Numerical Method in Finance

Homework 3

- Yu-Ching Liao ycliao3@illinois.edu

Question 1

Using either April 10th end of day market close data, replicate VIX index level via the VIX White paper https://cdn.cboe.com/api/global/us_indices/governance/Cboe_Volatility_Index_Mathematics_ to an external site.

As discussed during the lecture, the VIX index calculation was motivated by the Carr-Madan formula we went through today in class.

Getting near term and next term

In [167…
```
import pandas as pd
import math
```

In [184…
```
df1 = pd.read_csv("$spx-options-exp-2023-05-03-weekly-show-all-stacked-04-11
df2 = pd.read_csv("$spx-options-exp-2023-05-19-weekly-show-all-stacked-04-11
```

Create Order Book

In [185…
```
strikes_1 = sorted(list(set(df1['Strike'].values)))
strikes_2 = sorted(list(set(df2['Strike'].values)))
order_book_01 = {
    'Strike': [],
    "Call_Bid": [],
    "Call_Ask": [],
    "Put_Bid": [],
```

```
    "Put_Ask": []
}
for i in range(len(strikes_1)):
    d = df1[df1['Strike'] == strikes_1[i]]
    order_book_01['Strike'].append(strikes_1[i])
    order_book_01['Call_Bid'].append(d[d['Type'] == 'Call']['Bid'].values[0]
    order_book_01['Call_Ask'].append(d[d['Type'] == 'Call']['Ask'].values[0]
    order_book_01['Put_Bid'].append(d[d['Type'] == 'Put']['Bid'].values[0])
    order_book_01['Put_Ask'].append(d[d['Type'] == 'Put']['Ask'].values[0])
order_book_01 = pd.DataFrame(order_book_01)
display(order_book_01)

order_book_02 = {
    'Strike': [],
    "Call_Bid": [],
    "Call_Ask": [],
    "Put_Bid": [],
    "Put_Ask": []
}
for i in range(len(strikes_2)):
    d = df2[df2['Strike'] == strikes_2[i]]
    order_book_02['Strike'].append(strikes_2[i])
    order_book_02['Call_Bid'].append(d[d['Type'] == 'Call']['Bid'].values[0]
    order_book_02['Call_Ask'].append(d[d['Type'] == 'Call']['Ask'].values[0]
    order_book_02['Put_Bid'].append(d[d['Type'] == 'Put']['Bid'].values[0])
    order_book_02['Put_Ask'].append(d[d['Type'] == 'Put']['Ask'].values[0])
order_book_02 = pd.DataFrame(order_book_02)
display(order_book_02)
```

	Strike	Call_Bid	Call_Ask	Put_Bid	Put_Ask
0	1200	2910.60	2916.00	0.0	0.1
1	1400	2711.40	2716.70	0.0	0.1
2	1600	2512.00	2517.30	0.0	0.1
3	1800	2312.60	2318.00	0.0	0.1
4	2000	2113.30	2118.70	0.0	0.1
...	...	...	...	...	...
93	4400	1.20	1.45	275.4	280.6
94	4500	0.25	0.40	374.0	379.3
95	4600	0.10	0.25	473.5	478.8
96	4800	0.00	0.10	672.8	678.1
97	5000	0.00	0.10	872.1	877.5

98 rows × 5 columns

	Strike	Call_Bid	Call_Ask	Put_Bid	Put_Ask
0	200	3903.1	3908.50	0.0	0.05
1	400	3704.2	3709.60	0.0	0.05
2	600	3505.3	3510.70	0.0	0.05
3	800	3306.5	3311.90	0.0	0.05
4	1000	3107.6	3113.00	0.0	0.05
...	...	...	...	...	...
325	6800	0.0	0.05	2654.6	2660.00
326	7000	0.0	0.05	2853.5	2858.90
327	7200	0.0	0.05	3052.4	3057.70
328	7400	0.0	0.05	3251.2	3256.60
329	7600	0.0	0.05	3450.1	3455.50

