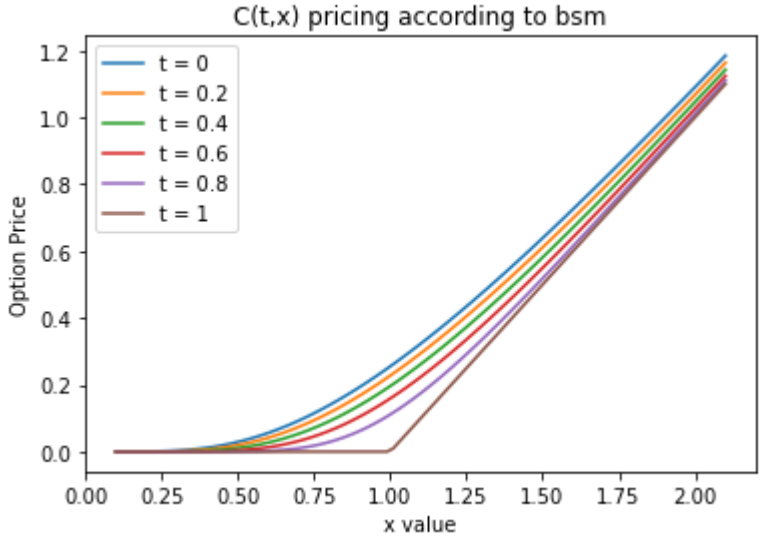
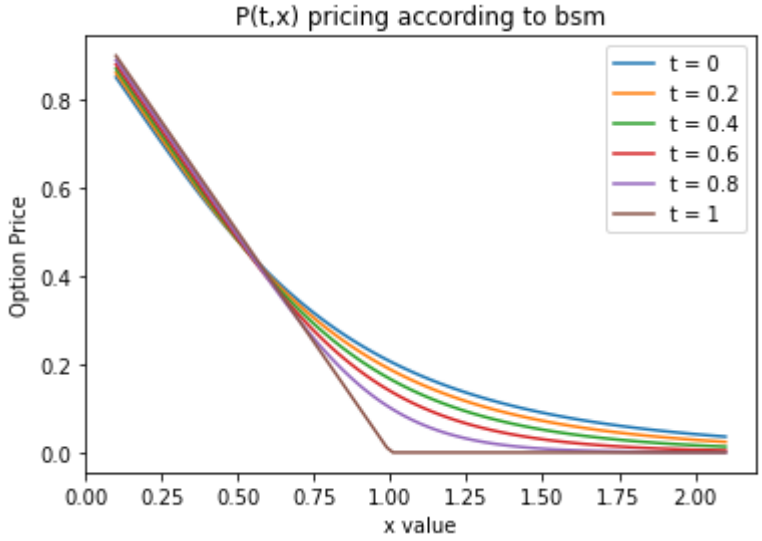
In [5]:

