2-D Particle in the Box (Laplace)

January 11, 2018

In [1]:
```
import numpy as np
import matplotlib.pyplot as pt
import numpy.linalg as la
import scipy.sparse.linalg as lin
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

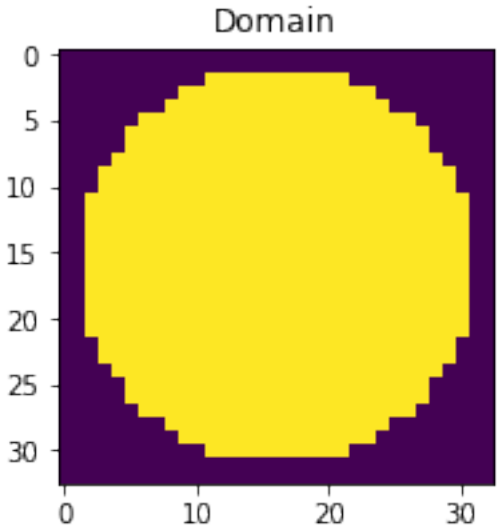
In [15]:
```
# dimension of solution vector
N=30 # choose EVEN number for square and ODD number for disk to ensure accuracy
location=np.zeros((N,N))

for i in range(0,N):      # label the position of each element in matrix
    for j in range(0,N):
        location[i,j]=j+N*i
```

In [16]:
```
def matrix_to_index(a,b): # get index from matrix(i,j)
    return int(location[a,b])

def index_to_i(a):        # find x_coord of 'a' in matrix
    ind_1=np.where(location==a)[0][0]
    return ind_1

def index_to_j(a):        # find y_coord of 'a' in matrix
    ind_1=np.where(location==a)[1][0]
    return ind_1
```

In [17]:
```
# define boundary of solving equation by setting grids in domain to 1.
# examples:

# 1) square
# indomain=np.ones((N+2,N+2))
# indomain[0]=0
# indomain[len(indomain)-1]=0
# indomain[:,0]=0
# indomain[:,len(indomain)-1]=0

# 2) disk
indomain=np.zeros((N+3,N+3))
```

```
        for i in range(N):
            for j in range(N):
                if np.sqrt((i-N/2)**2+(j-N/2)**2)<N/2:
                    indomain[i+1,j+1]=1

        # if a coordinate is in the domain of solution?
        def if_in_domain(a,b): # if in: 1 ; if not: 0
            return int(indomain[a+1,b+1])
```

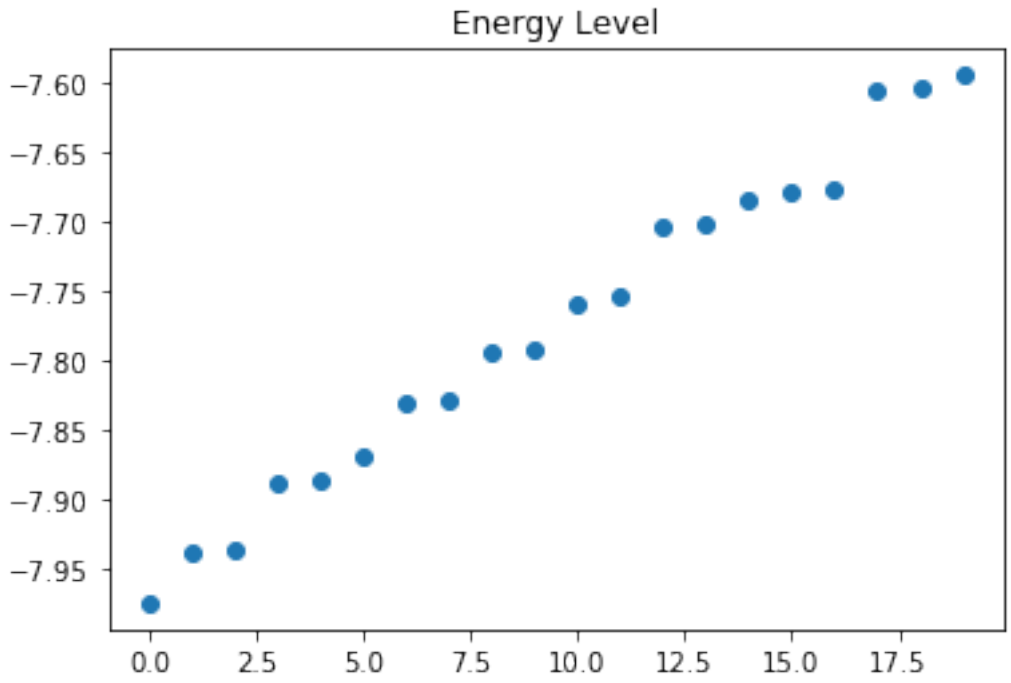
In [18]:
```
        pt.figure(figsize=(3,3))
        pt.imshow(indomain)
        pt.title('Domain') # domain is in yellow and the canvas is black
        pt.show()
```



In [19]:
```
        # constructing Hamiltonian: H = -p^2/2m, V = 0
        # after simplification: -4 u_i,j + u_i,j+1 + u_i,j-1 + u_i+1,j + u_i-1,j