330 rows × 5 columns

VIX Calculation

In [192…

```
rates=[0.0453, 0.0489]
datafiles=[order_book_01, order_book_02]
verbose=1

quotedata=[[], []]
for j in (0,1):
    f = datafiles[j]
    display(f)
    for line in f.index:
        ar=[]
        for v in f.iloc[line].values:
            ar.append(float(v))
        quotedata[j].append(ar)

#get minutes and years to near term and next term expiration dates (p5)

date_1 = pd.Timestamp('2023-04-11')
date_2 = pd.Timestamp('2023-05-03')
date_3 = pd.Timestamp('2023-05-19')

# Calculate the difference in minutes between the dates
minutes_diff_1 = (date_2 - date_1).total_seconds() / 60
minutes_diff_2 = (date_3 - date_1).total_seconds() / 60
print(minutes_diff_1, minutes_diff_2)

Nt=[minutes_diff_1, minutes_diff_2]              #minutes
T=[Nt[0]/(60*24*365), Nt[1]/(60*24*365)]         #years

if(verbose>=1):
        print('Nt:', Nt)
```

```
        print('T:', T)

#Step 1: Select the options to be used in the VIX Index calculation
#Compute F for near term and next term (p6)
F=[None, None]
for j in (0,1):
        mindiff=None
        diff=None
        mindiff=None
        Fstrike=None
        Fcall=None
        Fput=None
        for d in quotedata[j]:
                diff=abs( ((d[1]+d[2])/2) - ((d[3]+d[4])/2) )
                if(mindiff is None or diff<mindiff):
                        mindiff=diff
                        Fstrike=d[0]
                        Fcall=(d[1]+d[2])/2
                        Fput=(d[3]+d[4])/2
        F[j]=Fstrike + math.exp(rates[j]*T[j]) * (Fcall - Fput)

if(verbose>=1): print('F:', F)


#select the options to be used in the VIX Index calculation (p6,7)
selectedoptions=[[], []]
k0=[None, None]
for j in (0,1):
        i=0
        for d in quotedata[j]:
                if(d[0]<F[j]):
                        k0[j]=d[0]
                        k0i=i
                i+=1

        d=quotedata[j][k0i]
        ar=[d[0], 'put/call average', (((d[1]+d[2])/2)+((d[3]+d[4])/2))/2]
        selectedoptions[j].append(ar)

        i=k0i-1
        b=True
        previousbid=None
        while(b and i>=0):
                d=quotedata[j][i]
                if(d[3]>0):
                        ar=[d[0], 'put', (d[3]+d[4])/2]
                        selectedoptions[j].insert(0,ar)
                else:
                        if(previousbid==0): b=False
                previousbid=d[3]
                i-=1

        i=k0i+1
        b=True
        previousbid=None
        while(b and i<len(quotedata[j])):
```

```
                d=quotedata[j][i]
                if(d[1]>0):
                        ar=[d[0], 'call', (d[1]+d[2])/2]
                        selectedoptions[j].append(ar)
                else:
                        if(previousbid==0): b=False
                previousbid=d[1]
                i+=1

if(verbose==2):
        print('selectedoptions (near term):')
        for e in selectedoptions[0]: print('', e)
        print('selectedoptions (next term):')
        for e in selectedoptions[1]: print('', e)

#Step 2: Calculate volatility for both near-term and next-term options (p8)
for j in (0,1):
        i=0
        for d in selectedoptions[j]:
                if(i==0):
                        deltak=selectedoptions[j][1][0]-selectedoptions[j][0
                elif(i==len(selectedoptions[j])-1):
                        deltak=selectedoptions[j][i][0]-selectedoptions[j][i
                else:
                        deltak=(selectedoptions[j][i+1][0]-selectedoptions[j
                contributionbystrike=(deltak/(d[0]*d[0])) * math.exp(rates[j
                selectedoptions[j][i].append(contributionbystrike)
                i+=1
if(verbose==2):
        print('contributions by strike (near term):')
        for e in selectedoptions[0]: print('', e)
        print('contributions by strike (next term):')
        for e in selectedoptions[1]: print('', e)


aggregatedcontributionbystrike=[None, None]
for j in (0,1):
        aggregatedcontributionbystrike[j]=0
        for d in selectedoptions[j]:
                aggregatedcontributionbystrike[j]+=d[3]
        aggregatedcontributionbystrike[j]=(2/T[j])*aggregatedcontributionbys


sigmasquared=[None, None]
for j in (0,1):
        sigmasquared[j]=aggregatedcontributionbystrike[j] - (1/T[j])*(F[j]/k

if(verbose): print('sigmasquared:', sigmasquared)

#Step 3: Calculate the 30-day weighted average of sigmasquared[0] and sigmas
N30=30*1440
N365=365*1440
VIX=100 * math.sqrt( ((T[0]*sigmasquared[0])*(Nt[1]-N30)/(Nt[1]-Nt[0]) + (T[

print('VIX:', VIX)
```