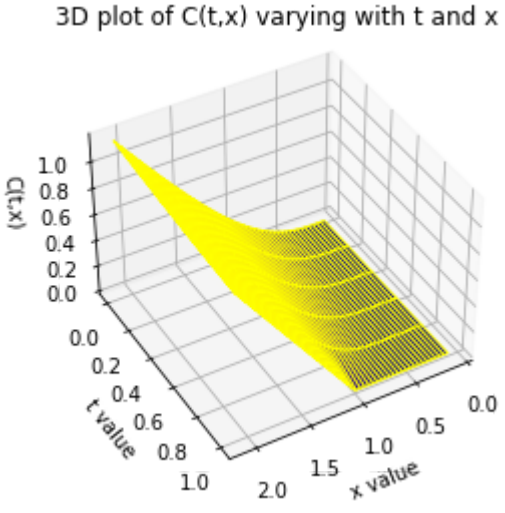
```
import math
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt

def plus(S,K,T,r,sigma):
    return (math.log(S/K)+(r+0.5*sigma*sigma)*T)/(sigma*math.sqrt(T))

def minus(S,K,T,r,sigma):
    return (math.log(S/K)+(r-0.5*sigma*sigma)*T)/(sigma*math.sqrt(T))

def BSM_call_option(S,K,T,t,r,sig):
    if(t==T):
        return np.maximum(0,S-K)
    return (S*norm.cdf(plus(S,K,T-t,r,sig)))-(K*math.exp(-r*(T-t))*norm.cdf(minu
s(S,K,T-t,r,sig)))

def BSM_put_option(S,K,T,t,r,sig):
    if(t==T):
        return np.maximum(0,K-S)
    return K*math.exp(-r*(T-t))-S+BSM_call_option(S,K,T,t,r,sig)


T = 1
K = 1
r = 0.05
sigma = 0.6
t = [0,0.2,0.4,0.6,0.8,1]
x = np.linspace(0.1,2.1,100)
for m in t:
    BSM_call=[]
    for i in x:
        BSM_call.append(BSM_call_option(i,K,T,m,r,sigma))
    plt.plot(x,BSM_call,label='t = '+ str(m))
plt.title('C(t,x) pricing according to bsm')
plt.xlabel('x value')
plt.ylabel('Option Price')
plt.legend()
plt.show()

for m in t:
    BSM_put=[]
    for i in x:
        BSM_put.append(BSM_put_option(i,K,T,m,r,sigma))
    plt.plot(x,BSM_put,label='t = '+ str(m))
plt.title('P(t,x) pricing according to bsm')
plt.xlabel('x value')
plt.ylabel('Option Price')
plt.legend()
plt.show()

BSM_call={}
BSM_put={}
p = []
q = []
BSM_call_t = []
BSM_put_t = []
for i in range(0,len(t)):
    for j in range(0,100):
```

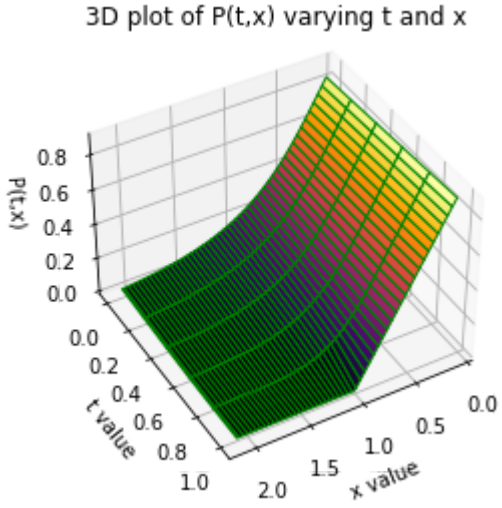
```
        BSM_call[i,j]=BSM_call_option(x[j],K,T,t[i],r,sigma)
        BSM_put[i,j]=BSM_put_option(x[j],K,T,t[i],r,sigma)
        p.append(x[j])
        q.append(t[i])
        BSM_call_t.append(BSM_call[i,j])
        BSM_put_t.append(BSM_put[i,j])

ax = plt.axes(projection ='3d')


X = np.reshape(p, (6, 100))
Y = np.reshape(q, (6, 100))
Z = np.reshape(BSM_call_t, (6, 100))
ax.plot_surface(X, Y, Z,cmap ='plasma', edgecolor ='yellow')

ax.set_title('3D plot of C(t,x) varying with t and x' )
ax.set_xlabel('x value')
ax.set_ylabel('t value')
ax.set_zlabel('C(t,x)')
ax.view_init(40, 60)
plt.show()

ax = plt.axes(projection ='3d')
X = np.reshape(p, (6, 100))
Y = np.reshape(q, (6, 100))
Z = np.reshape(BSM_put_t, (6, 100))
ax.view_init(40, 60)
ax.plot_surface(X, Y, Z, cmap ='inferno', edgecolor ='green')
ax.set_title('3D plot of P(t,x) varying t and x' )
ax.set_xlabel('x value')
ax.set_ylabel('t value')
ax.set_zlabel('P(t,x)')
plt.show()

t=np.linspace(0,0.99,100)
BSM_call={ }
BSM_put={ }
p=[ ]
q=[ ]
BSM_call_t=[ ]
BSM_put_t=[ ]
for i in range(0,len(t)):
    for j in range(0,100):
        BSM_call[i,j]=BSM_call_option(x[j],K,T,t[i],r,sigma)
        BSM_put[i,j]=BSM_put_option(x[j],K,T,t[i],r,sigma)

        p.append(x[j])
        q.append(t[i])
        BSM_call_t.append(BSM_call[i,j])
        BSM_put_t.append(BSM_put[i,j])

ax = plt.axes(projection ='3d')

X = np.reshape(p, (100, 100))
Y = np.reshape(q, (100, 100))
Z = np.reshape(BSM_call_t, (100, 100))
ax.plot_surface(X, Y, Z,cmap ='plasma',edgecolor='red')

ax.set_title('3D plot of C(t,x) varying with t and x' )
ax.set_xlabel('x value')
ax.set_ylabel('t value')
```

