Computational Physics

Assignment 5

Cheng-Ching Lin

1. Poisson Equation in 3D

(a). In Cartesian Coordinate

$$\nabla^2 \phi = -\frac{\rho}{\varepsilon_0}$$

$$\nabla^2 \phi + \frac{\rho}{\varepsilon_0} = 0$$

$$\frac{\phi^n_{i+1,j,k} + \phi^n_{i-1,j,k} - 2\phi^n_{i,j,k}}{h^2} + \frac{\phi^n_{i,j+1,k} + \phi^n_{i,j-1,k} - 2\phi^n_{i,j,k}}{h^2} + \frac{\phi^n_{i,j,k+1} + \phi^n_{i,j,k-1} - 2\phi^n_{i,j,k}}{h^2} + \frac{\rho_{i,j,k}}{\varepsilon_0} = 0$$

For

$$\begin{aligned} x_i &= h(i-1) + x_0 \\ y_j &= h(j-1) + y_0 \\ z_k &= h(k-1) + z_0 \\ t_n &= \tau(n-1) + t_0 \end{aligned}$$

We make time vary into the Poisson Equation. It will become some kinds of diffusion equation.

$$\nabla^2 \phi + \frac{\rho}{\varepsilon_0} = \frac{\partial \phi}{\partial t}$$

$$\frac{\phi^n_{i+1,j,k} + \phi^n_{i-1,j,k} - 2\phi^n_{i,j,k}}{h^2} + \frac{\phi^n_{i,j+1,k} + \phi^n_{i,j-1,k} - 2\phi^n_{i,j,k}}{h^2} + \frac{\phi^n_{i,j,k+1} + \phi^n_{i,j,k-1} - 2\phi^n_{i,j,k}}{h^2} + \frac{\rho_{i,j,k}}{\varepsilon_0} = \frac{\phi^{n+1}_{i,j,k} - \phi^n_{i,j,k}}{\tau}$$

$$\frac{\tau}{h^2}\left(\phi^n_{i+1,j,k} + \phi^n_{i-1,j,k} + \phi^n_{i,j+1,k} + \phi^n_{i,j-1,k} + \phi^n_{i,j,k+1} + \phi^n_{i,j,k-1} - 6\phi^n_{i,j,k}\right) + \phi^n_{i,j,k} + \frac{\tau \rho_{i,j,k}}{\varepsilon_0} = \phi^{n+1}_{i,j,k}$$

For the stability of this equation and also to make this equation simpler, we choose $\frac{\tau}{h^2} = \frac{1}{6}$.

$$\frac{\left(\phi^n_{i+1,j,k} + \phi^n_{i-1,j,k} + \phi^n_{i,j+1,k} + \phi^n_{i,j-1,k} + \phi^n_{i,j,k+1} + \phi^n_{i,j,k-1}\right)}{6} + \frac{h^2 \rho_{i,j,k}}{6\varepsilon_0} = \phi^{n+1}_{i,j,k}$$

(b). In cylinder coordinate

In cylinder coordinate, the Laplace Operator will become this terms:

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} = \frac{1}{r}\frac{\partial \phi}{\partial r} + \frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2 \phi}{\partial \psi^2} + \frac{\partial^2 \phi}{\partial z^2}$$

For

$$\begin{aligned} r_i &= h(i-1) + r_0 \\ \psi_j &= h(j-1) + \psi_0 \\ z_k &= h(k-1) + z_0 \\ t_n &= \tau(n-1) + t_0 \\ 0 \le\ & h \le 2\pi \end{aligned}$$

$$\frac{1}{r_i}\frac{\phi^n_{i+1,j,k}-\phi^n_{i-1,j,k}}{2h}+\frac{\phi^n_{i+1,j,k}+\phi^n_{i-1,j,k}-2\phi^n_{i,j,k}}{h^2}+\frac{1}{r_i^2}\frac{\phi^n_{i,j+1,k}+\phi^n_{i,j-1,k}-2\phi^n_{i,j,k}}{h^2}$$
$$+\frac{\phi^n_{i,j,k+1}+\phi^n_{i,j,k-1}-2\phi^n_{i,j,k}}{h^2}+\frac{\rho_{i,j,k}}{\varepsilon_0}=\frac{\phi^{n+1}_{i,j,k}-\phi^n_{i,j,k}}{\tau}$$

$$\frac{\tau}{h^2r_i^2}\left(\frac{h}{2}r_i\phi^n_{i+1,j,k}-\frac{h}{2}r_i\phi^n_{i-1,j,k}+r_i^2\phi^n_{i+1,j,k}+r_i^2\phi^n_{i-1,j,k}-2r_i^2\phi^n_{i,j,k}+\phi^n_{i,j+1,k}+\phi^n_{i,j-1,k}-2\phi^n_{i,j,k}\right.$$
$$\left.+r_i^2\phi^n_{i,j,k+1}+r_i^2\phi^n_{i,j,k-1}-2r_i^2\phi^n_{i,j,k}\right)+\phi^n_{i,j,k}+\frac{\tau\rho_{i,j,k}}{\varepsilon_0}=\phi^{n+1}_{i,j,k}$$

$$\frac{\tau}{h^2r_i^2}\left(\frac{h}{2}r_i\phi^n_{i+1,j,k}-\frac{h}{2}r_i\phi^n_{i-1,j,k}+r_i^2\phi^n_{i+1,j,k}+r_i^2\phi^n_{i-1,j,k}+\phi^n_{i,j+1,k}+\phi^n_{i,j-1,k}+r_i^2\phi^n_{i,j,k-1}+r_i^2\phi^n_{i,j,k+1}\right)$$
$$+\frac{\tau}{h^2r_i^2}(-2r_i^2-2-2r_i^2)\phi^n_{i,j,k}+\phi^n_{i,j,k}+\frac{\tau\rho_{i,j,k}}{\varepsilon_0}=\phi^{n+1}_{i,j,k}$$

If we do what we just do in (a) and make it simpler, then we need to solve the equation.

$$\frac{\tau}{h^2r_i^2}(4r_i^2+2)=1$$

And it's very complicated, so it shows that cylinder coordinate doesn't prefer to use in numerical solution.

(c). In spherical coordinate

In cylinder coordinate, the Laplace Operator will become this terms:

$$\nabla^2\phi=\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}=\frac{2}{r}\frac{\partial\phi}{\partial r}+\frac{\partial^2\phi}{\partial r^2}+\frac{cot\theta}{r^2}\frac{\partial\phi}{\partial\theta}+\frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2}+\frac{1}{r^2sin\theta}\frac{\partial^2\phi}{\partial\psi^2}$$

For

$$r_i=h(i-1)+r_0$$
$$\theta_j=h(j-1)+\theta_0$$
$$\psi_k=h(k-1)+\psi_0$$
$$t_n=\tau(n-1)+t_0$$
$$0\le h\le 2\pi$$

$$\frac{2}{r_i}\frac{\phi^n_{i+1,j,k}-\phi^n_{i-1,j,k}}{2h}+\frac{\phi^n_{i+1,j,k}+\phi^n_{i-1,j,k}-2\phi^n_{i,j,k}}{h^2}+\frac{cot\theta_j}{r_i^2}\frac{\phi^n_{i,j+1,k}-\phi^n_{i,j-1,k}}{2h}$$
$$+\frac{1}{r_i^2}\frac{\phi^n_{i,j+1,k}+\phi^n_{i,j-1,k}-2\phi^n_{i,j,k}}{h^2}+\frac{1}{r_i^2sin\theta_j}\frac{\phi^n_{i,j,k+1}+\phi^n_{i,j,k-1}-2\phi^n_{i,j,k}}{h^2}+\frac{\phi^n_{i,j,k}}{\tau}+\frac{\rho_{i,j,k}}{\varepsilon_0}$$
$$=\frac{\phi^{n+1}_{i,j,k}}{\tau}$$

$$\frac{\tau cos\theta_j}{r_i^2h^2sin\theta_j}\left(r_ihtan\theta_j\phi^n_{i+1,j,k}-r_ihtan\theta_j\phi^n_{i-1,j,k}+r_i^2tan\theta_j\phi^n_{r+1,i,j}+r_i^2tan\theta_j\phi^n_{i-1,j,k}+\frac{h}{2}\phi^n_{i,j+1,k}\right.$$
$$\left.-\frac{h}{2}\phi^n_{i,j-1,k}+tan\theta_j\phi^n_{i,j-1,k}+tan\theta_j\phi^n_{i,j+1,k}+\frac{h^2}{cos\theta_j}\phi^n_{i,j,k-1}+\frac{h^2}{cos\theta_j}\phi^n_{i,j,k+1}\right)$$
$$+\frac{\tau cos\theta_j}{r_i^2h^2sin\theta_j}\left(-2r_i^2tan\theta_j-2tan\theta_j-\frac{2h^2}{cos\theta_j}\right)\phi^n_{i,j,k}+\phi^n_{i,j,k}+\frac{\tau\rho_{i,j,k}}{\varepsilon_0}=\phi^{n+1}_{i,j,k}$$

So,…

$$\frac{2\tau\cos\theta_j}{r_i^2 h^2 sin\theta_j}\left(-2r_i^2 tan\theta_j - 2tan\theta_j - \frac{2h^2}{cos\theta_j}\right) = -1$$

We can see that the equation is so hard to solve it.

Therefore, the Cartesian coordinate always prefers to use in numerical solutions.

2. A charge in a box

First, I create the function for calculate the electric potential.

Code:

```
import matplotlib.pyplot as plt
import numpy as np
import math

def phi(i,j,k,n,pp,h):
    tau = pow(h, 2) / 6 #I suppose tau = (h^2)/6
    t = n*tau
    xi =int(round((0.5-i)/h))
    yj =int(round((0.5-j)/h))
    zk =int(round((0.5-k)/h))
    if i == 0 and j == 0  and k == 0: # At the origin, there is a charge. I use Poisson equation here.
        ans = (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
            xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1] + pp[xi, yj, zk - 1, n - 1]+(1*h*h)) / 6
    elif i == 0.5 or i == -0.5 or j == 0.5 or j == -0.5 or k == 0.5 or k == -0.5: #Boundary condition
        ans = 0
    elif t==0: #Initial condition
        ans = 0
    else: #other place in box, I use Laplace equation to solve.
        ans =
(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/
6
    return ans
```

(1). Calculate the electric potential in 10*10*10 grid and 100*100*100 grid, and also plot the electric potential on (0,0,z)

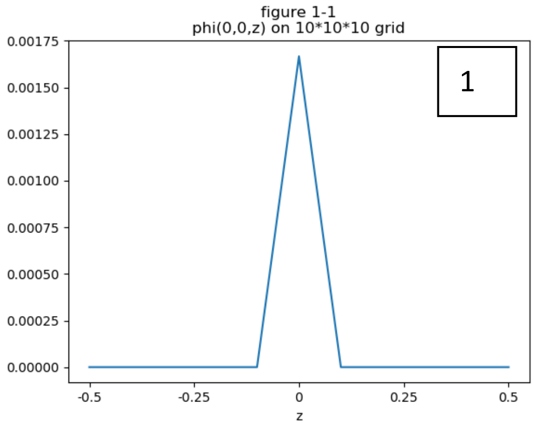
Code:

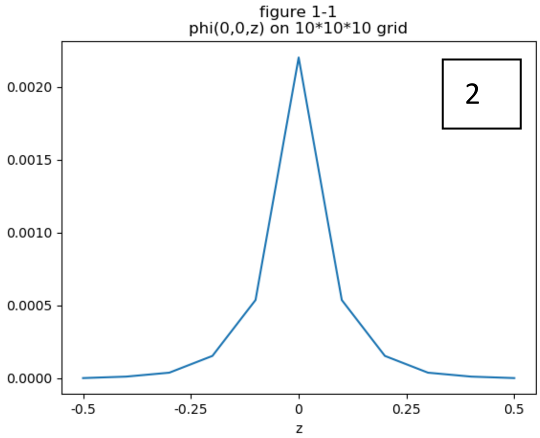
```
def test_1():
    h = 0.1 #the space span is 0.1
    pp = np.zeros((11,11,11,101))
    for ndx in range(101): # 100 steps
        n = ndx
        for kdx in range(11): # 10 steps on z-axis
            k = 0.5-kdx*h # calculate the real distance
            for jdx in range(11):# 10 steps on y-axis
                j = 0.5-jdx*h # calculate the real distance
                for idx in range(11):# 10 steps on x-axis
                    i = 0.5-idx*h # calculate the real distance
                    temp = phi(i,j,k,n,pp,h)
                    pp[idx, jdx, kdx, ndx] = temp
    if ndx%10==0:
```

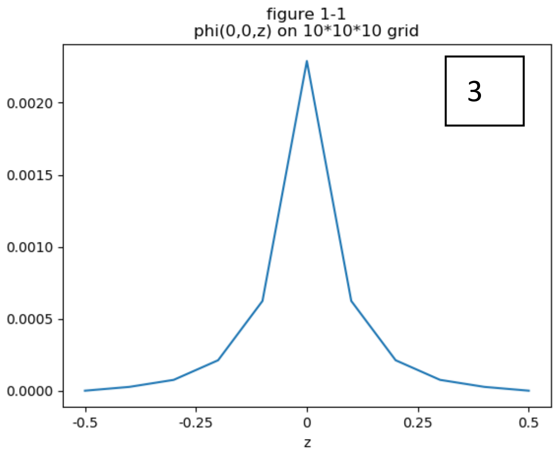
```
        plt.plot(pp[50, 50, :, ndx])
        plt.xticks([0, 25, 50, 75, 100], [-0.5, -0.25, 0, 0.25, 0.5])
        plt.title("figure 1-2\nphi(0,0,z) on 100*100*100 grid")
        plt.xlabel("z")
plt.show()
    return 0

def test_2():
    h = 0.01 #the space span is 0.01
    pp = np.zeros((101, 101, 101, 101))
    for ndx in range(101):# 100 steps on time
        n = ndx
        for kdx in range(101):# 100 steps on z-axis
            k = 0.5 - kdx * h
            for jdx in range(101):# 100 steps on y-axis
                j = 0.5 - jdx * h
                for idx in range(101):# 100 steps on x-axis
                    i = 0.5 - idx * h
                    temp = phi(i, j, k, n, pp, h)
                    pp[idx, jdx, kdx, ndx] = temp
   if ndx%10==0:
        plt.plot(pp[50, 50, :, ndx])
        plt.xticks([0, 25, 50, 75, 100], [-0.5, -0.25, 0, 0.25, 0.5])
        plt.title("figure 1-2\nphi(0,0,z) on 100*100*100 grid")
        plt.xlabel("z")
plt.show()

    return 0
```