	Strike	Call_Bid	Call_Ask	Put_Bid	Put_Ask
0	1200	2910.60	2916.00	0.0	0.1
1	1400	2711.40	2716.70	0.0	0.1
2	1600	2512.00	2517.30	0.0	0.1
3	1800	2312.60	2318.00	0.0	0.1
4	2000	2113.30	2118.70	0.0	0.1
...	...	...	...	...	...
93	4400	1.20	1.45	275.4	280.6
94	4500	0.25	0.40	374.0	379.3
95	4600	0.10	0.25	473.5	478.8
96	4800	0.00	0.10	672.8	678.1
97	5000	0.00	0.10	872.1	877.5

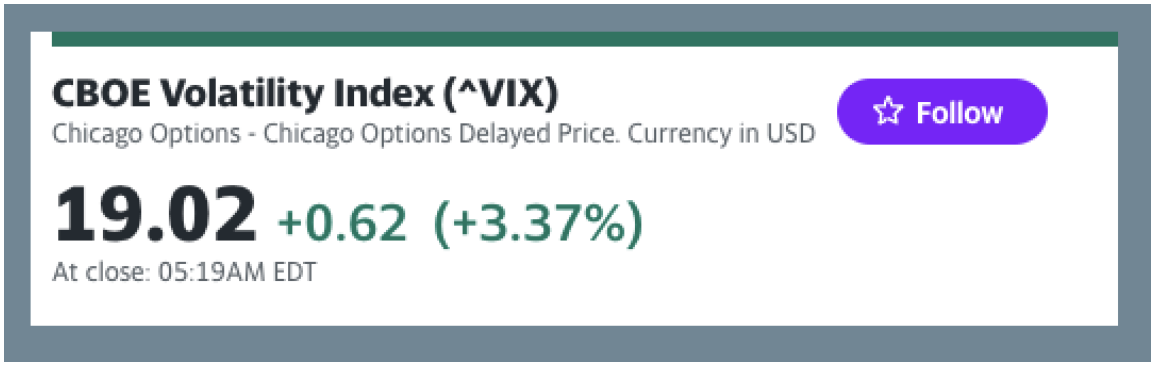
98 rows × 5 columns

	Strike	Call_Bid	Call_Ask	Put_Bid	Put_Ask
0	200	3903.1	3908.50	0.0	0.05
1	400	3704.2	3709.60	0.0	0.05
2	600	3505.3	3510.70	0.0	0.05
3	800	3306.5	3311.90	0.0	0.05
4	1000	3107.6	3113.00	0.0	0.05
...	...	...	...	...	...
325	6800	0.0	0.05	2654.6	2660.00
326	7000	0.0	0.05	2853.5	2858.90
327	7200	0.0	0.05	3052.4	3057.70
328	7400	0.0	0.05	3251.2	3256.60
329	7600	0.0	0.05	3450.1	3455.50

330 rows × 5 columns

```
31680.0 54720.0
Nt: [31680.0, 54720.0]
T: [0.06027397260273973, 0.10410958904109589]
F: [4122.593438059378, 4127.78877133232]
sigmasquared: [0.03372252533389646, 0.03740112832871767]
VIX: 18.987445123190366
```

This result is approximate to the value that yahoo finance website provide.