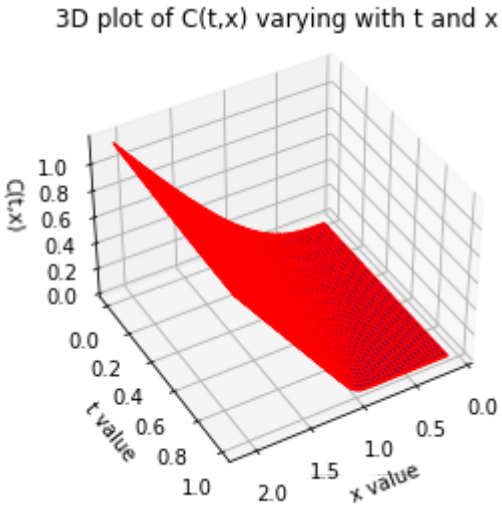
```
ax.set_zlabel('C(t,x)')
ax.view_init(40, 60)
plt.show()

ax = plt.axes(projection ='3d')
X = np.reshape(p, (100, 100))
Y = np.reshape(q, (100, 100))
Z = np.reshape(BSM_put_t, (100, 100))
ax.view_init(40, 60)
ax.plot_surface(X, Y, Z, cmap ='inferno',edgecolor='green')
ax.set_title('3D plot of P(t,x) varying t and x' )
ax.set_xlabel('x value')
ax.set_ylabel('t value')
ax.set_zlabel('P(t,x)')
plt.show()
```

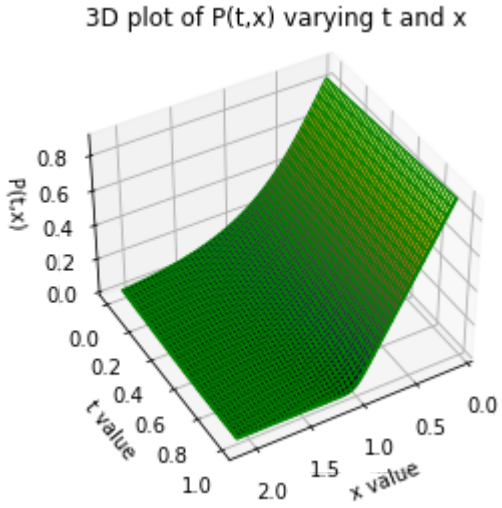

C(t,x) pricing according to bsm
t = 0
t = 0.2
t = 0.4
t = 0.6
t = 0.8
t = 1
Option Price
x value


P(t,x) pricing according to bsm
t = 0
t = 0.2
t = 0.4
t = 0.6
t = 0.8
t = 1
Option Price
x value


3D plot of C(t,x) varying with t and x
C(t,x)
t value
x value


3D plot of P(t,x) varying t and x
P(t,x)
0.0
0.2
0.4
0.6
0.8
t value
0.0
0.5
1.0
1.5
2.0
x value


3D plot of C(t,x) varying with t and x
C(t,x)
0.0
0.2
0.4
0.6
0.8
1.0
t value
0.0
0.5
1.0
1.5
2.0
x value


3D plot of P(t,x) varying t and x
P(t,x)
0.0
0.2
0.4
0.6
0.8
t value
0.0
0.5
1.0
1.5
2.0
x value

In [ ]: