
            a = 0
36:
37:
            b = 1
38:
        else:
39:
            func = input_func
40:
        try:
41:
            func(a)
42:
        except ZeroDivisionError:
43:
            a += 0.00000000000001
44:
        try:
45:
            func(b)
46:
        except ZeroDivisionError:
47:
            b = 0.0000000000001
48:
        n = int(abs(b - a) / h)
49:
        i0 = h * ((0.5 * (func(a) + func(b))) + sum(
50:
            [func(a + k * h) for k in range(1, int((b - a) / h))]))
51:
        epsilon = delta + 10
52:
        while epsilon > delta:
53:
            h /= 2
            n *= 2
54:
55:
            i1 = (0.5 * i0) + (h * sum([func(a + k * h) for k in range(1, n, 2)]))
56:
            epsilon = abs(i1 - i0) / 3
57:
            i0 = i1
58:
        return i1, epsilon
59:
60:
61: def mag2(x, xp, y, yp, z, zp):
62:
        return (x - xp)**2 + (y - yp)**2 + (z - zp)**2
63:
```

```
64:
   65: def V(x, y, z, integrate=simpson):
           pre = 1 / (4 * pi)
   66:
   67:
   68:
           def func(theta):
   69:
               xp = 3 * cos(theta)
   70:
               yp = 2 * sin(theta)
   71:
               zp = 0
   72:
               return pre / sqrt(mag2(x, xp, y, yp, z, zp)) * sqrt(
   73:
                   mag2(xp, 0, yp, 0, zp, 0))
   74:
   75:
           return integrate(func, 0, 2 * pi, h=0.1)
   76:
   77:
   78: def E(x, y, z, integrate=simpson):
   79:
           pre = 1 / (4 * pi)
   80:
   81:
           def func(theta):
   82:
               xp = 3 * cos(theta)
   83:
               yp = 2 * sin(theta)
   84:
               zp = 0
   85:
               return (pre * sqrt(mag2(x, xp, y, yp, z, zp))) / pow(
   86:
                   mag2(x, xp, y, yp, z, zp), 3 / 2) * sqrt(mag2(xp, 0, yp, 0, zp, 0))
   87:
           return integrate(func, 0, 2 * pi, h=0.1)
   88:
   89:
   90:
   91: def p2():
   92:
           """The potential and electric field of a linear charge distribution"""
   93:
   94:
           def b():
   95:
               print("Potential:", V(1, 4, 7))
   96:
               print("Electric Field:", E(1, 4, 7))
   97:
  98:
           def c():
  99:
               print("Potential:", V(1, 2, 5, integrate=adaptive))
  100:
               print("ELectric Field:", E(1, 2, 5, integrate=adaptive))
  101:
  102:
           def d():
               res = 100
  103:
  104:
               scale = 3.5 / res
  105:
               data = [V(x * scale, y * scale, 1)[0] for x in range(-res, res + 1)] for y in
range(-res, res + 1)]
               imshow(data)
  106:
  107:
               show()
  108:
  109:
           b()
  110:
           c()
  111:
           d()
  112:
  113:
  114: if __name__ == "__main__":
  115:
           p2()
```