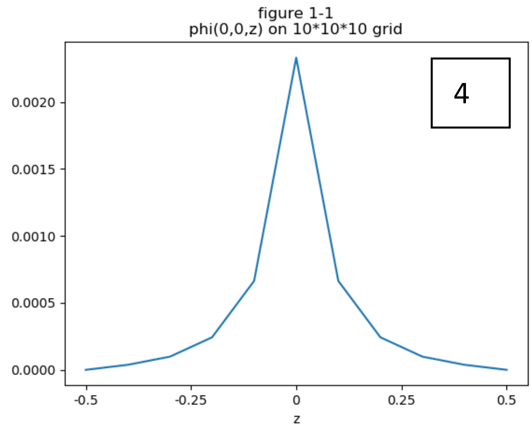
and the result:

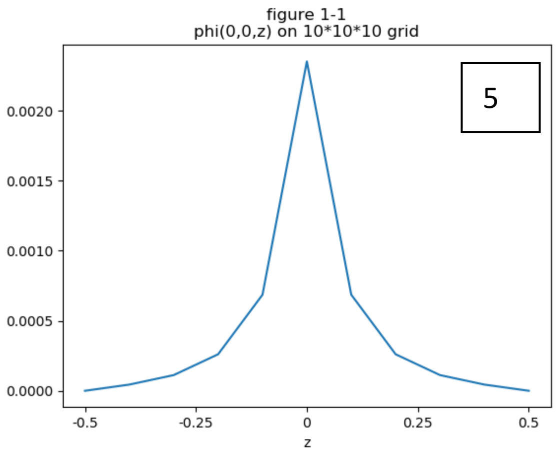
10*10*10

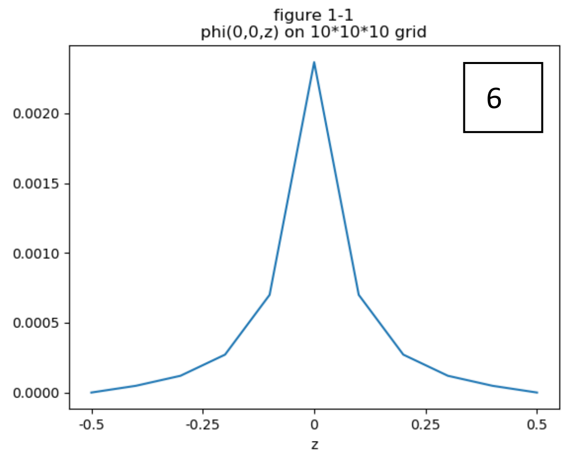


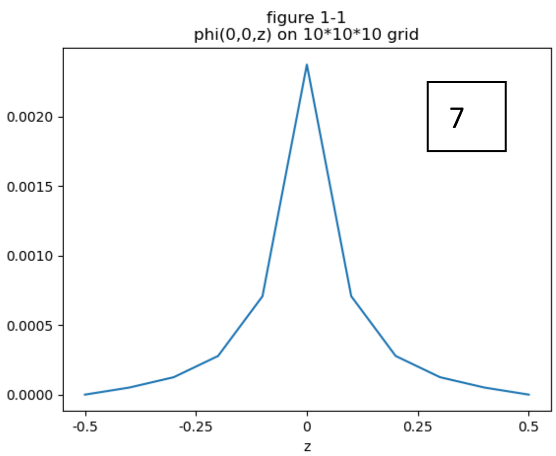


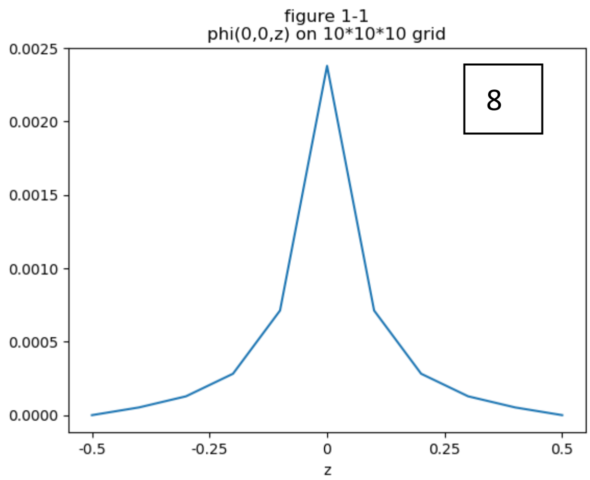












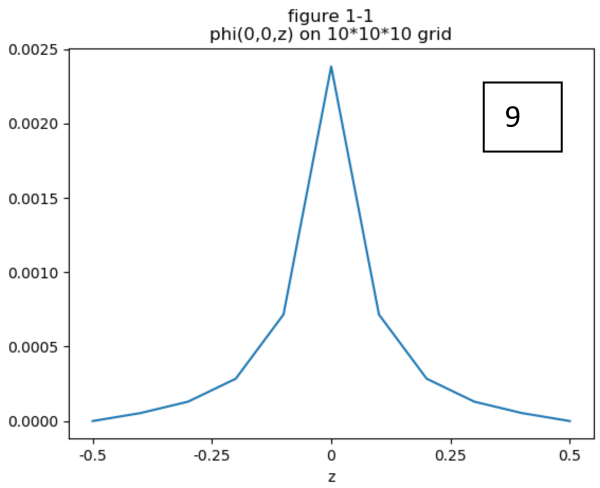


figure 1-1
phi(0,0,z) on 10*10*10 grid
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
-0.5
-0.25
0
0.25
0.5
z
9

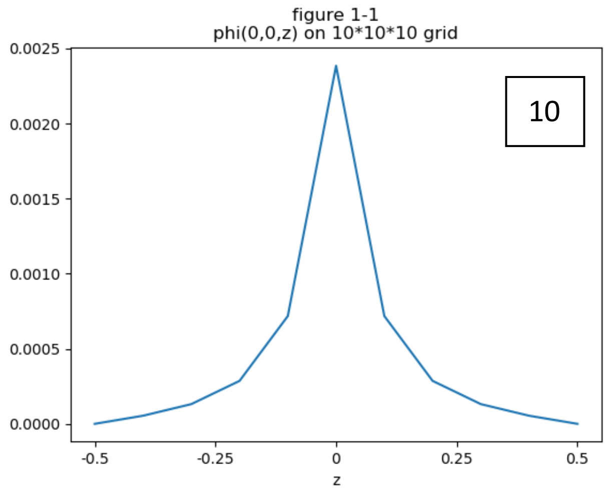


figure 1-1
phi(0,0,z) on 10*10*10 grid
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
-0.5
-0.25
0
0.25
0.5
z
10

100*100*100

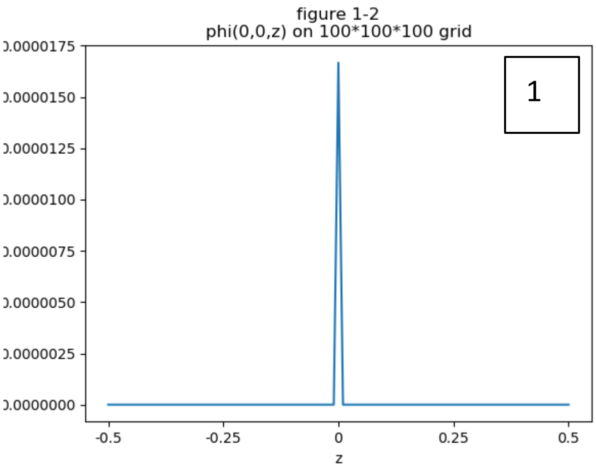


figure 1-2
phi(0,0,z) on 100*100*100 grid
1
z

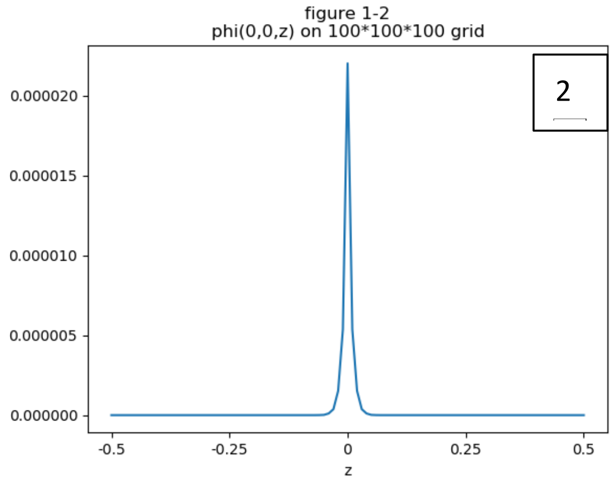


figure 1-2
phi(0,0,z) on 100*100*100 grid
2
z

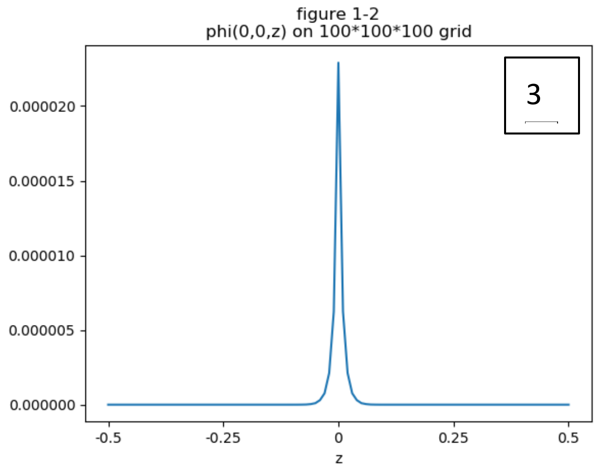


figure 1-2
phi(0,0,z) on 100*100*100 grid
3
z

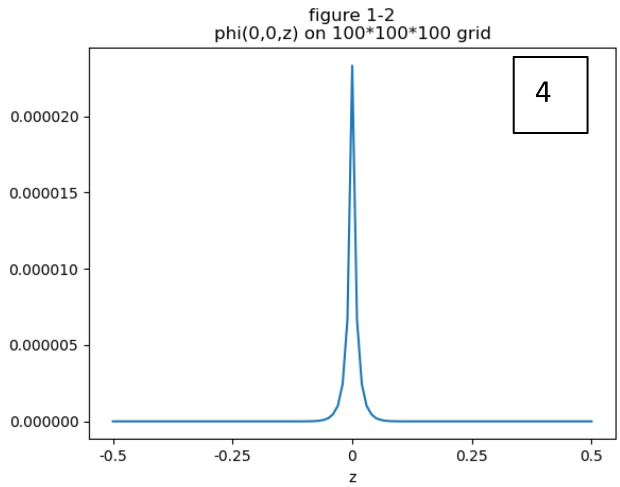


figure 1-2
phi(0,0,z) on 100*100*100 grid
4
z

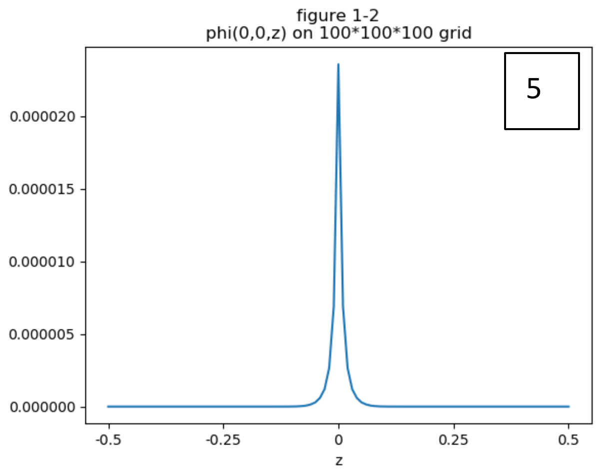


figure 1-2
phi(0,0,z) on 100*100*100 grid
5
z

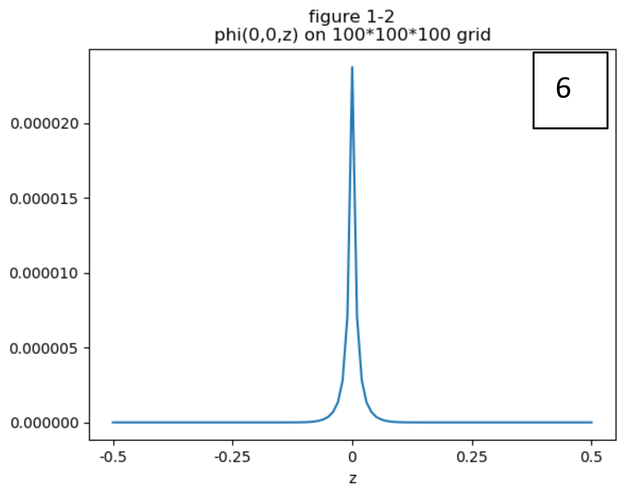
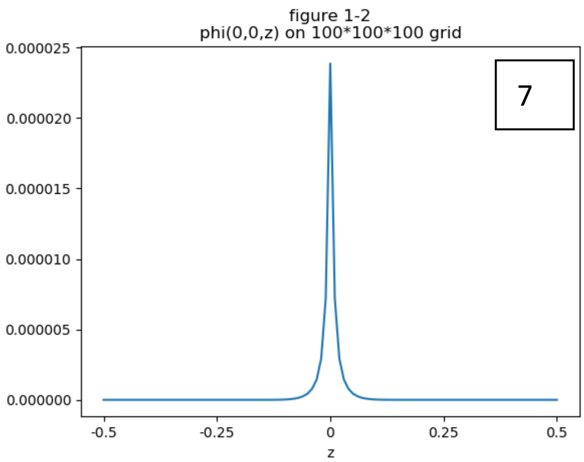
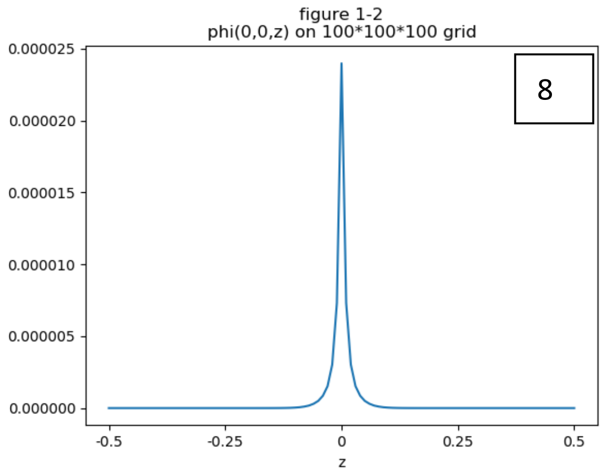
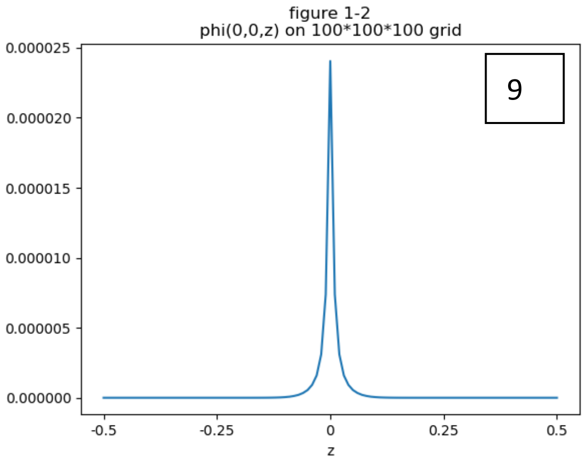
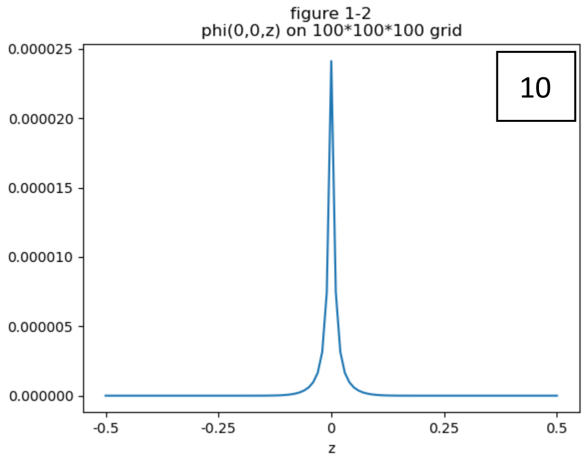


figure 1-2
phi(0,0,z) on 100*100*100 grid
6
z









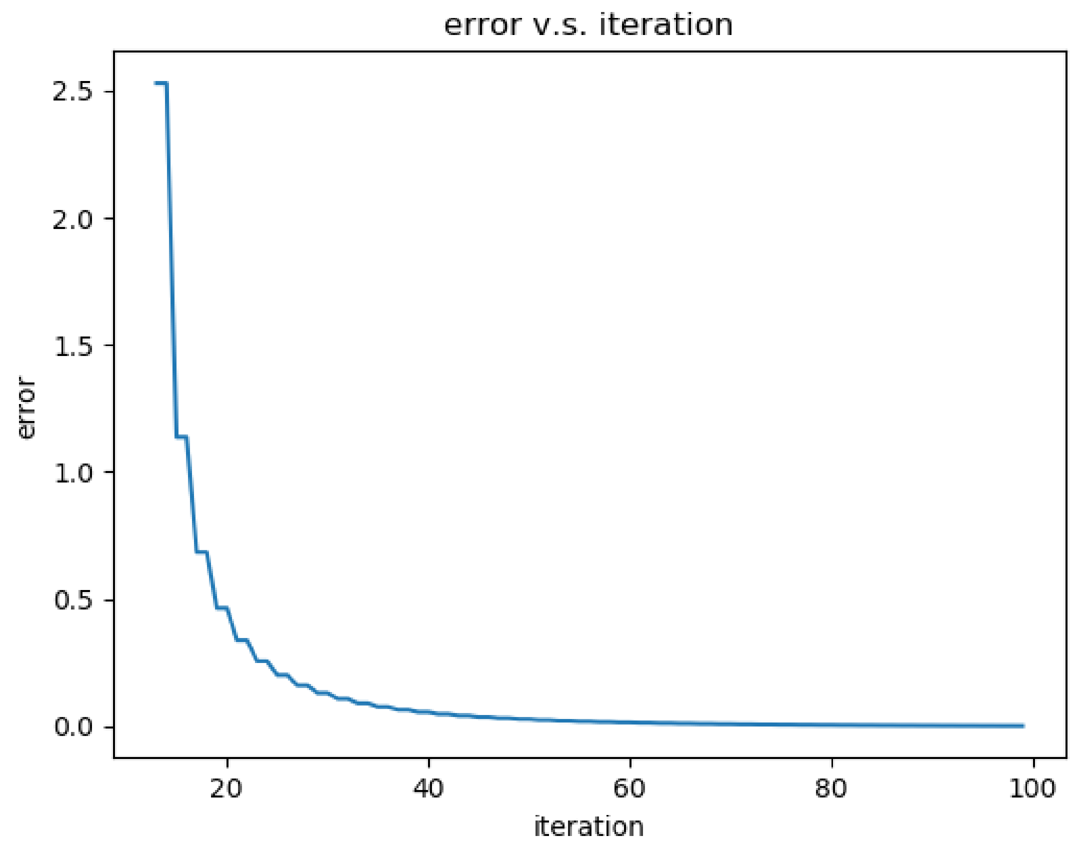
Because the equation is some kinds of distribution equation, we can see some kind of distribution in this figure. We can see that in the same time steps, the second figure is more sharper than first one, that is because the space resolutions is more higher, we can see that there is a peak (a charge) in the center.

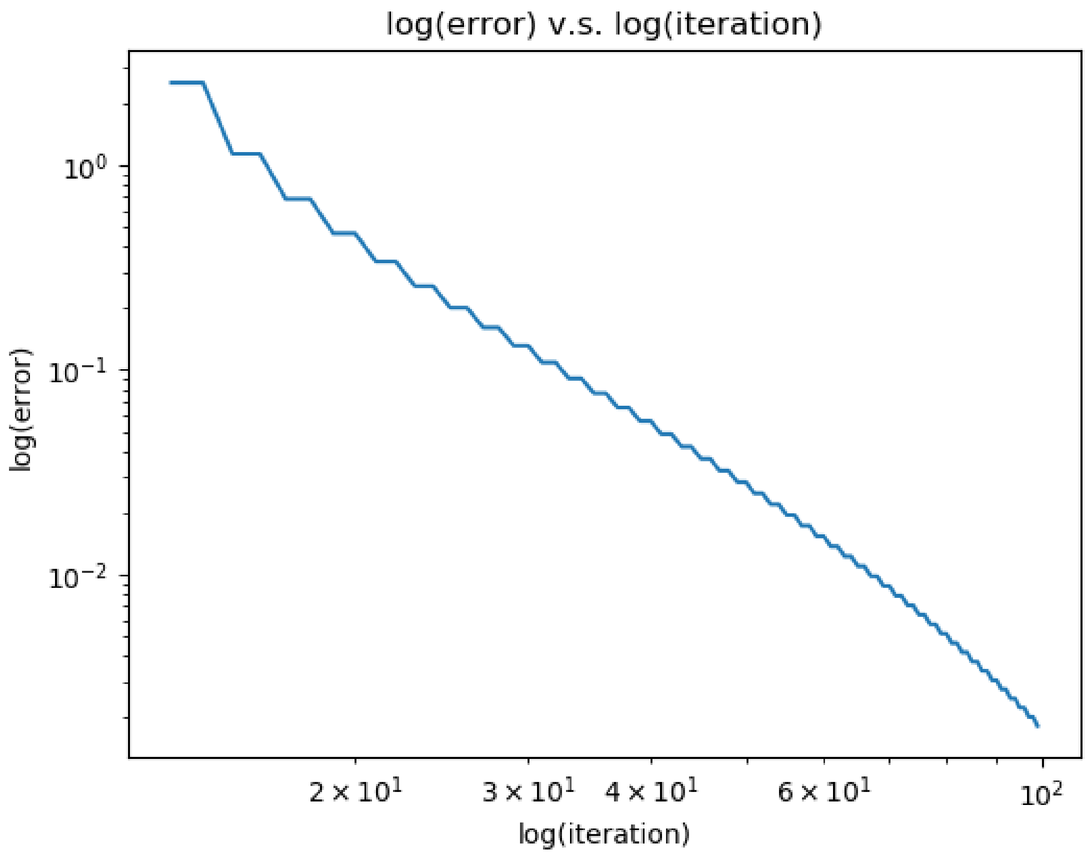
(2). The error v.s. iterations

Code:

```
def test_3():
    h = 0.1
    pp = np.zeros((11,11,11,101))
    for ndx in range(101):
        n = ndx
        for kdx in range(11):
            k = 0.5-kdx*h
            for jdx in range(11):
                j = 0.5-jdx*h
                for idx in range(11):
                    i = 0.5-idx*h
                    temp = phi(i,j,k,n,pp,h)
                    pp[idx, jdx, kdx, ndx] = temp
    deltatemp =0
    deltamax = 0
    colmax = []

    for ndx in range(1,101):
        for jdx in range(11):
            for idx in range(11):
                for kdx in range(11):
                    if pp[idx,jdx,kdx,ndx]!=0 or pp[idx,jdx,kdx,ndx-1]!=0:
                        deltatemp = (pp[idx, jdx, kdx, ndx] - pp[idx, jdx, kdx, ndx - 1])/pp[idx, jdx, kdx, ndx - 1]
                    if deltatemp > deltamax:
                        deltamax = deltatemp
        print(ndx)
        print(deltamax)
        colmax.append(deltamax)
        deltamax = 0
    plt.plot(colmax)
    plt.title("error v.s. iteration")
    plt.xlabel("iteration")
    plt.ylabel("error")
    plt.show()
    plt.loglog(colmax)
    plt.title("log(error) v.s. log(iteration)")
    plt.xlabel("log(iteration)")
    plt.ylabel("log(error)")
    plt.show()
```

And the result is:





(3). Calculate the electric potential in 10*10*10 grid and 100*100*100 grid, and also plot the electric potential on (0,0,z).

This time, the boundary has electric potential, and there is no charge in origin:

$$\frac{q}{\sqrt{x^2 + y^2 + z^2}}$$

First, I need to create another function to calculate the electric potential.

Code:

```
def phi_2(i,j,k,n,pp,h):
    tau = pow(h, 2) / 6
    t = n*tau
    xi = int(round((1.5 - i) / h))
    yj = int(round((1.5 - j) / h))
    zk = int(round((1.5 - k) / h))
    if i == 0.5 or i == -0.5 or j == 0.5 or j == -0.5 or k == 0.5 or k == -0.5:
        if i<-0.5 or i>0.5 or j<-0.5 or j>0.5 or k<-0.5 or k>0.5 :
            if t == 0:
                ans = 0
            elif i>=1.5 or i<=-1.5 or j>=1.5 or j<=-1.5 or k>=1.5 or k<=-1.5:
                ans =0
            else:
                ans = (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
                    xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
        elif i<-0.5 or i>0.5 or j<-0.5 or j>0.5 or k<-0.5 or k>0.5:
            ans =
(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/
6
        else:#source term
            ans = (1)/(math.sqrt(pow(i,2)+pow(j,2)+pow(k,2)))
    elif t==0:# initial condition
        ans = 0
    #boundary term
    elif k >= 1.5:
        if j >= 1.5:
            if i>=1.5:
                ans = (pp[xi + 1, yj, zk, n - 1] + pp[
                    xi, yj + 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1]) / 6
            elif i<=-1.5:
                ans =(pp[xi - 1, yj, zk, n - 1]  + pp[
                    xi, yj + 1, zk, n - 1]  + pp[xi, yj, zk + 1, n - 1]) / 6
            else :
                ans =(pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[
                    xi, yj + 1, zk, n - 1]  + pp[xi, yj, zk + 1, n - 1]) / 6
```

```
        elif j<=-1.5:
            if i>=1.5:
                ans =( pp[xi + 1, yj, zk, n - 1] + pp[xi, yj - 1, zk, n - 1]  + pp[xi, yj, zk + 1, n - 1]) / 6
            elif i<=-1.5:
                ans =(pp[xi - 1, yj, zk, n - 1] + pp[xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1]) / 6
            else:
                ans = (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj - 1, zk, n - 1]  + pp[xi, yj,
zk + 1, n - 1]) / 6
        elif i>=1.5:
            ans = (pp[xi + 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
                xi, yj - 1, zk, n - 1]  + pp[xi, yj, zk + 1, n - 1]) / 6
        elif i <=-1.5:
            ans = (pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
                xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1]) / 6
        else:
            ans = (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
                xi, yj - 1, zk, n - 1] + pp[xi, yj, zk+ 1, n - 1]) / 6
    elif k <= -1.5:
        if j >= 1.5:
            if i>=1.5:
                ans = (pp[xi + 1, yj, zk, n - 1] + pp[
                    xi, yj + 1, zk, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
            elif i<=-1.5:
                ans =(pp[xi - 1, yj, zk, n - 1]  + pp[
                    xi, yj + 1, zk, n - 1]  + pp[xi, yj, zk - 1, n - 1]) / 6
            else :
                ans =(pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[
                    xi, yj + 1, zk, n - 1]  + pp[xi, yj, zk - 1, n - 1]) / 6
        elif j<=-1.5:
            if i>=1.5:
                ans =( pp[xi + 1, yj, zk, n - 1] + pp[xi, yj - 1, zk, n - 1]  + pp[xi, yj, zk - 1, n - 1]) / 6
            elif i<=-1.5:
                ans =(pp[xi - 1, yj, zk, n - 1] + pp[xi, yj - 1, zk, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
            else:
                ans = (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj - 1, zk, n - 1]  + pp[xi, yj,
zk - 1, n - 1]) / 6
        elif i>=1.5:
            ans = (pp[xi + 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
                xi, yj - 1, zk, n - 1]  + pp[xi, yj, zk - 1, n - 1]) / 6
        elif i <=-1.5:
            ans = (pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
                xi, yj - 1, zk, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
```

```
        else:
            ans = (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
                xi, yj - 1, zk, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
    elif i>=1.5 or i<=-1.5 or j>=1.5 or j<=-1.5 or k>=1.5 or k<=-1.5:
        ans = 0
    else:
        ans =
(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/
6
    return ans
```