        H=np.zeros((N**2,N**2))

        for i in range(0,N**2):
            x=index_to_i(i)
            y=index_to_j(i)

            if if_in_domain(x,y)==1: # check every time if it is in domain
                H[i, matrix_to_index(x,y)] = -4

            if if_in_domain(x-1,y)==1:
                H[i, matrix_to_index(x-1,y)] = 1
```

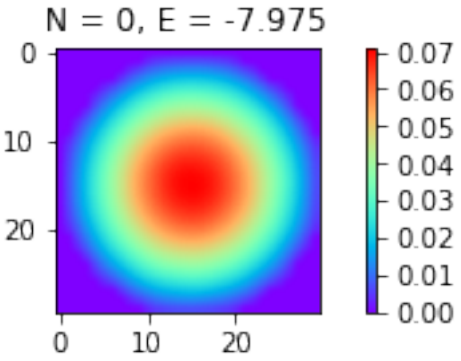
```
            if if_in_domain(x+1,y)==1:
                H[i, matrix_to_index(x+1,y)] = 1

            if if_in_domain(x,y+1)==1:
                H[i, matrix_to_index(x,y+1)] = 1

            if if_in_domain(x,y-1)==1:
                H[i, matrix_to_index(x,y-1)] = 1
        # H = -H   # convergence problem???
        vals, vec = lin.eigsh(H,k=20,which='SA') # Smallest (magnitude) eigenvalues
```

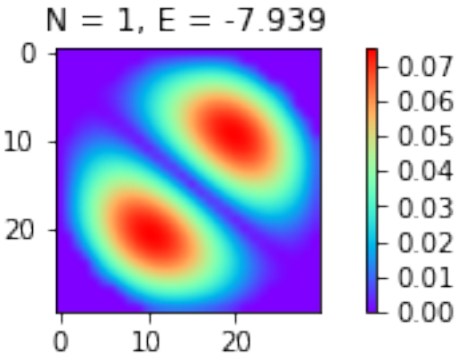
In [20]:
```
# Energy Degeneracy
pt.plot(vals[:20],'o')
pt.title('Energy Level')
pt.show()
```

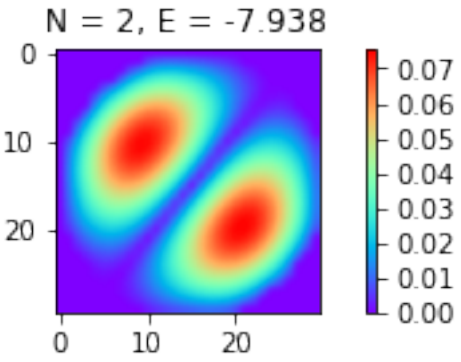


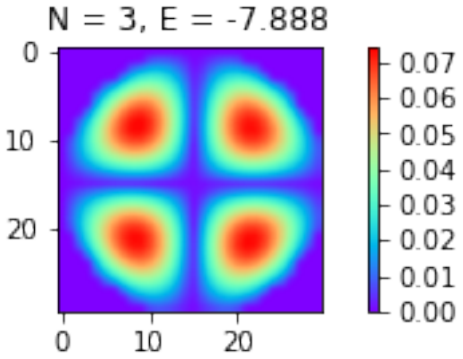
In [23]:
```
# plotting different modes of excitations
# eigenvectors are extracted by COLUMNS: la.eigh(H)[0][:,i]

pt.figure()
for j in range(0,20):
    eig=vec[:,j]
    solu=np.zeros((N,N))

    for i in range(0,len(eig)):
```

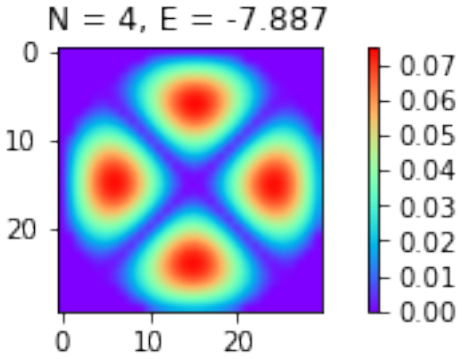
```
        solu[index_to_i(i),index_to_j(i)]=eig[i]
        if (if_in_domain(index_to_i(i),index_to_j(i)) == 0):
            solu[index_to_i(i),index_to_j(i)] = 0

pt.subplot(211)
pt.imshow(abs(solu),cmap='rainbow',interpolation='gaussian') # plotting probability
pt.title('N = '+str(j)+', E = '+str(np.round(vals[j],3)))
pt.colorbar()
pt.show()
```

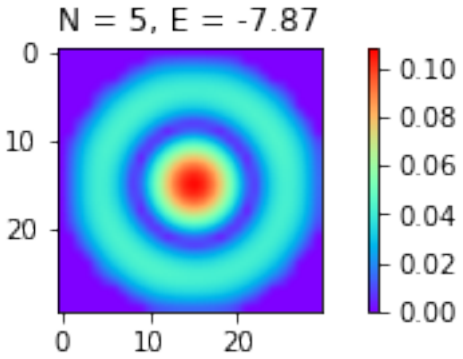


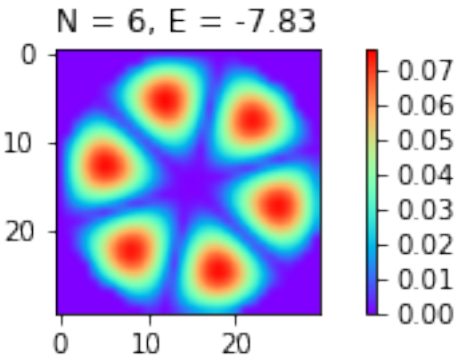


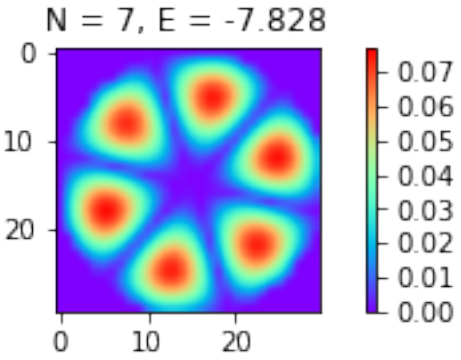


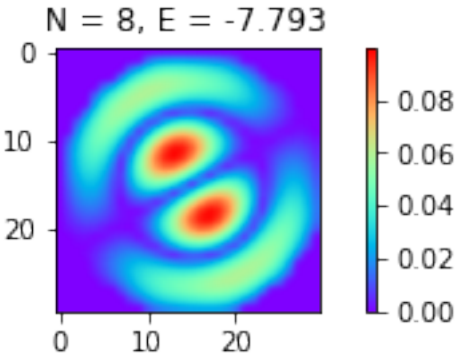

N = 3, E = -7.888
0
10
20
0
10
20
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00

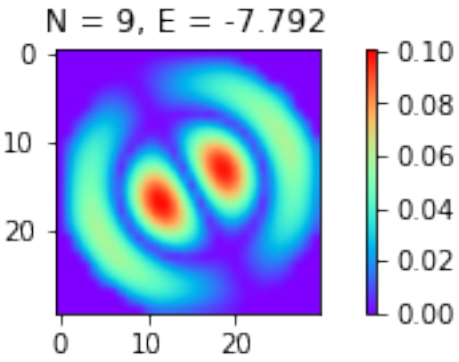

N = 4, E = -7.887
