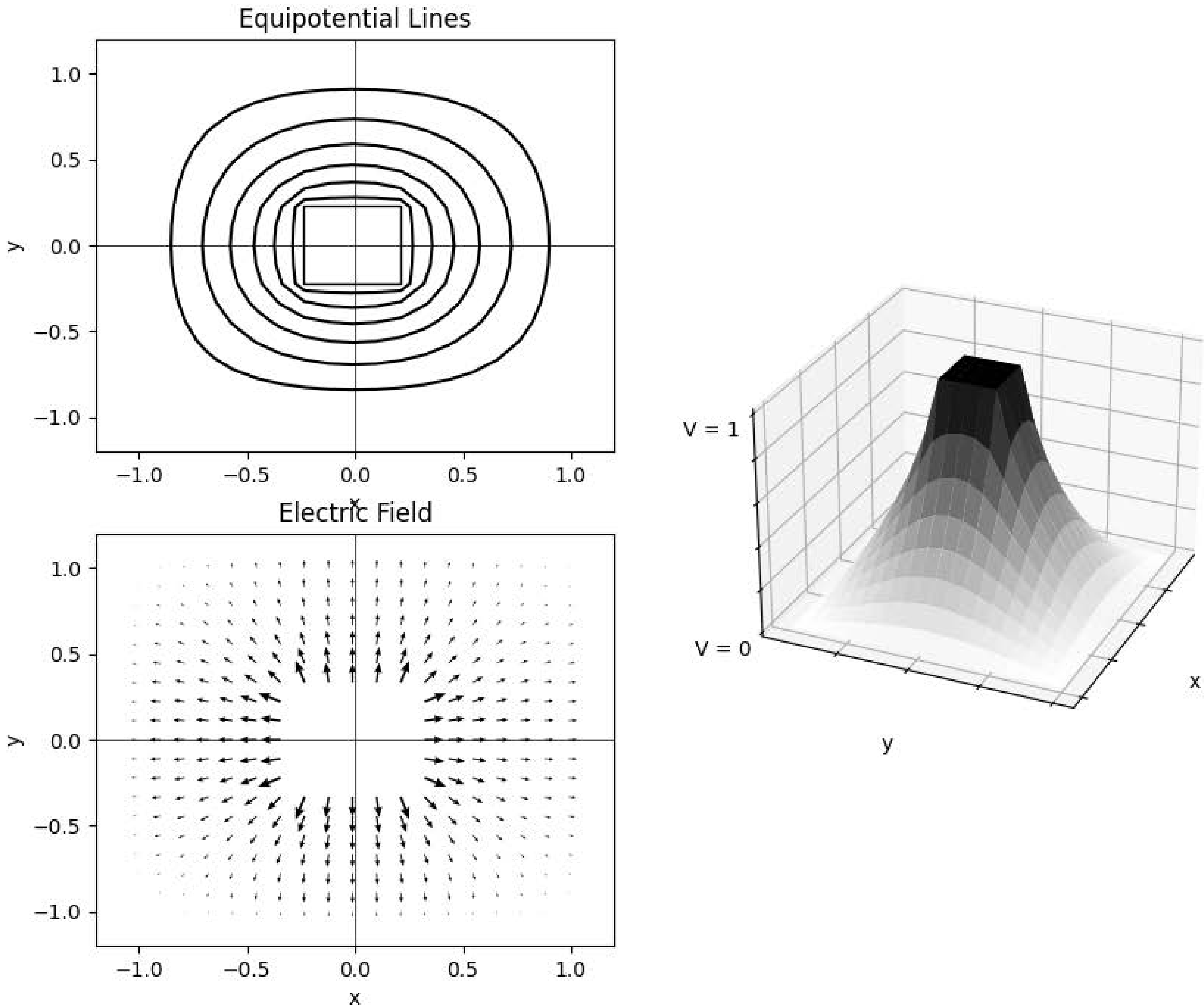

Equipotential Lines
Electric Field
V = 1
V = 0
x
y

```
import matplotlib.pyplot as plt
import copy

import numpy as np
"""
Chapter 5
Potentials and Fields
"""

V = [
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

scalex = 9.1
scaley = 9
scaledim = 9
xlim = [-1.2,1.2]
ylim = [-1.2,1.2]
model = ((-0.24,-0.22), 0.45, 0.45)

"""
V = [
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    ]

scalex = 5
scaley = 5
scaledim = 11
```

```
xlim = [-0.45,0.45]
ylim = [-0.45,0.45]
model = [(-0.24, -0.22), 0.45, 0.45]
"""

fig = plt.figure(figsize=(10, 8))
fig.tight_layout(pad=1.0)


def equi(V):
    ax = fig.add_subplot(221)
    if(xlim is not None and ylim is not None):
        ax.set_xlim(xlim)
        ax.set_ylim(ylim)
    x = np.zeros((len(V), len(V)))
    y = np.zeros((len(V), len(V)))
    for i in range(1, len(V)-1):
        for j in range(1, len(V[i])-1):
            x[i][j] = ((j-scalex)/scaledim)
            y[i][j] = ((i-scaley)/scaledim)
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    ax.contour(x, y, V, zorder=0, colors='k', linestyles='solid')
    ax.axhline(y=0, linewidth=0.5, color='k')
    ax.axvline(x=0, linewidth=0.5, color='k')
    ax.add_patch(plt.Rectangle(
       model[0], model[1], model[2], color='black', fill=False, zorder=2))
    ax.set_title('Equipotential Lines')


def cal_electric_field(V):
    ax = fig.add_subplot(223)
    if(xlim is not None and ylim is not None):
        ax.set_xlim(xlim)
        ax.set_ylim(ylim)
    ax.set_title('Electric Field')
    ax.axhline(y=0, linewidth=0.5, color='k')
    ax.axvline(x=0, linewidth=0.5, color='k')
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    E_x = np.zeros((len(V), len(V)))
    E_y = np.zeros((len(V), len(V)))
    pair = dict()
    for i in range(0, len(V)-1):
        for j in range(0, len(V[i])-1):
            dx = ((j + 1) - (j - 1))
