In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
def EGBM(K):
    alpha=0.05
    #mu=-0.45
    sig=1
    So=100
    T=4
    x=np.random.normal(size=100000)
    S=So*np.exp((alpha-sig/2)*T + sig*np.sqrt(T)*x)
    B=np.exp(-alpha*T)*(S-K)
    C_vec=B
    C_vec=np.where(C_vec>0,C_vec,0)
    return(C_vec)
```

In [3]:
```
np.std(EGBM(80))
```

Out[3]:
```
665.4505154012433
```

In [4]:
```
np.std(EGBM(100))
```

Out[4]:
```
536.6771004817291
```

In [5]:
```
np.std(EGBM(120))
```

Out[5]:
```
855.0827137504558
```

In [6]:
```
def AntitheticEGBM(K):
    alpha=0.05
    #mu=-0.45
    sig=1
    So=100
    T=4
    x=np.random.normal(size=100000)
    S=So*np.exp((alpha-sig/2)*T + sig*np.sqrt(T)*x)
    S_ne=So*np.exp((alpha-sig/2)*T + sig*np.sqrt(T)*(-x))
    B=np.exp(-alpha*T)*(S-K)
    B_ne=np.exp(-alpha*T)*(S_ne-K)
    C=sum(np.where(B>0,B,0))
    C=C+sum(np.where(B_ne>0,B_ne,0))
    C_vec=np.where(B_ne>0,B_ne,0)+np.where(B>0,B,0)
    C_vec=np.where(C_vec>0,C_vec/2,0)
    return(C_vec)
```

In [7]:
```
np.std(AntitheticEGBM(80))
```

Out[7]:
```
467.1931607070753
```

```
In [8]: np.std(AntitheticEGBM(100))

Out[8]: 457.82138774020547

In [9]: np.std(AntitheticEGBM(120))

Out[9]: 493.6409989370343

In [10]: def AGBM(K):
             alpha=0.05
             #mu=-0.45
             sig=1
             So=100
             dT=0.01
             T=4
             S=np.repeat(So,100000)
             t=np.linspace(0.01,T,num=400)
             ST=[]
             ST.append(S)
             for i in t:
                 Vec=np.random.normal(size=100000)
                 S=S*np.exp((alpha-sig/2)*dT+sig*np.sqrt(dT)*Vec)
                 ST.append(S)

             test=np.array(ST)
             S_bar=np.mean(test,axis=0)
             B=np.exp(-alpha*T)*(S_bar-K)
             C=np.where(B>0,B,0)
             return(C)

In [11]: np.std(AGBM(80))

Out[11]: 206.29030970235738

In [12]: np.std(AGBM(100))

Out[12]: 224.87447481711393

In [13]: np.std(AGBM(120))

Out[13]: 197.62470951865345
```

In [14]:

```
def AGBM_anti(K):
    alpha=0.05
    #mu=-0.45
    sig=1
    So=100
    dT=0.01
    T=4
    S=np.repeat(So,100000)
    S_ne=np.repeat(So,100000)
    t=np.linspace(0.01,T,num=400)
    ST=[]
    ST_ne=[]
    ST.append(S)
    ST.append(S_ne)
    for i in t:
        Vec=np.random.normal(size=100000)
        S=S*np.exp((alpha-sig/2)*dT+sig*np.sqrt(dT)*Vec)
        S_ne=S_ne*np.exp((alpha-sig/2)*dT+sig*np.sqrt(dT)*(-Vec))
        ST.append(S)
        ST_ne.append(S_ne)

    test=np.array(ST)
    test_ne=np.array(ST_ne)
    S_bar=np.mean(test,axis=0)
    S_bar_ne=np.mean(test_ne,axis=0)
    B=np.exp(-alpha*T)*(S_bar-K)
    B_ne=np.exp(-alpha*T)*(S_bar_ne-K)
    C=(np.where(B>0,B,0)+np.where(B_ne>0,B_ne,0))/2
    return(C)
```

In [15]:

```
np.std(AGBM_anti(80))
```

Out[15]:

```
129.32659457705202
```

In [16]:

```
np.std(AGBM_anti(100))
```

Out[16]:

```
139.04517758906402
```

In [17]:

```
np.std(AGBM_anti(120))
```

Out[17]:

```
144.87762694597055
```

In [18]:

```
def LGBM():
    alpha=0.05
    #mu=-0.45
    sig=1
    So=100
    dT=0.01
    T=4
    S=np.repeat(So,100000)
    t=np.linspace(0.01,4,num=400)
    ST=[]
    ST.append(S)
    for i in t:
        Vec=np.random.normal(size=100000)
        S=S*np.exp((alpha-sig/2)*dT+sig*np.sqrt(dT)*Vec)
        ST.append(S)

    test=np.array(ST)
    S=np.min(test,axis=0)
    ST=test[400,:]
    B=np.exp(-alpha*T)*(ST-S)
    C=np.where(B>0,B,0)
    return(C)
```

In [19]:

```
np.std(LGBM())
```

Out[19]:

```
652.378264570656
```

In [20]:

```
def LGBM_anti():
    alpha=0.05
    #mu=-0.45
    sig=1
    So=100
    dT=0.01
    T=4
    S=np.repeat(So,100000)
    S_ne=np.repeat(So,100000)
    t=np.linspace(0.01,4,num=400)
    ST=[]
    ST_ne=[]
    ST.append(S)
    ST_ne.append(S_ne)
    for i in t:
        Vec=np.random.normal(size=100000)
        S=S*np.exp((alpha-sig/2)*dT+sig*np.sqrt(dT)*Vec)
        S_ne=S_ne*np.exp((alpha-sig/2)*dT+sig*np.sqrt(dT)*(-Vec))
        ST.append(S)
        ST_ne.append(S_ne)

    test=np.array(ST)
    test_ne=np.array(ST_ne)
    S=np.min(test,axis=0)
    S_ne=np.min(test_ne,axis=0)
    ST=test[400,:]
    ST_ne=test_ne[400,:]
    B=np.exp(-alpha*T)*(ST-S)
    B_ne=np.exp(-alpha*T)*(ST_ne-S_ne)
    C=(np.where(B>0,B,0) +np.where(B_ne>0,B_ne,0))/2
    return(C)
```

In [21]:

```
np.std(LGBM_anti())
```

Out[21]:

```
433.30280256436953
```

In [22]:

```
def pilot_crack(rho,K):
    r=0.05
    X0=50
    Y0=45
    sigx=0.2
    sigy=0.3
    T=1
    mean=[0,0]
    cov=[[1,rho],[rho,1]]
    x,y=np.random.multivariate_normal(mean,cov,10000).T
    XT=X0*np.exp((r-sigx/2)*T+sigx*x)
    YT=Y0*np.exp((r-sigy/2)*T+sigy*y)
    B=np.exp(-r*T)*(XT-YT-K)
    C=np.where(B>0,B,0)
    return(np.cov(C,x))
```

In [23]:
```
pilot_crack(0.5,0)
```

Out[23]:
```
array([[60.53862953,  2.82262196],
       [ 2.82262196,  0.98902203]])
```

In [24]:
```
pilot_crack(0.5,5)
```

Out[24]:
```
array([[41.28441538,  2.51240479],
       [ 2.51240479,  1.00315108]])
```

In [25]:
```
pilot_crack(0.5,10)
```

Out[25]:
```
array([[22.6505033 ,  1.82793826],
       [ 1.82793826,  0.98585485]])
```

In [26]:
```
def crack(rho,K):
    r=0.05
    X0=50
    Y0=45
    sigx=0.2
    sigy=0.3
    T=1
    mean=[0,0]
    cov=[[1,rho],[rho,1]]
    x,y=np.random.multivariate_normal(mean,cov,10000).T
    XT=X0*np.exp((r-sigx/2)*T+sigx*x)
    YT=Y0*np.exp((r-sigy/2)*T+sigy*y)
    B=np.exp(-r*T)*(XT-YT-K)
    C=np.where(B>0,B,0)
    return(C)
```

In [27]:
```
np.std(crack(0.5,0))
```

Out[27]:
```
7.82572128045003
```

In [28]:
```
np.std(crack(0.5,5))
```

Out[28]:
```
6.344230381324534
```

In [29]:
```
np.std(crack(0.5,10))
```

Out[29]:
```
4.828932903274982
```

In [30]:

```
def crack1(rho,K):
    r=0.05
    X0=50
    Y0=45
    sigx=0.2
    sigy=0.3
    T=1
    mean=[0,0]
    cov=[[1,rho],[rho,1]]
    x,y=np.random.multivariate_normal(mean,cov,10000).T
    XT=X0*np.exp((r-sigx/2)*T+sigx*x)
    YT=Y0*np.exp((r-sigy/2)*T+sigy*y)
    B=np.exp(-r*T)*(XT-YT-K)
    C=np.where(B>0,B,0)+C*x
    return(C)
```

In [31]:

```
np.std(crack1(0.5,0))
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-31-fe9c9cd9a299> in <module>()
----> 1 np.std(crack1(0.5,0))

<ipython-input-30-9b7b6025985b> in crack1(rho, K)
     12     YT=Y0*np.exp((r-sigy/2)*T+sigy*y)
     13     B=np.exp(-r*T)*(XT-YT-K)
---> 14     C=np.where(B>0,B,0)+C*x
     15     return(C)

UnboundLocalError: local variable 'C' referenced before assignment
```

In [32]:

```
def M_iid_gen():
    Sum=[0]
    while(True):
        x=np.random.uniform()
        if(x>Sum[-1]):
            Sum.append(x);
        else:
            break;

    return(len(Sum))

def M_iid_gen_2():
    Sum_anti=[1]
    while(True):
        x=np.random.uniform()
        if((1-x)<Sum_anti[-1]):
            Sum_anti.append((1-x))
        else:
            break;

    return(len(Sum_anti))

Sum=[] # Without Antithetic
Sum_anti=[] # With antithetic
for i in range(0,10000):
    Sum.append(M_iid_gen())
    Sum_anti.append((M_iid_gen()+M_iid_gen_2())/2)

X1=np.array(Sum)
X2=np.array(Sum_anti)
```

In [33]:

```
X1.std()
```

Out[33]:

```
0.8677831756838802
```

In [34]:

```
X2.std()
```

Out[34]:

```
0.6133836972075472
```

In [35]:

```
def int_esti():
    x=np.random.uniform(size=10000)
    return(np.exp(x**2))
```

In [36]:

```
X=int_esti()
```

In [37]:

```
X.mean()
```

Out[37]:

```
1.452123699216208
```

In [38]:

```
X.std()
```

Out[38]:

```
0.46934698660876467
```

In [39]:

```
def int_est_2():
    x1=np.random.uniform(size=10000)
    x2=1-x1
    return((np.exp(x1**2)+np.exp(x2**2))/2)

X=int_est_2()
```

In [40]:

```
X.mean()
```

Out[40]:

```
1.462367913466333
```

In [41]:

```
X.std()
```

Out[41]:

```
0.1665324869562846
```

In [42]:

```
def pilot():
    y=np.random.uniform(size=10000)
    x=np.exp(y**2)
    co=np.cov(x,y)
    va=np.var(y)
    return(-co/va)
```

In [43]:

```
pilot()
```

Out[43]:

```
array([[-2.72291863, -1.54083002],
       [-1.54083002, -1.00010001]])
```

In [44]:

```
def con_var():
    x=np.uniform.random(size=10000)
    Cont=np.exp(x**2)-1.5367*(x-0.5)
    return(np.mean(Cont))
```

In [45]:

```
X=con_var()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-45-713ab5a4dde6> in <module>()
----> 1 X=con_var()

<ipython-input-44-f9fd392ced19> in con_var()
      1 def con_var():
----> 2     x=np.uniform.random(size=10000)
      3     Cont=np.exp(x**2)-1.5367*(x-0.5)
      4     return(np.mean(Cont))

AttributeError: module 'numpy' has no attribute 'uniform'
```

In [ ]:

```
```