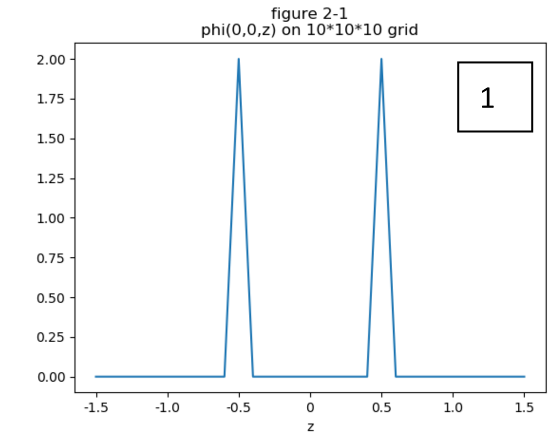
After this code, we need to calculate the electric potential in 10*10*10 grid and 100*100*100 grid.
Code:

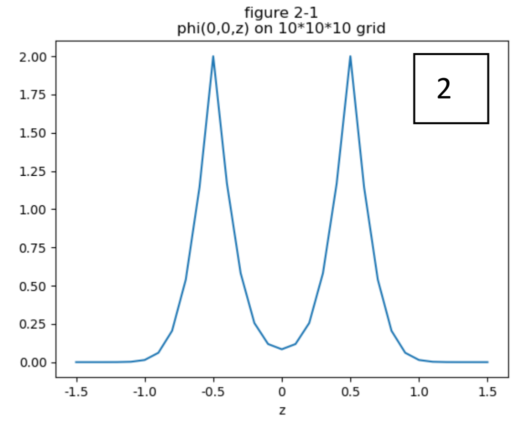
```
def test_4():
    h = 0.1
    pp = np.zeros((31,31,31,101))
    for ndx in range(101):
        n = ndx
        for kdx in range(31):
            k = 1.5-kdx*h
            for jdx in range(31):
                j = 1.5-jdx*h
                for idx in range(31):
                    i = 1.5-idx*h
                    temp = phi_2(i,j,k,n,pp,h)
                    pp[idx, jdx, kdx, ndx] = temp
        if ndx%10 == 0:
            plt.plot(pp[15,15,:,ndx])
            plt.xticks([0,5,10,15,20, 25, 30], [-1.5,-1.0, -0.5, 0, 0.5, 1.0, 1.5])
            plt.title("figure 2-1\nphi(0,0,z) on 10*10*10 grid")
            plt.xlabel("z")
            plt.show()


def test_5():
    h = 0.01
    pp = np.zeros((301, 301, 301, 11))
    for ndx in range(11):
        n = ndx
        for kdx in range(301):
            k = 1.5 - kdx * h
            for jdx in range(301):
                j = 1.5 - jdx * h
```

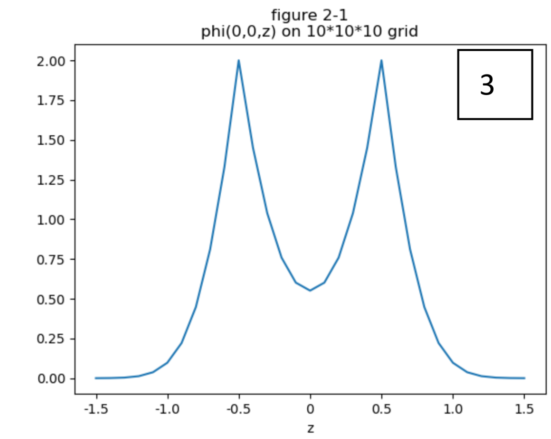
```
        for idx in range(301):
            i = 1.5 - idx * h
            temp = phi_2(i, j, k, n, pp, h)
            pp[idx, jdx, kdx, ndx] = temp
plt.plot(pp[150, 150, :, ndx])
plt.xticks([0, 50, 100, 150, 200, 250, 300], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.title("figure 2-2\nphi(0,0,z) on 100*100*100 grid")
plt.xlabel("z")
plt.show()
```

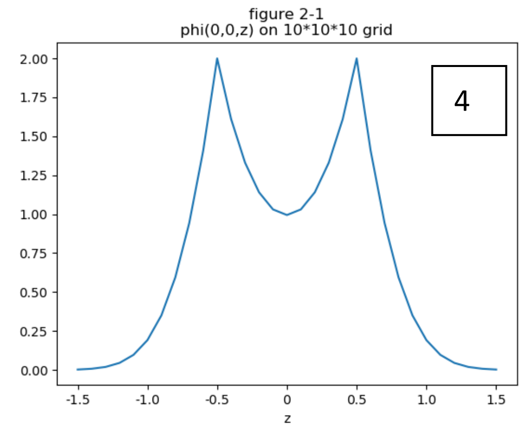
And the result:

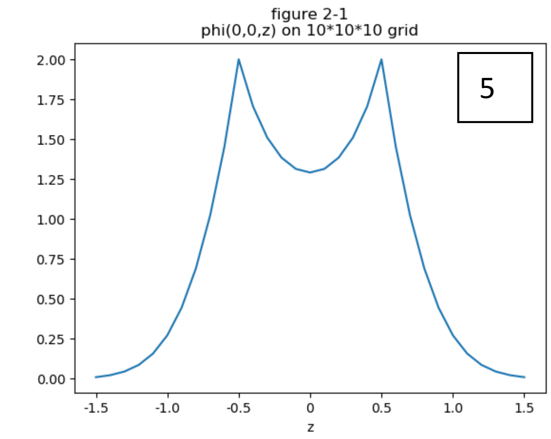
10*10*10

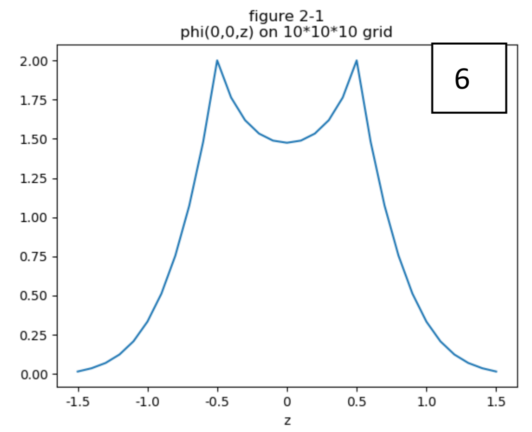


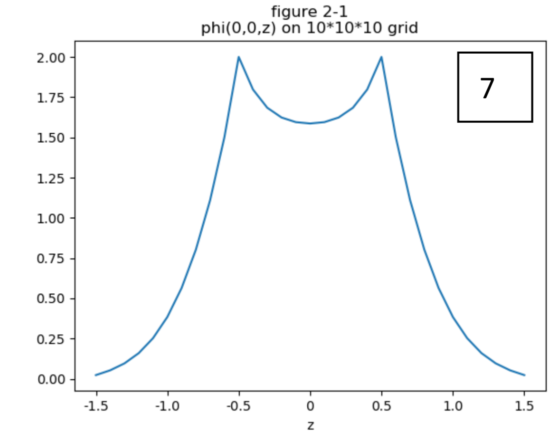


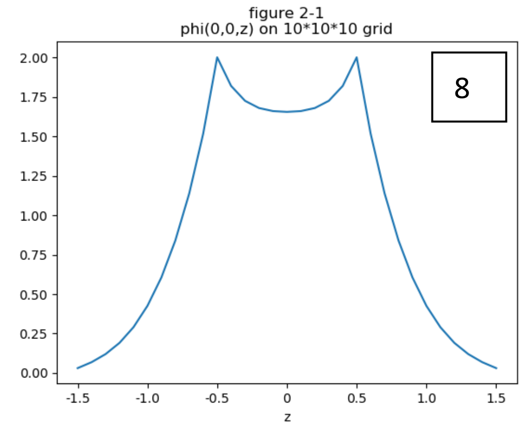












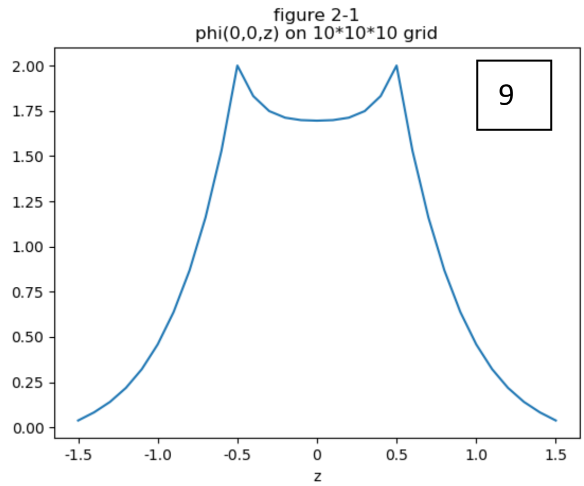


figure 2-1
phi(0,0,z) on 10*10*10 grid
9
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-1.5
-1.0
-0.5
0
0.5
1.0
1.5
z

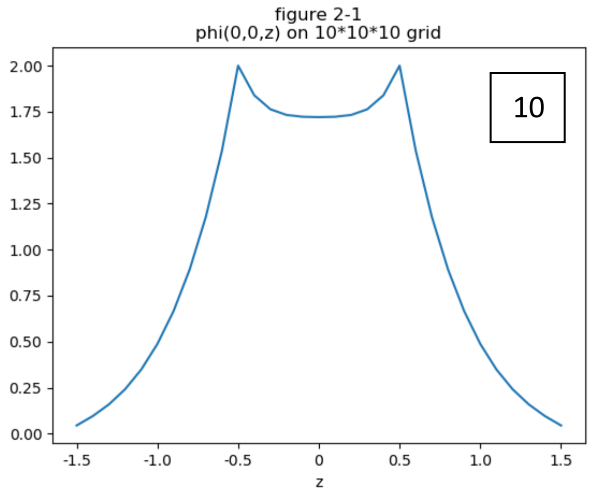


figure 2-1
phi(0,0,z) on 10*10*10 grid
10
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-1.5
-1.0
-0.5
0
0.5
1.0
1.5
z

100*100*100:

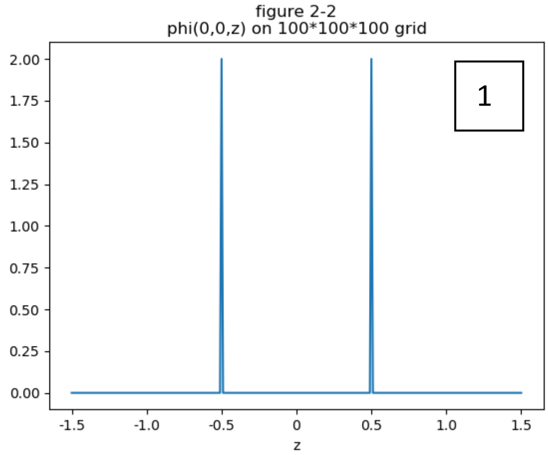


figure 2-2
phi(0,0,z) on 100*100*100 grid
1
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-1.5
-1.0
-0.5
0
0.5
1.0
1.5
z

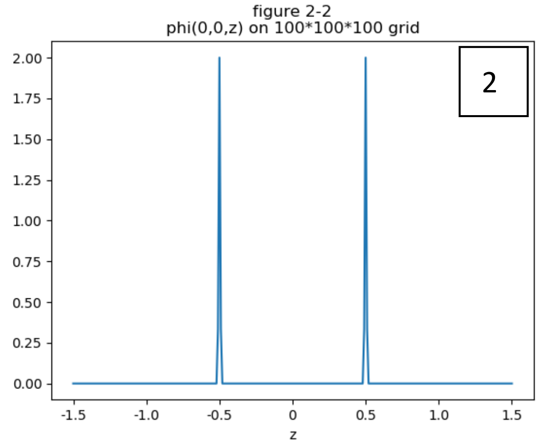


figure 2-2
phi(0,0,z) on 100*100*100 grid
2
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-1.5
-1.0
-0.5
0
0.5
1.0
1.5
z

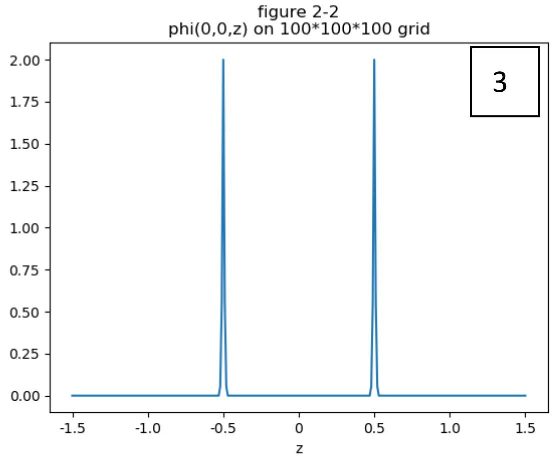


figure 2-2
phi(0,0,z) on 100*100*100 grid
3
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-1.5
-1.0
-0.5
0
0.5
1.0
1.5
z

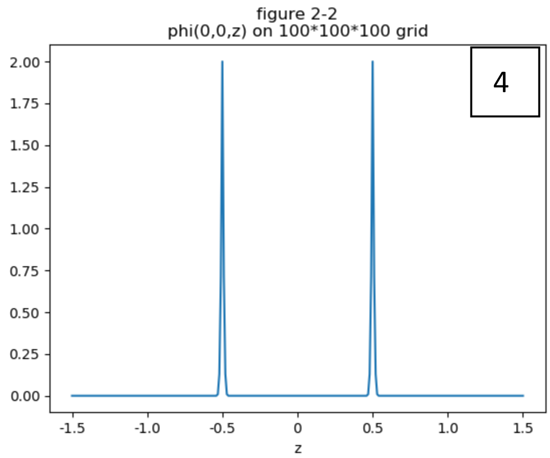


figure 2-2
phi(0,0,z) on 100*100*100 grid
4
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-1.5
-1.0
-0.5
0
0.5
1.0
1.5
z

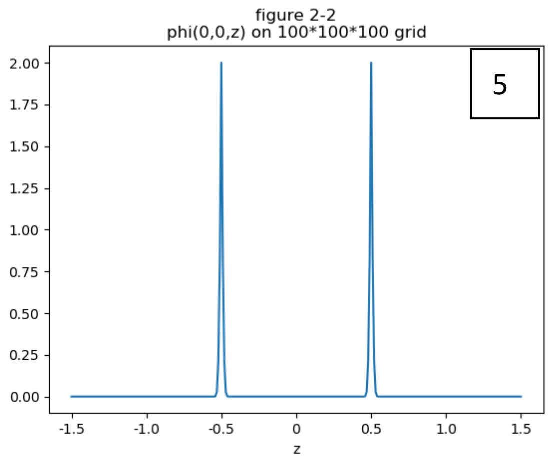


figure 2-2
phi(0,0,z) on 100*100*100 grid
5
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-1.5
-1.0
-0.5
0
0.5
1.0
1.5
z

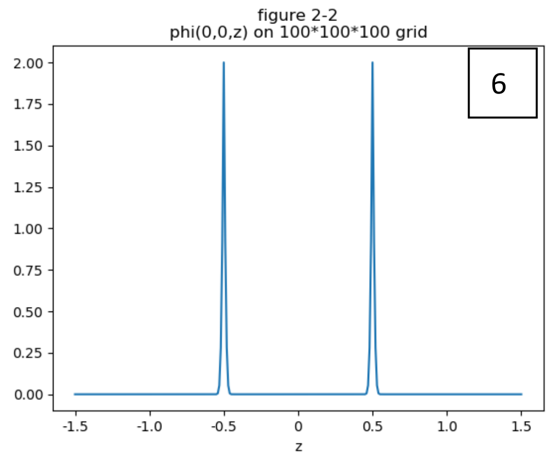


figure 2-2
phi(0,0,z) on 100*100*100 grid
6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-1.5
-1.0
-0.5
0
0.5
1.0
1.5
z

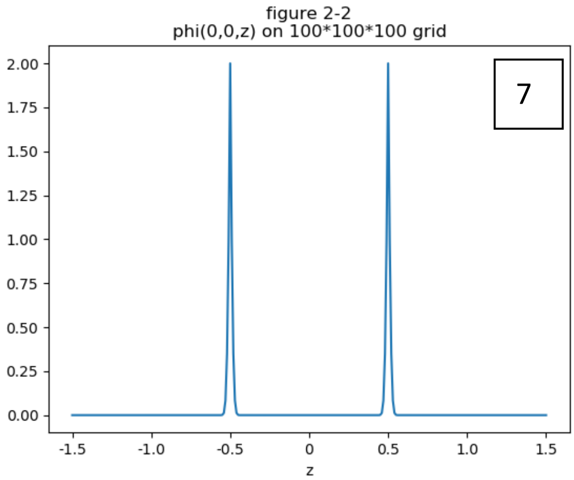


figure 2-2
phi(0,0,z) on 100*100*100 grid
7
z

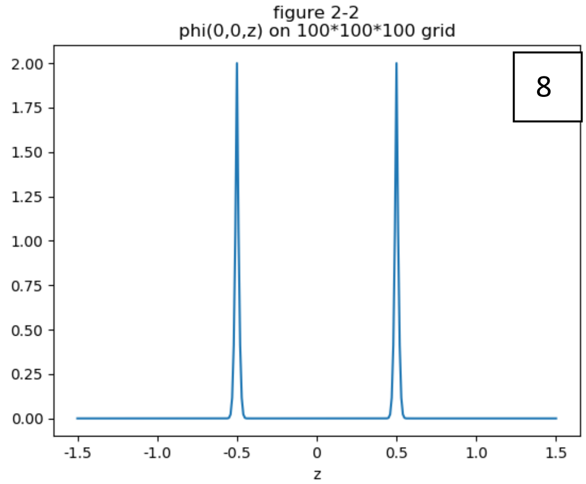


figure 2-2
phi(0,0,z) on 100*100*100 grid
8
z

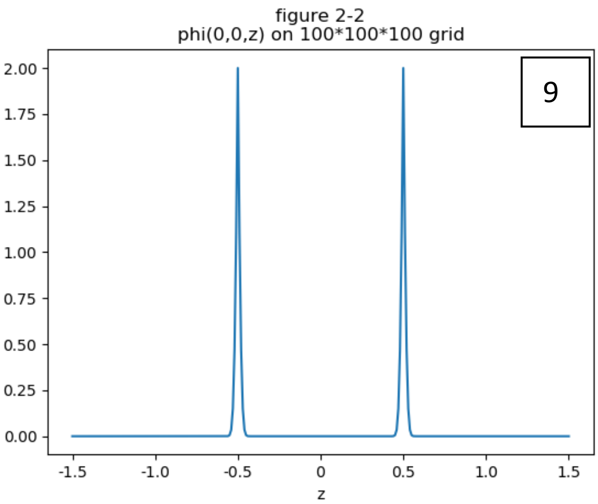


figure 2-2
phi(0,0,z) on 100*100*100 grid
9
z

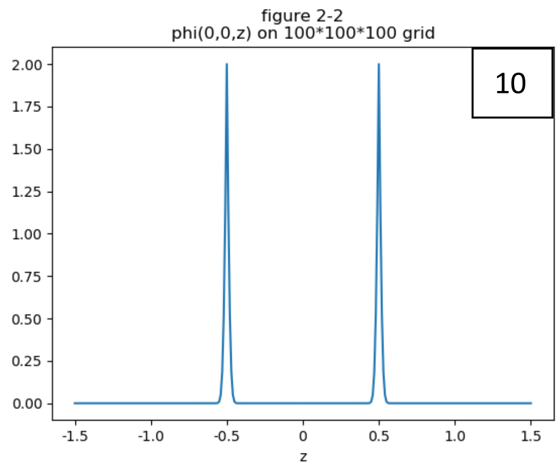


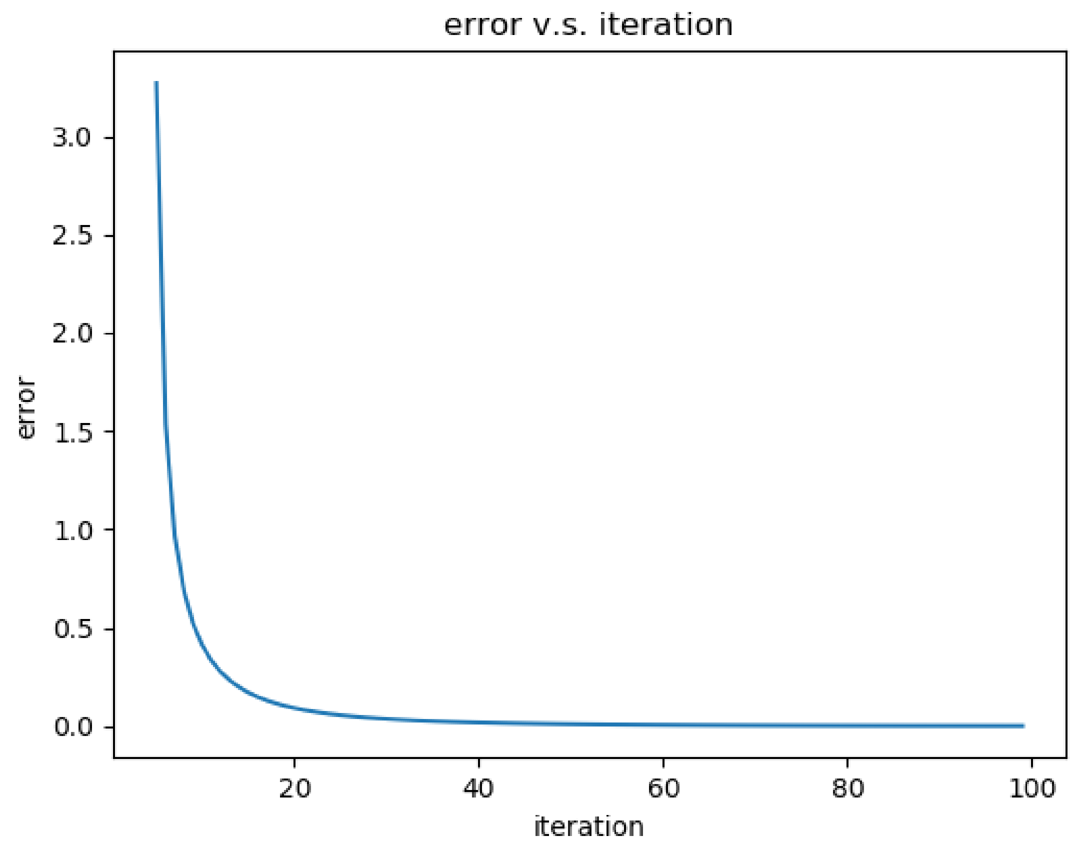
figure 2-2
phi(0,0,z) on 100*100*100 grid
10
z

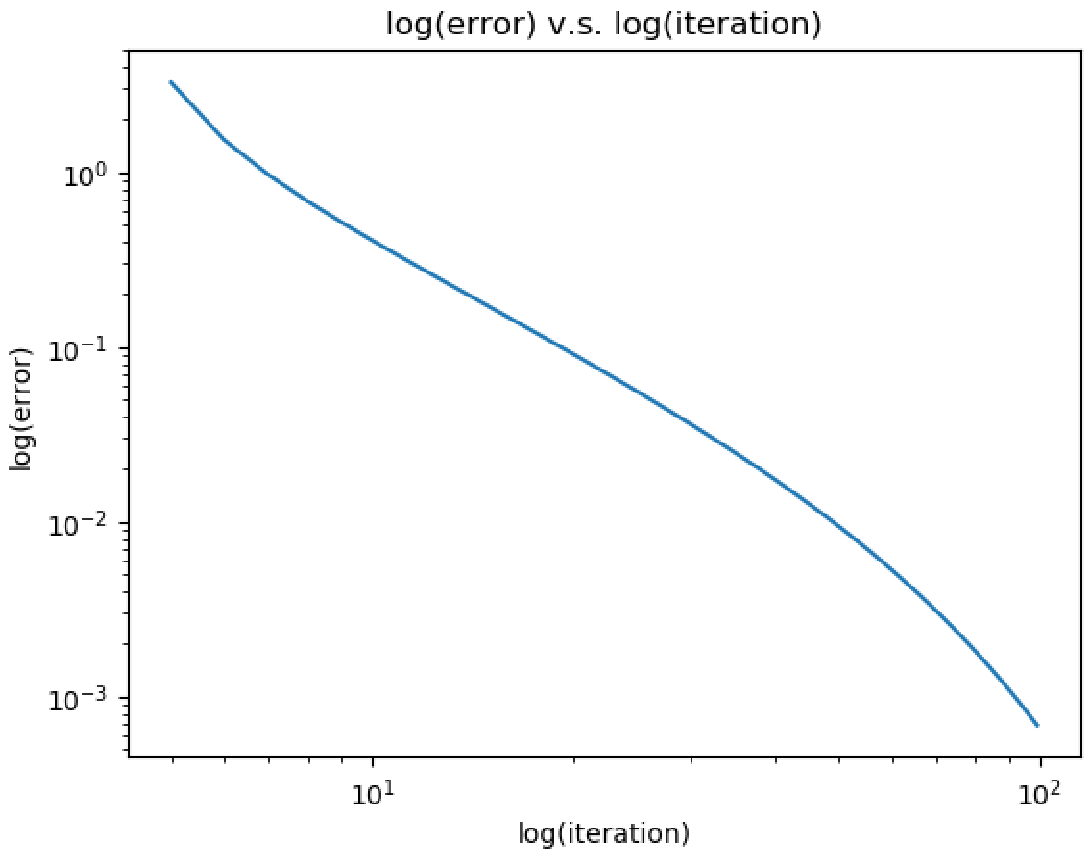
(4). The max difference v.s. iterations

Code:

```
def test_6():
    h = 0.1
    pp = np.zeros((11,11,11,101))
    for ndx in range(101):
        n = ndx
        for kdx in range(11):
            k = 0.5-kdx*h
            for jdx in range(11):
                j = 0.5-jdx*h
                for idx in range(11):
                    i = 0.5-idx*h
                    temp = phi_2(i,j,k,n,pp,h)
                    pp[idx, jdx, kdx, ndx] = temp
    deltamax = 0
    colmax = []

    for ndx in range(1,101):
        for jdx in range(11):
            for idx in range(11):
                for kdx in range(11):
                    if pp[idx,jdx,kdx,ndx]!=0 or pp[idx,jdx,kdx,ndx-1]!=0:
                        deltatemp = (pp[idx, jdx, kdx, ndx] - pp[idx, jdx, kdx, ndx - 1])/pp[idx,jdx,kdx,ndx-1]
                    if deltatemp > deltamax:
                        deltamax = deltatemp
        colmax.append(deltamax)
        deltamax = 0
    plt.plot(colmax)
    plt.title("error v.s. iteration")
    plt.xlabel("iteration")
    plt.ylabel("error")
    plt.show()
    plt.loglog(colmax)
    plt.title("log(error) v.s. log(iteration)")
    plt.xlabel("log(iteration)")
    plt.ylabel("log(error)")
    plt.show()
```

And the result is:





3. A box in a box

(1). Find E on xy-plane(z=0):

First, I create the function to calculate the phi.

Code:

```
import matplotlib.pyplot as plt
import numpy as np

h = 0.1 #space span
tau = pow(h,2)/6 #time span, I suppose tau = h^2/6

def phi(i,j,k,n,pp):
    t = n*tau
    xi = int(round((1.5 - i) / h))
    yj = int(round((1.5 - j) / h))
    zk = int(round((1.5 - k) / h))
    if i == 1.5 or i == -1.5 or j == 1.5 or j == -1.5 or k == 1.5 or k == -1.5: #outside the outer box
        ans = 0
    elif i == -0.5 and j <= 0.5 and j >= -0.5 and k <=0.5 and k >= -0.5: #inside box, I use poisson equation
        ans = h*h/6+(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/6
    elif i ==0.5 and j <= 0.5 and j >= -0.5 and k <=0.5 and k >= -0.5:#inside box, I use poisson equation
        ans = h*h/6+(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/6
    elif j == -0.5 and i <= 0.5 and i >= -0.5 and k <=0.5 and k >= -0.5:#inside box, I use poisson equation
        ans = h*h/6+(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/6
    elif j == 0.5 and i <= 0.5 and i >= -0.5 and k <=0.5 and k >= -0.5:#inside box, I use poisson equation
        ans = h*h/6+(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/6
    elif k == -0.5 and j <= 0.5 and j >= -0.5 and i <=0.5 and i >= -0.5:#inside box, I use poisson equation
        ans = h*h/6+(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/6
    elif k == 0.5 and j <= 0.5 and j >= -0.5 and i <=0.5 and i >= -0.5:#inside box, I use poisson equation
        ans = h*h/6+(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/6
    elif t==0: #initial condition
```

```
        ans = 0
    else: #I use the laplace equation
        ans =
(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/
6
    return ans
```

And then, I start to find E. The E equation is :

$$\mathrm{E} = -\nabla\phi$$

$$\vec{\mathrm{E}} = -\left(\frac{\partial\phi}{\partial x}\hat{x} + \frac{\partial\phi}{\partial y}\hat{y} + \frac{\partial\phi}{\partial z}\hat{z}\right)$$

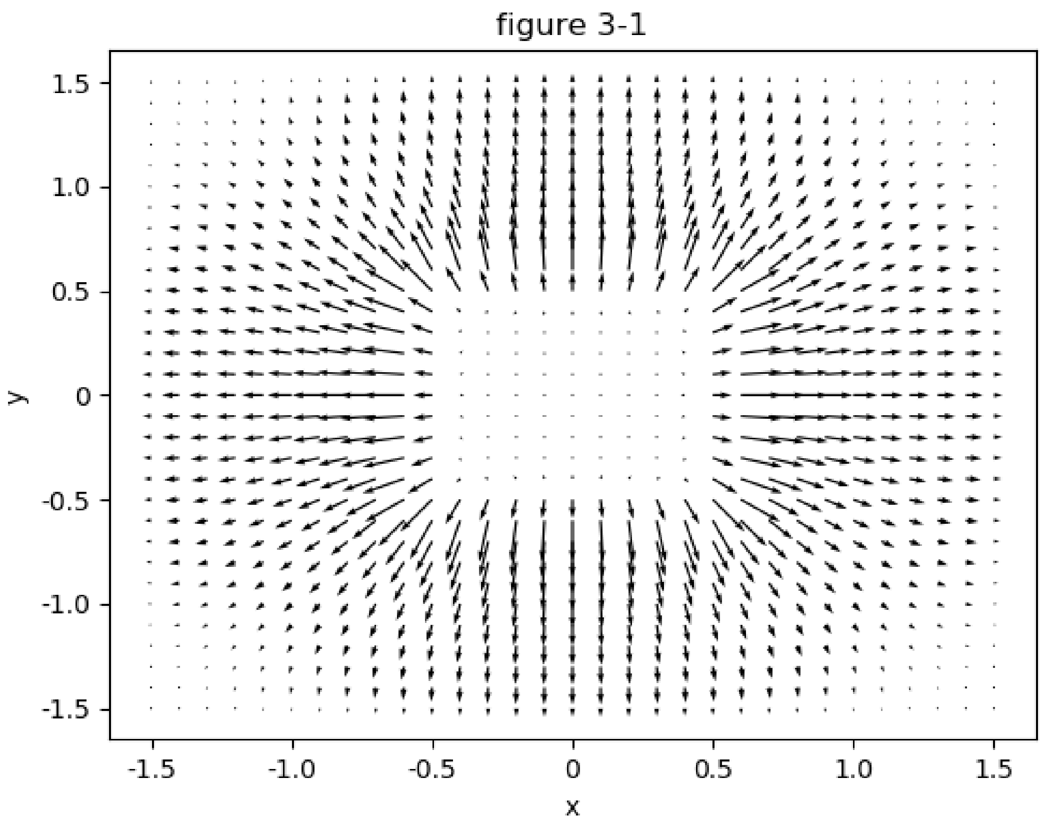
$$\vec{\mathrm{E}} = -(\frac{\phi_{i+1} - \phi_{i-1}}{2h}\hat{x} + \frac{\phi_{j+1} - \phi_{j-1}}{2h}\hat{y} + \frac{\phi_{k+1} - \phi_{k-1}}{2h}\hat{z})$$

Code:

```
def test_1():
    pp = np.zeros((31,31,31,1001))
    ex = np.zeros((31,31))
    ey = np.zeros((31,31))
    for ndx in range(1001):#1000 time steps
        n = ndx
        for kdx in range(31):#30 space steps
            k = 1.5-kdx*h
            for jdx in range(31):#30 space steps
                j = 1.5-jdx*h
                for idx in range(31):#30 space steps
                    i = 1.5-idx*h
                    temp = phi(i,j,k,n,pp)
                    pp[idx, jdx, kdx, ndx] = temp

    #Fined the E
    for idx in range(31):
        for jdx in range(31):
            if idx == 0 or jdx == 0:
                if jdx >= 30 or idx >= 30: #outside the outer box
                    ex[idx, jdx] = 0
                    ey[idx, jdx] = 0
                else:
                    ex[idx,jdx] = -(pp[idx+1,jdx,15,1000])/(2*h)
                    ey[idx,jdx] = -(pp[idx,jdx+1,15,1000])/(2*h)
            elif jdx >= 30 or idx >= 30:
                if idx == 0 or jdx == 0:
                    ex[idx, jdx] = 0
```

```
                ey[idx, jdx] = 0
            else:
                ex[idx,jdx] = -(-pp[idx-1,jdx,15,1000])/(2*h)
                ey[idx,jdx] = -(-pp[idx,jdx-1,15,1000])/(2*h)
        else:
            ex[idx,jdx] = -(pp[idx+1,jdx,15,1000]-pp[idx-1,jdx,15,1000])/(2*h)
            ey[idx,jdx] = -(pp[idx,jdx+1,15,1000]-pp[idx,jdx-1,15,1000])/(2*h)

plt.quiver(ey,ex)
plt.xticks([0,5,10,15,20,25,30],[-1.5,-1.0,-0.5,0,0.5,1.0,1.5])
plt.yticks([0,5,10,15,20,25,30],[-1.5,-1.0,-0.5,0,0.5,1.0,1.5])
plt.title("figure 3-1")
plt.show()
```

And the result:



We can see that the space between two boxes, the electric field flows from inner to outer, and all symmetric at origin.

(2).to find charge distribution:

The equation of charge distribution is:

$$\nabla \cdot \vec{\mathrm{E}} = \frac{\rho}{\varepsilon_0}$$

For $\rho = \frac{q}{V}$ is charge density, and we suppose $\varepsilon_0 = 1$. So, the equation will become this:

$$\frac{\partial \vec{\mathrm{E}}}{\partial x} + \frac{\partial \vec{E}}{\partial y} + \frac{\partial \vec{E}}{\partial z} = \frac{q}{V}$$

$$\frac{\mathrm{E}_{i+1} - E_{i-1}}{2h} + \frac{E_{j+1} - E_{j-1}}{2h} + \frac{E_{k+1} - E_{k-1}}{2h} = \frac{q}{V}$$

Code:

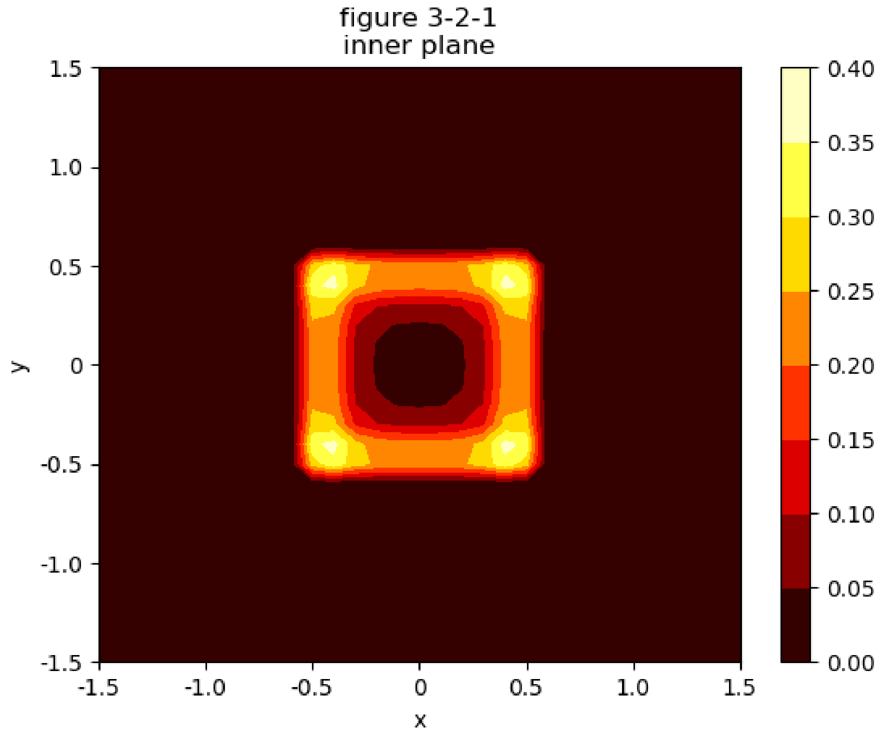
```
def test_2():
    pp = np.zeros((31, 31, 31, 1001))
    ex = np.zeros((31, 31))
    ey = np.zeros((31, 31))
    exo = np.zeros((31, 31))
    eyo = np.zeros((31, 31))
    qi = np.zeros((31,31))
    qo = np.zeros((31,31))
    #calculate electric potential
    for ndx in range(101):
        n = ndx
        for kdx in range(31):
            k = 1.5 - kdx * h
            for jdx in range(31):
                j = 1.5 - jdx * h
                for idx in range(31):
                    i = 1.5 - idx * h
                    temp = phi(i, j, k, n, pp)
                    pp[idx, jdx, kdx, ndx] = temp
    # calculate inner plane electric field
    for idx in range(31):
        for jdx in range(31):
            if idx == 0 or jdx == 0:
                if jdx >= 30 or idx >= 30:
                    ex[idx, jdx] = 0
                    ey[idx, jdx] = 0
                else:
                    ex[idx,jdx] = -(pp[idx+1,jdx,10,100])/(2*h)
                    ey[idx,jdx] = -(pp[idx,jdx+1,10,100])/(2*h)
            elif jdx >= 30 or idx >= 30:
                if idx == 0 or jdx == 0:
```

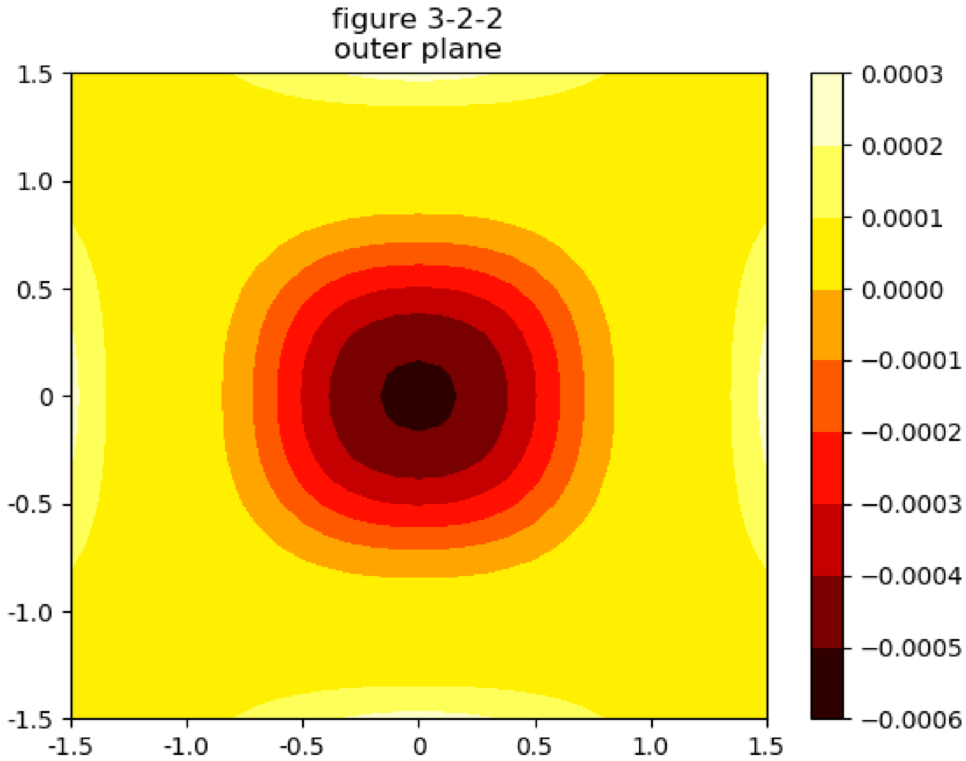
```
                ex[idx, jdx] = 0
                ey[idx, jdx] = 0
            else:
                ex[idx,jdx] = -(-pp[idx-1,jdx,10,100])/(2*h)
                ey[idx,jdx] = -(-pp[idx,jdx-1,10,100])/(2*h)
        else:
            ex[idx,jdx] = -(pp[idx+1,jdx,10,100]-pp[idx-1,jdx,10,100])/(2*h)
            ey[idx,jdx] = -(pp[idx,jdx+1,10,100]-pp[idx,jdx-1,10,100])/(2*h)
# calculate outer plane electric field
for idx in range(31):
    for jdx in range(31):
        if idx == 0 or jdx == 0:
            if jdx >= 30 or idx >= 30:
                exo[idx, jdx] = 0
                eyo[idx, jdx] = 0
            else:
                exo[idx,jdx] = -(pp[idx+1,jdx,1,99])/(2*h*6)
                eyo[idx,jdx] = -(pp[idx,jdx+1,1,99])/(2*h*6)
        elif jdx >= 30 or idx >= 30:
            if idx == 0 or jdx == 0:
                exo[idx, jdx] = 0
                eyo[idx, jdx] = 0
            else:
                exo[idx,jdx] = -(-pp[idx-1,jdx,1,99])/(2*h*6)
                eyo[idx,jdx] = -(-pp[idx,jdx-1,1,99])/(2*h*6)
        else:
            exo[idx,jdx] = -(pp[idx+1,jdx,1,99]-pp[idx-1,jdx,1,99])/(2*h*6)
            eyo[idx,jdx] = -(pp[idx,jdx+1,1,99]-pp[idx,jdx-1,1,99])/(2*h*6)

# calculate inner plane charge distribution
for idx in range(10,21):# the inner plane only on 1 length unit long
    for jdx in range(10,21):
            qi[idx,jdx] = ((ex[idx+1,jdx]-ex[idx-1,jdx])/(2*h))+((ey[idx,jdx+1]-ey[idx,jdx-1])/(2*h))
# calculate outer plane charge distribution
for idx in range(31):
    for jdx in range(31):
        if idx == 0 :
            if jdx >= 30:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            elif jdx ==0:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
```

```
                    (eyo[idx, jdx + 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
        elif jdx == 0:
            if idx >= 30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
        elif idx >= 30:
            if jdx == 0:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
            elif jdx >= 30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
        elif jdx >= 30:
            if idx == 0:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        ( - eyo[idx, jdx - 1]) / (2 * h))
            elif idx>=30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
        else:
            qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
bb = plt.contourf(qi, cmap=plt.cm.hot)
plt.colorbar(bb,orientation = "vertical")
plt.xticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.yticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.title("figure 3-2-1\ninner plane")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

```
aa = plt.contourf(qo, cmap = plt.cm.hot)
plt.colorbar(aa, orientation="vertical")
plt.xticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.yticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.title("figure 3-2-2\nouter plane")
plt.show()
```

and the result:





We can see the figure on the inner plane, the charge will more on the edge of plane, especially on the corner. On the outer plane, we can see that the charge will be crowed on the center for the projection of inner box.

(3). Find E on xy-plane(z=0):

This time, we move the inner box to place at a distance of 0.3 from one of the sides of the outer box. First, I need to create a function to calculate phi, and I move the box to (0.5~-0.5, -0.2~-1.2,0.5~-0.5).

Code:

```
def phi_2(i,j,k,n,pp):
    t = n*tau
    xi =int(round((1.5-i)/h))
    yj =int(round((1.5-j)/h))
    zk =int(round((1.5-k)/h))
    if i == 1.5 or i == -1.5 or j == 1.5 or j == -1.5 or k == 1.5 or k == -1.5:
        ans = 0
    elif i == 1.2 and j <= 0.5 and j >= -0.5 and k <=0.5 and k >= -0.5:
        ans = h * h / 6 + (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
            xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
    elif xi == 13 and j <= 0.5 and j >= -0.5 and k <=0.5 and k >= -0.5:
        ans = h * h / 6 + (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
            xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
    elif j == -0.5 and i <= 1.2 and i >= 0.2 and k <=0.5 and k >= -0.5:
        ans = h * h / 6 + (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
            xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
    elif j == 0.5 and i <= 1.2 and i >= 0.2 and k <=0.5 and k >= -0.5:
        ans = h * h / 6 + (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
            xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
    elif k == -0.5 and j <= 0.5 and j >= -0.5 and i <=1.2 and i >= -0.2:
        ans = h * h / 6 + (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
            xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
    elif k == 0.5 and j <= 0.5 and j >= -0.5 and i <=1.2 and i >= 0.2:
        ans = h * h / 6 + (pp[xi + 1, yj, zk, n - 1] + pp[xi - 1, yj, zk, n - 1] + pp[xi, yj + 1, zk, n - 1] + pp[
            xi, yj - 1, zk, n - 1] + pp[xi, yj, zk + 1, n - 1] + pp[xi, yj, zk - 1, n - 1]) / 6
    elif t==0:
        ans = 0
    else:
        ans =
(pp[xi+1,yj,zk,n-1]+pp[xi-1,yj,zk,n-1]+pp[xi,yj+1,zk,n-1]+pp[xi,yj-1,zk,n-1]+pp[xi,yj,zk+1,n-1]+pp[xi,yj,zk-1,n-1])/
6
    return ans
```

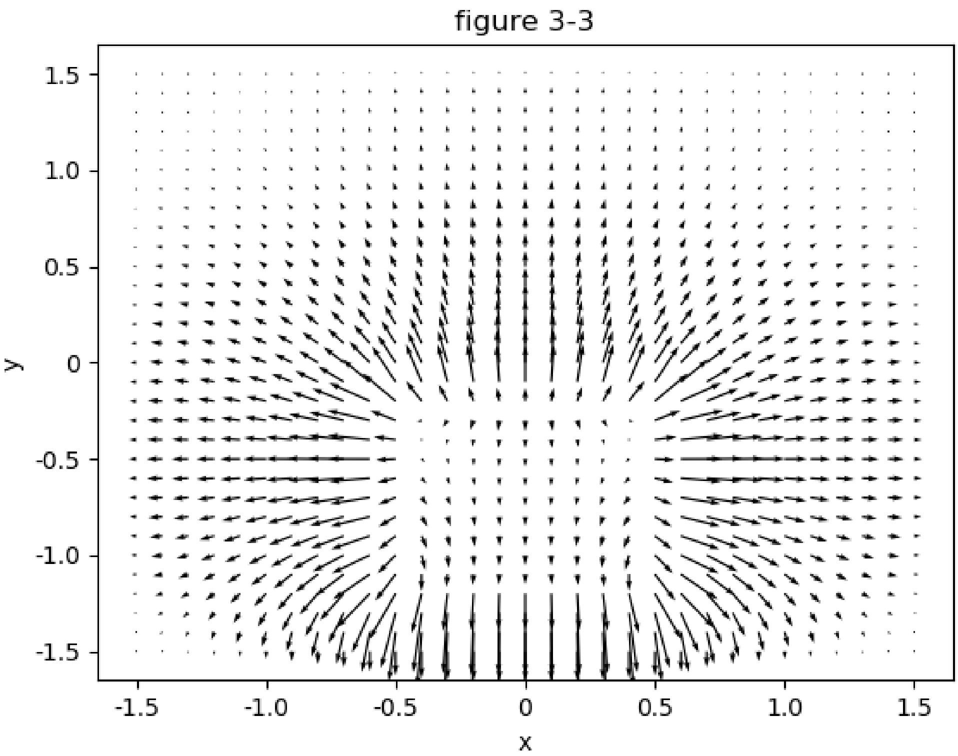
and I start to find E.

Code:

```
def test_3():
    pp = np.zeros((31,31,31,1001))
    ex = np.zeros((31,31))
    ey = np.zeros((31,31))
    for ndx in range(1001):
        n = ndx
        for kdx in range(31):
            k = 1.5 - kdx * h
            for jdx in range(31):
                j = 1.5 - jdx * h
                for idx in range(31):
                    i = 1.5 - idx * h
                    temp = phi_2(i, j, k, n, pp)
                    pp[idx, jdx, kdx, ndx] = temp


    for idx in range(31):
        for jdx in range(31):
            if idx == 0 or jdx == 0:
                if jdx >= 30 or idx >= 30:
                    ex[idx, jdx] = 0
                    ey[idx, jdx] = 0
                else:
                    ex[idx,jdx] = -(pp[idx+1,jdx,15,1000])/(2*h)
                    ey[idx,jdx] = -(pp[idx,jdx+1,15,1000])/(2*h)
            elif jdx >= 30 or idx >= 30:
                if idx == 0 or jdx == 0:
                    ex[idx, jdx] = 0
                    ey[idx, jdx] = 0
                else:
                    ex[idx,jdx] = -(-pp[idx-1,jdx,15,1000])/(2*h)
                    ey[idx,jdx] = -(-pp[idx,jdx-1,15,1000])/(2*h)
            else:
                ex[idx,jdx] = -(pp[idx+1,jdx,15,1000]-pp[idx-1,jdx,15,1000])/(2*h)
                ey[idx,jdx] = -(pp[idx,jdx+1,15,1000]-pp[idx,jdx-1,15,1000])/(2*h)

    plt.quiver(ey,ex)
    plt.xticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
    plt.yticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
    plt.title("figure 3-3")
```

```
    plt.ylabel("y")
    plt.xlabel("x")
    plt.show()
```

the result:



Because the inner box is not at the center anymore, we can see that the electric field is stronger when the plane is closer to outer box. Also, we can see inside the inner box, the electric field is not symmetry anymore.

(4).charge distribution

Code:

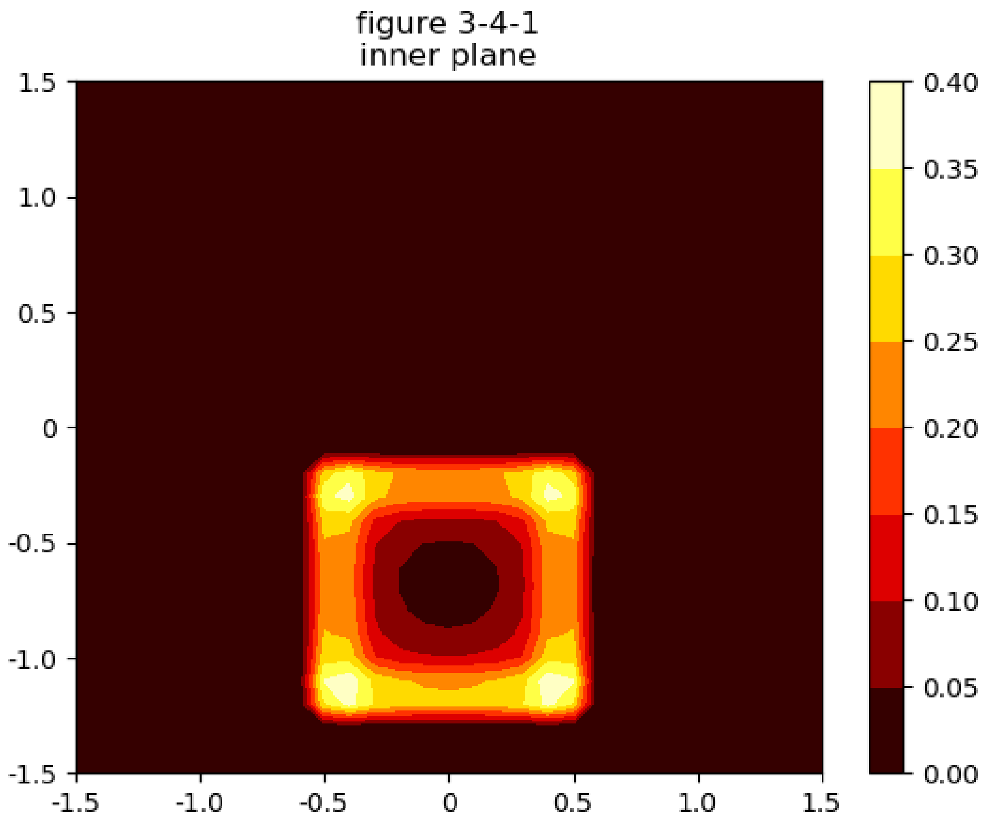
```
def test_4():
    pp = np.zeros((31, 31, 31, 101))
    ex = np.zeros((31, 31))
    ey = np.zeros((31, 31))
    exo = np.zeros((31, 31))
    eyo = np.zeros((31, 31))
    qi = np.zeros((31,31))
    qo = np.zeros((31,31))
    #calculate electric potential
    for ndx in range(101):
        n = ndx
        for kdx in range(31):
            k = 1.5 - kdx * h
```

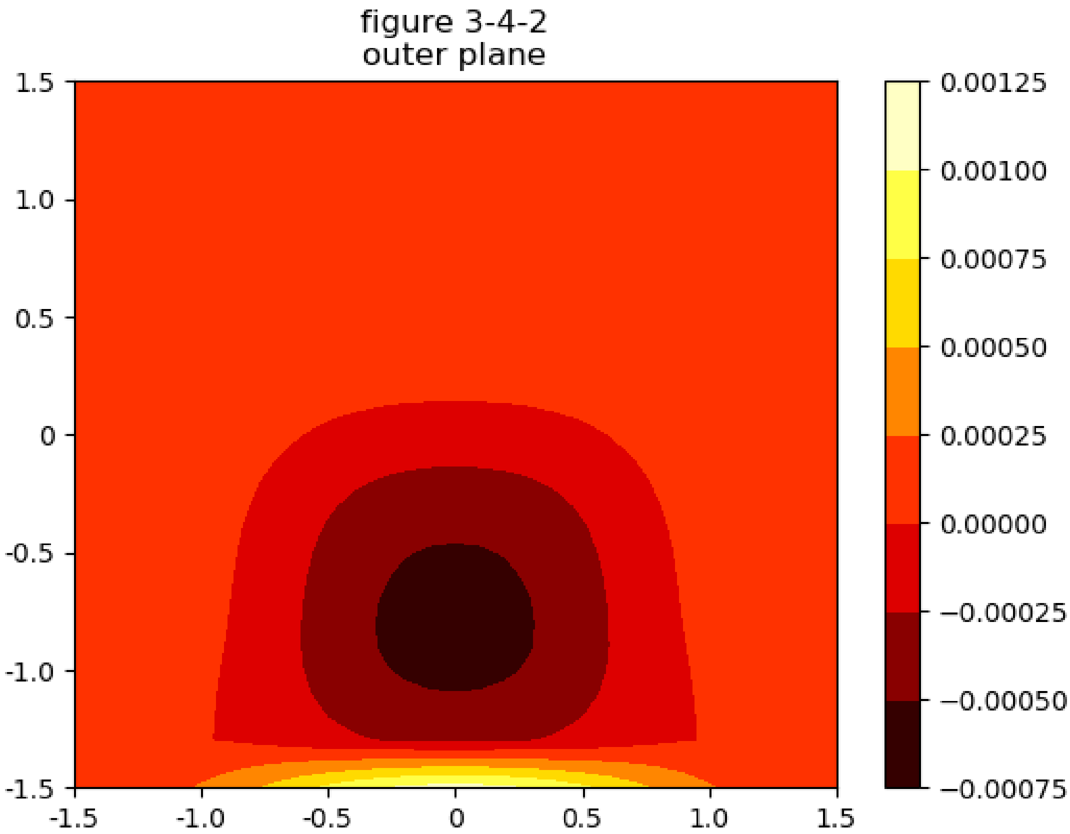
```
        for jdx in range(31):
            j = 1.5 - jdx * h
            for idx in range(31):
                i = 1.5 - idx * h
                temp = phi_2(i, j, k, n, pp)
                pp[idx, jdx, kdx, ndx] = temp
# calculate inner box electric field
for idx in range(31):
    for jdx in range(31):
        if idx == 0 or jdx == 0:
            if jdx >= 30 or idx >= 30:
                ex[idx, jdx] = 0
                ey[idx, jdx] = 0
            else:
                ex[idx,jdx] = -(pp[idx+1,jdx,10,100])/(2*h)
                ey[idx,jdx] = -(pp[idx,jdx+1,10,100])/(2*h)
        elif jdx >= 30 or idx >= 30:
            if idx == 0 or jdx == 0:
                ex[idx, jdx] = 0
                ey[idx, jdx] = 0
            else:
                ex[idx,jdx] = -(-pp[idx-1,jdx,10,100])/(2*h)
                ey[idx,jdx] = -(-pp[idx,jdx-1,10,100])/(2*h)
        else:
            ex[idx,jdx] = -(pp[idx+1,jdx,10,100]-pp[idx-1,jdx,10,100])/(2*h)
            ey[idx,jdx] = -(pp[idx,jdx+1,10,100]-pp[idx,jdx-1,10,100])/(2*h)
# calculate outer box electric field
for idx in range(31):
    for jdx in range(31):
        if idx == 0 or jdx == 0:
            if jdx >= 30 or idx >= 30:
                exo[idx, jdx] = 0
                eyo[idx, jdx] = 0
            else:
                exo[idx,jdx] = -(pp[idx+1,jdx,1,99])/(2*h*6)
                eyo[idx,jdx] = -(pp[idx,jdx+1,1,99])/(2*h*6)
        elif jdx >= 30 or idx >= 30:
            if idx == 0 or jdx == 0:
                exo[idx, jdx] = 0
                eyo[idx, jdx] = 0
            else:
                exo[idx,jdx] = -(-pp[idx-1,jdx,1,99])/(2*h*6)
```