            dy = ((i + 1) - (i - 1))
            if(V[i][j] != 1):
                E_x[i][j] = -(V[i][j+1] - V[i][j-1]) / (2 * dx)
                E_y[i][j] = -(V[i+1][j] - V[i-1][j]) / (2 * dy)
            pair[(j, i)] = (E_x[i][j], E_y[i][j])
    for x, y in pair.keys():
        if(pair[(x, y)][0] == 0 and pair[(x, y)][1] == 0):
            pass
        else:
            ax.quiver((x-scalex)/scaledim, (y-scaley)/scaledim, pair[(x, y)][0], pair[(
                x, y)][1], scale=2.5, minshaft=2, minlength=-2, headwidth=4)
```

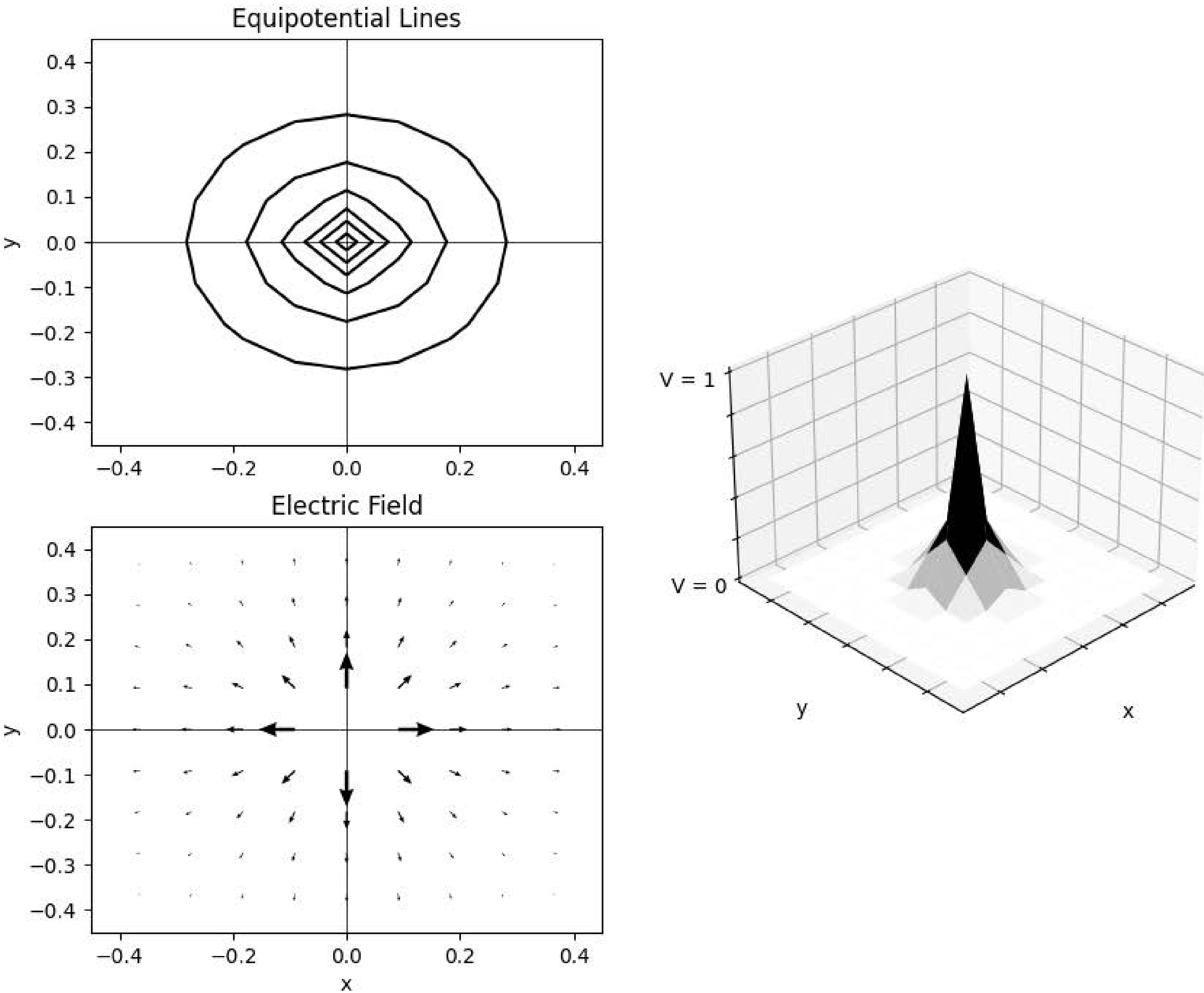
```
def create_3Ddiagram(V):
    ax = fig.add_subplot(122, projection='3d')
    V = np.array(V)
    x = np.zeros((len(V), len(V)))
    y = np.zeros((len(V), len(V)))
    for i in range(0, len(V)):
        for j in range(0, len(V[i])):
            x[i][j] = (j+1)
            y[i][j] = (i+1)
    ax.set_yticklabels([])
    ax.set_xticklabels([])
    ax.set_zticklabels(['V = 0', '', '', '', '', 'V = 1'])
    ax.plot_surface(x, y, V, rstride=1, cstride=1, cmap='binary')
    ax.set_xlabel("x")
    ax.set_ylabel("y")


def plot_graphs(V2):
    create_3Ddiagram(V2)
    cal_electric_field(V2)
    equi(V2)


def main(V, count):
    count[0] += 1
    dV, V2 = updateV(V)
    if(dV <= 0.1):
        create_3Ddiagram(V2)
        cal_electric_field(V2)
        equi(V2)
    else:
        main(V2, count)


def updateV(V):
    V2 = copy.deepcopy(V)
    dV = 0
    for i in range(1, len(V)-1):
        for j in range(1, len(V[i])-1):
            if(V[i][j] == 1 or V[i][j] == -1):
                V2[i][j] = V[i][j]
            else:
                V2[i][j] = (V[i-1][j] + V[i+1][j] + V[i][j-1] + V[i][j+1])/4
            dV += abs(V[i][j] - V2[i][j])
    return dV, V2


if __name__ == '__main__':
    count = [0]
    main(V, count)
    print(str(count[0]))
    plt.show()
```


Equipotential Lines
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
y
−0.4
−0.2
0.0
0.2
0.4
Electric Field
x
V = 1
V = 0
y
x

```
import matplotlib.pyplot as plt
import copy

import numpy as np
""
Chapter 5
Potentials and Fields
""
""
V = [
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

scalex = 9.1
scaley = 9
scaledim = 9
xlim = [-1.2,1.2]
ylim = [-1.2,1.2]
model = ((-0.24,-0.22), 0.45, 0.45)
""

V = [
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    ]

scalex = 5
scaley = 5
scaledim = 11
```

```
xlim = [-0.45,0.45]
ylim = [-0.45,0.45]
model = [(-0.24, -0.22), 0.45, 0.45]
"""
"""
fig = plt.figure(figsize=(10, 8))
fig.tight_layout(pad=1.0)


def equi(V):
    ax = fig.add_subplot(221)
    if(xlim is not None and ylim is not None):
        ax.set_xlim(xlim)
        ax.set_ylim(ylim)
    x = np.zeros((len(V), len(V)))
    y = np.zeros((len(V), len(V)))
    for i in range(1, len(V)-1):
        for j in range(1, len(V[i])-1):
            x[i][j] = ((j-scalex)/scaledim)
            y[i][j] = ((i-scaley)/scaledim)
    #ax.set_xlabel("x")
    ax.set_ylabel("y")
    ax.contour(x, y, V, zorder=0, colors='k', linestyles='solid')
    ax.axhline(y=0, linewidth=0.5, color='k')
    ax.axvline(x=0, linewidth=0.5, color='k')
    #ax.add_patch(plt.Rectangle(
       #model[0], model[1], model[2], color='black', fill=False, zorder=2))
    ax.set_title('Equipotential Lines')


def cal_electric_field(V):
    ax = fig.add_subplot(223)
    if(xlim is not None and ylim is not None):
        ax.set_xlim(xlim)
        ax.set_ylim(ylim)
    ax.set_title('Electric Field')
    ax.axhline(y=0, linewidth=0.5, color='k')
    ax.axvline(x=0, linewidth=0.5, color='k')
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    E_x = np.zeros((len(V), len(V)))
    E_y = np.zeros((len(V), len(V)))
    pair = dict()
    for i in range(0, len(V)-1):
        for j in range(0, len(V[i])-1):
            dx = ((j + 1) - (j - 1))
            dy = ((i + 1) - (i - 1))
            if(V[i][j] != 1):
                E_x[i][j] = -(V[i][j+1] - V[i][j-1]) / (2 * dx)
                E_y[i][j] = -(V[i+1][j] - V[i-1][j]) / (2 * dy)
            pair[(j, i)] = (E_x[i][j], E_y[i][j])
    for x, y in pair.keys():
        if(pair[(x, y)][0] == 0 and pair[(x, y)][1] == 0):
            pass
        else:
            ax.quiver((x-scalex)/scaledim, (y-scaley)/scaledim, pair[(x, y)][0], pair[(
                x, y)][1], scale=2.5, minshaft=2, minlength=-2, headwidth=4)
```

```
def create_3Ddiagram(V):
    ax = fig.add_subplot(122, projection='3d')
    V = np.array(V)
    x = np.zeros((len(V), len(V)))
    y = np.zeros((len(V), len(V)))
    for i in range(0, len(V)):
        for j in range(0, len(V[i])):
            x[i][j] = (j+1)
            y[i][j] = (i+1)
    ax.set_yticklabels([])
    ax.set_xticklabels([])
    ax.set_zticklabels(['V = 0', '', '', '', '', 'V = 1'])
    ax.plot_surface(x, y, V, rstride=1, cstride=1, cmap='binary')
    ax.set_xlabel("x")
    ax.set_ylabel("y")


def plot_graphs(V2):
    create_3Ddiagram(V2)
    cal_electric_field(V2)
    equi(V2)


def main(V, count, hold=False):
    count[0] += 1
    dV, V2 = updateV(V)
    if(dV <= 0.95 and not hold):
        hold = True
        create_3Ddiagram(V2)
    if(dV <= 0.1):
        cal_electric_field(V2)
        equi(V2)
    else:
        main(V2, count, hold)


def updateV(V):
    V2 = copy.deepcopy(V)
    dV = 0
    for i in range(1, len(V)-1):
        for j in range(1, len(V[i])-1):
            if(V[i][j] == 1 or V[i][j] == -1):
                V2[i][j] = V[i][j]
            else:
                V2[i][j] = (V[i-1][j] + V[i+1][j] + V[i][j-1] + V[i][j+1])/4
            dV += abs(V[i][j] - V2[i][j])
    return dV, V2


if __name__ == '__main__':
    count = [0]
    main(V, count)
    print(str(count[0]))
    plt.show()
```