0
10
20
0
10
20
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00

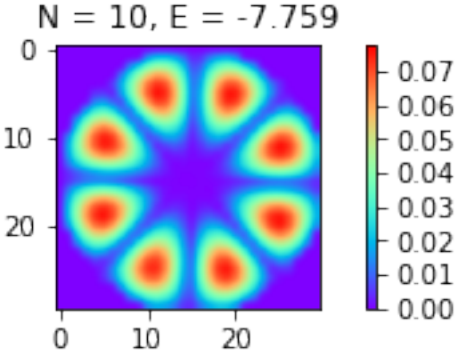

N = 5, E = -7.87
0
10
20
0
10
20
0.10
0.08
0.06
0.04
0.02
0.00

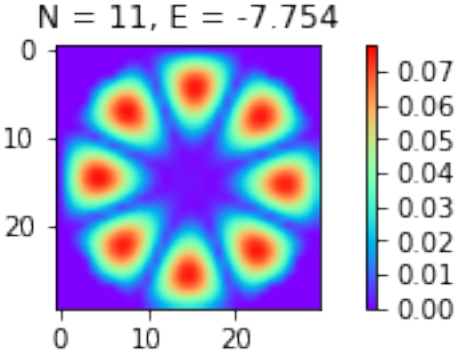

N = 6, E = -7.83
0
10
20
0
10
20
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00

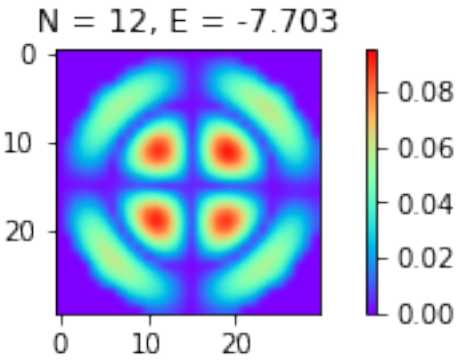

N = 7, E = -7.828
0
10
20
0
10
20
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00

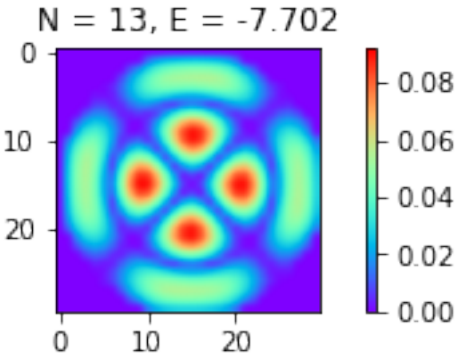

N = 8, E = -7.793
0
10
20
0
10
20
0.08
0.06
0.04
0.02
0.00

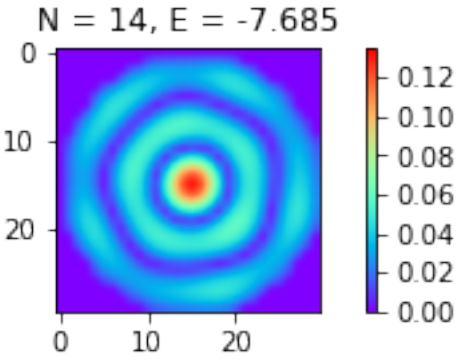

N = 9, E = -7.792
0
10
20
0
10
20
0.10
0.08
0.06
0.04
0.02
0.00

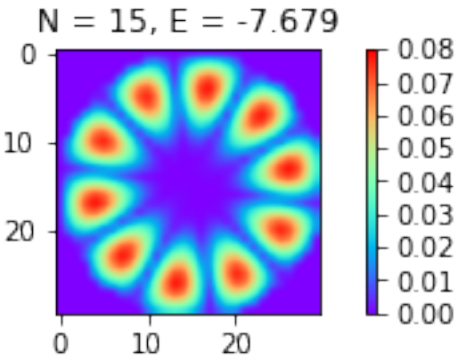

N = 10, E = -7.759
0
10
20
0
10
20
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00

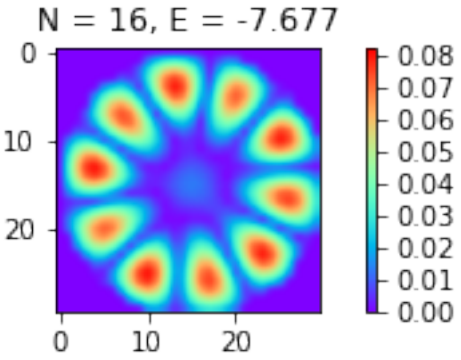

N = 11, E = -7.754
0
10
20
0
10
20
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00

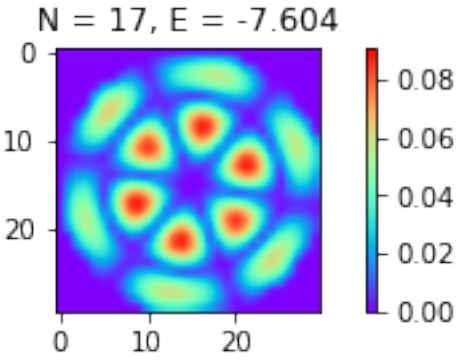

N = 12, E = -7.703
0
10
20
0
10
20
0.08
0.06
0.04
0.02
0.00

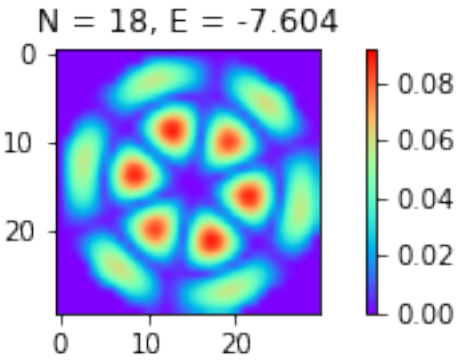

N = 13, E = -7.702
0
10
20
0
10
20
0.08
0.06
0.04
0.02
0.00

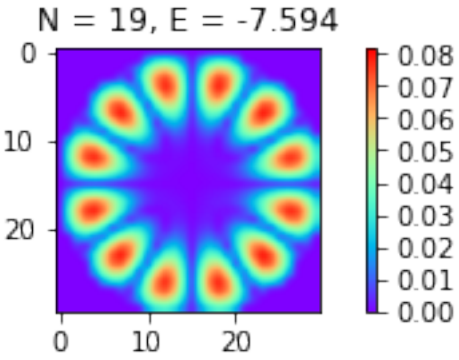

N = 14, E = -7.685
0
10
20
0
10
20
0.12
0.10
0.08
0.06
0.04
0.02
0.00


N = 15, E = -7.679
0
10
20
0
10
20
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00


N = 16, E = -7.677
0
10
20
0
10
20
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00


N = 17, E = -7.604
0
10
20
0
10
20
0.08
0.06
0.04
0.02
0.00


N = 18, E = -7.604
0
10
20
0
10
20
0.08
0.06
0.04
0.02
0.00


N = 19, E = -7.594
0
10
20
0
10
20
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00