```
                eyo[idx,jdx] = -(-pp[idx,jdx-1,1,99])/(2*h*6)
        else:
            exo[idx,jdx] = -(pp[idx+1,jdx,1,99]-pp[idx-1,jdx,1,99])/(2*h*6)
            eyo[idx,jdx] = -(pp[idx,jdx+1,1,99]-pp[idx,jdx-1,1,99])/(2*h*6)
# calculate inner box charge distribution
for idx in range(3,14):
    for jdx in range(10,21):
            qi[idx,jdx] = ((ex[idx+1,jdx]-ex[idx-1,jdx])/(2*h))+((ey[idx,jdx+1]-ey[idx,jdx-1])/(2*h))
# calculate outer box charge distribution
for idx in range(31):
    for jdx in range(31):
        if idx == 0 :
            if jdx >= 30:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            elif jdx ==0:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
        elif jdx == 0:
            if idx >= 30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
        elif idx >= 30:
            if jdx == 0:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
            elif jdx >= 30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
        elif jdx >= 30:
            if idx == 0:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        ( - eyo[idx, jdx - 1]) / (2 * h))
```

```
            elif idx>=30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
        else:
            qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
bb = plt.contourf(qi, cmap=plt.cm.hot)
plt.colorbar(bb, orientation="vertical")
plt.xticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.yticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.title("figure 3-4-1\ninner plane")
plt.show()
aa = plt.contourf(-qo, cmap=plt.cm.hot)
plt.colorbar(aa, orientation="vertical")
plt.xticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.yticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.title("figure 3-4-2\nouter plane")
plt.show()
```

The result:





These are just the same as previous one, but we can see that the outer plane closer to inner box, the electric field is stronger than others are.

(5).

I want to see what will happen on the closest plane on inner and outer box, so I find charge distribution again.

Code:

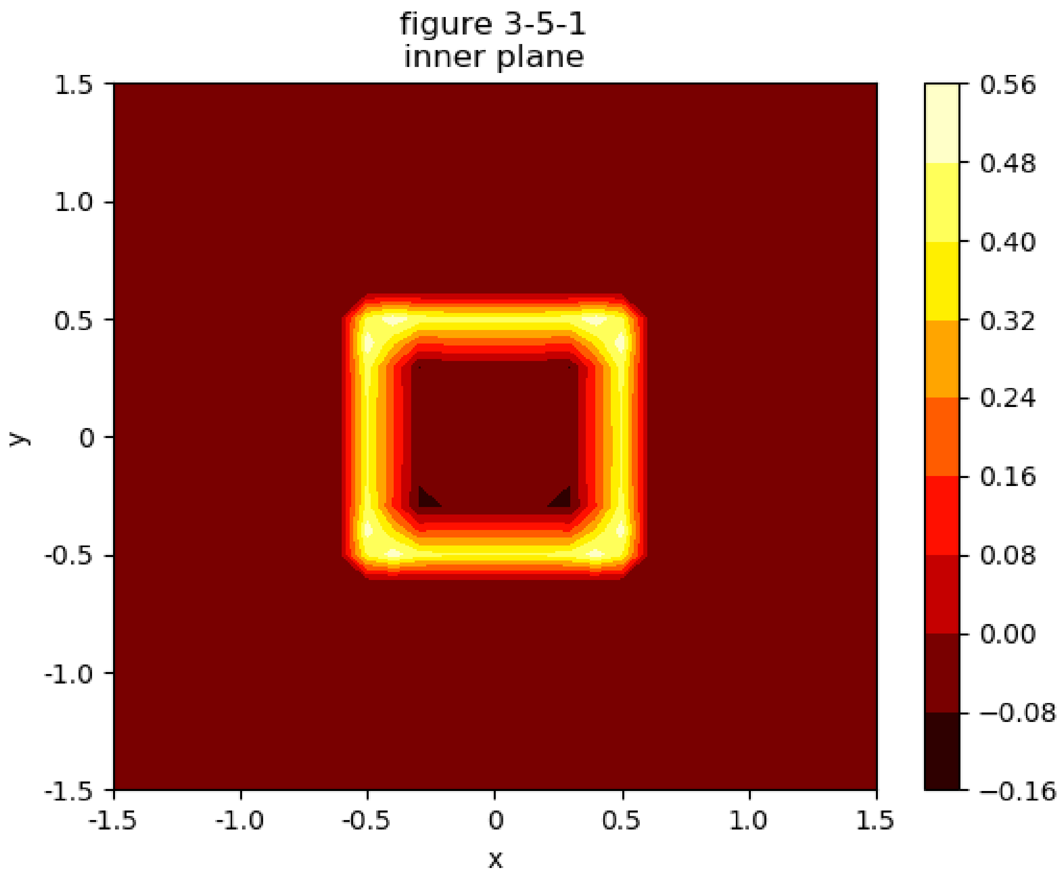
```
def test_5():
    pp = np.zeros((31, 31, 31, 1001))
    ex = np.zeros((31, 31))
    ey = np.zeros((31, 31))
    exo = np.zeros((31, 31))
    eyo = np.zeros((31, 31))
    qi = np.zeros((31,31))
    qo = np.zeros((31,31))
    for ndx in range(101):
        n = ndx
        for kdx in range(31):
            k = 1.5 - kdx * h
            for jdx in range(31):
                j = 1.5 - jdx * h
                for idx in range(31):
```

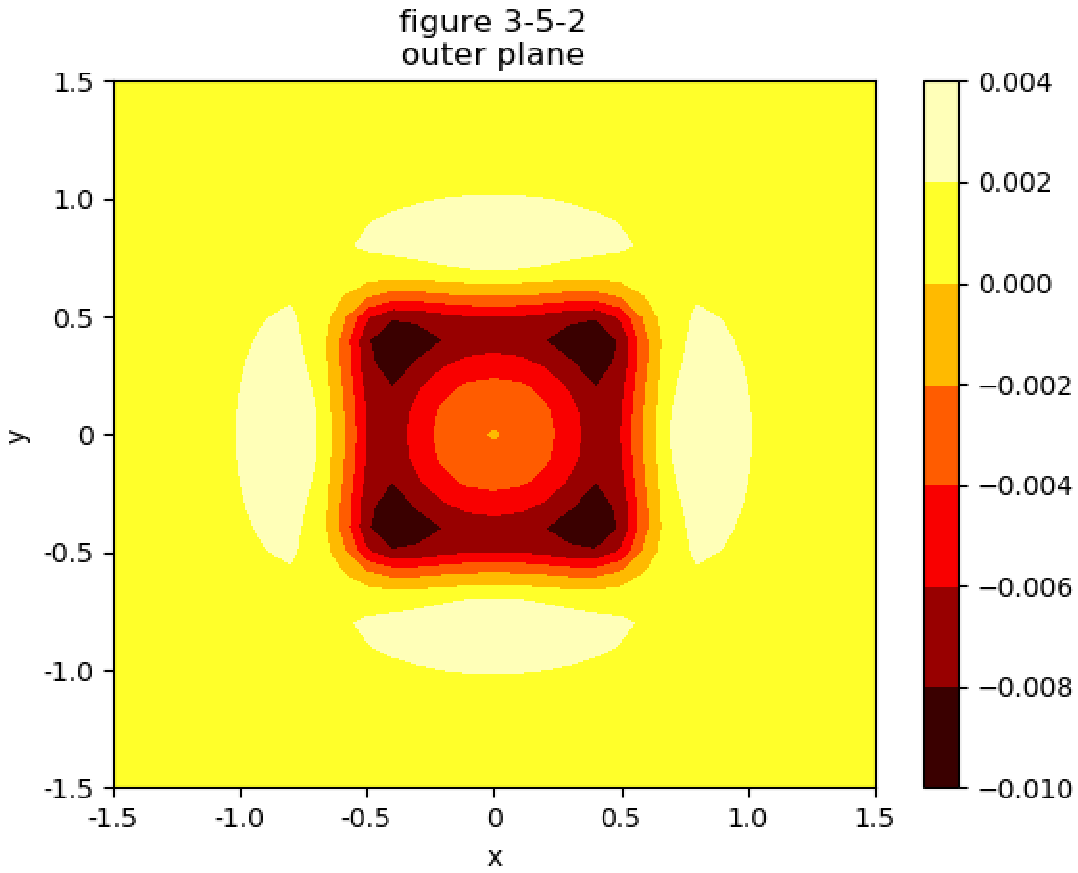
```
                i = 1.5 - idx * h
                temp = phi_2(i, j, k, n, pp)
                pp[idx, jdx, kdx, ndx] = temp



for idx in range(31):
    for jdx in range(31):
        if idx == 0 or jdx == 0:
            if jdx >= 30 or idx >= 30:
                ex[idx, jdx] = 0
                ey[idx, jdx] = 0
            else:
                ex[idx,jdx] = -(pp[10,jdx,idx+1,100])/(2*h)
                ey[idx,jdx] = -(pp[10,jdx+1,idx,100])/(2*h)
        elif jdx >= 30 or idx >= 30:
            if idx == 0 or jdx == 0:
                ex[idx, jdx] = 0
                ey[idx, jdx] = 0
            else:
                ex[idx,jdx] = -(-pp[10,jdx,idx-1,100])/(2*h)
                ey[idx,jdx] = -(-pp[10,jdx-1,idx,100])/(2*h)
        else:
            ex[idx,jdx] = -(pp[10,jdx,idx+1,100]-pp[10,jdx,idx-1,100])/(2*h)
            ey[idx,jdx] = -(pp[10,jdx+1,idx,100]-pp[10,jdx-1,idx,100])/(2*h)

for idx in range(31):
    for jdx in range(31):
        if idx == 0 or jdx == 0:
            if jdx >= 30 or idx >= 30:
                exo[idx, jdx] = 0
                eyo[idx, jdx] = 0
            else:
                exo[idx,jdx] = -(pp[1,jdx,idx+1,99])/(2*h*6)
                eyo[idx,jdx] = -(pp[1,jdx+1,idx,99])/(2*h*6)
        elif jdx >= 30 or idx >= 30:
            if idx == 0 or jdx == 0:
                exo[idx, jdx] = 0
                eyo[idx, jdx] = 0
            else:
                exo[idx,jdx] = -(-pp[1,jdx,idx-1,99])/(2*h*6)
                eyo[idx,jdx] = -(-pp[1,jdx-1,idx,99])/(2*h*6)
        else:
```

```python
            exo[idx,jdx] = -(pp[1,jdx,idx+1,99]-pp[1,jdx,idx-1,99])/(2*h*6)
            eyo[idx,jdx] = -(pp[1,jdx+1,idx,99]-pp[1,jdx-1,idx,99])/(2*h*6)




for idx in range(10,21):
    for jdx in range(10,21):
            qi[idx,jdx] = ((ex[idx+1,jdx]-ex[idx-1,jdx])/(2*h))+((ey[idx,jdx+1]-ey[idx,jdx-1])/(2*h))

for idx in range(31):
    for jdx in range(31):
        if idx == 0 :
            if jdx >= 30:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            elif jdx ==0:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
        elif jdx == 0:
            if idx >= 30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
        elif idx >= 30:
            if jdx == 0:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1]) / (2 * h))
            elif jdx >= 30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
        elif jdx >= 30:
            if idx == 0:
                qo[idx, jdx] = ((exo[idx + 1, jdx]) / (2 * h)) + (
                        ( - eyo[idx, jdx - 1]) / (2 * h))
```

```
            elif idx>=30:
                qo[idx, jdx] = ((- exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
            else:
                qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                        (- eyo[idx, jdx - 1]) / (2 * h))
        else:
            qo[idx, jdx] = ((exo[idx + 1, jdx] - exo[idx - 1, jdx]) / (2 * h)) + (
                (eyo[idx, jdx + 1] - eyo[idx, jdx - 1]) / (2 * h))
bb = plt.contourf(qi, cmap=plt.cm.hot)
plt.colorbar(bb, orientation="vertical")
plt.xticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.yticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.title("figure 3-5-1\ninner plane")
plt.show()
aa = plt.contourf(qo, cmap=plt.cm.hot)
plt.colorbar(aa, orientation="vertical")
plt.xticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.yticks([0, 5, 10, 15, 20, 25, 30], [-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5])
plt.title("figure 3-5-2\nouter plane")
plt.show()
```

The result:





The result is the same as (2). But the charge is more stronger than ever, and the outer plane is much clear to see the projection of inner box.