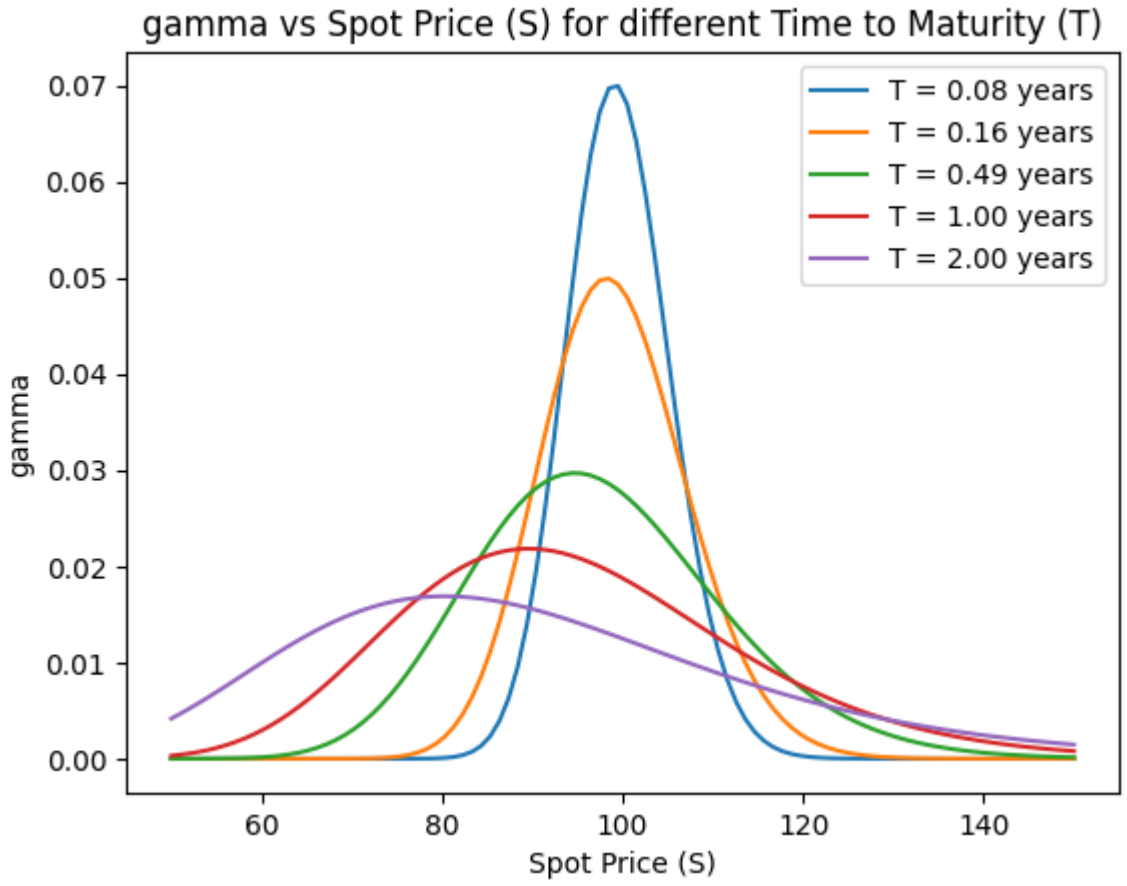


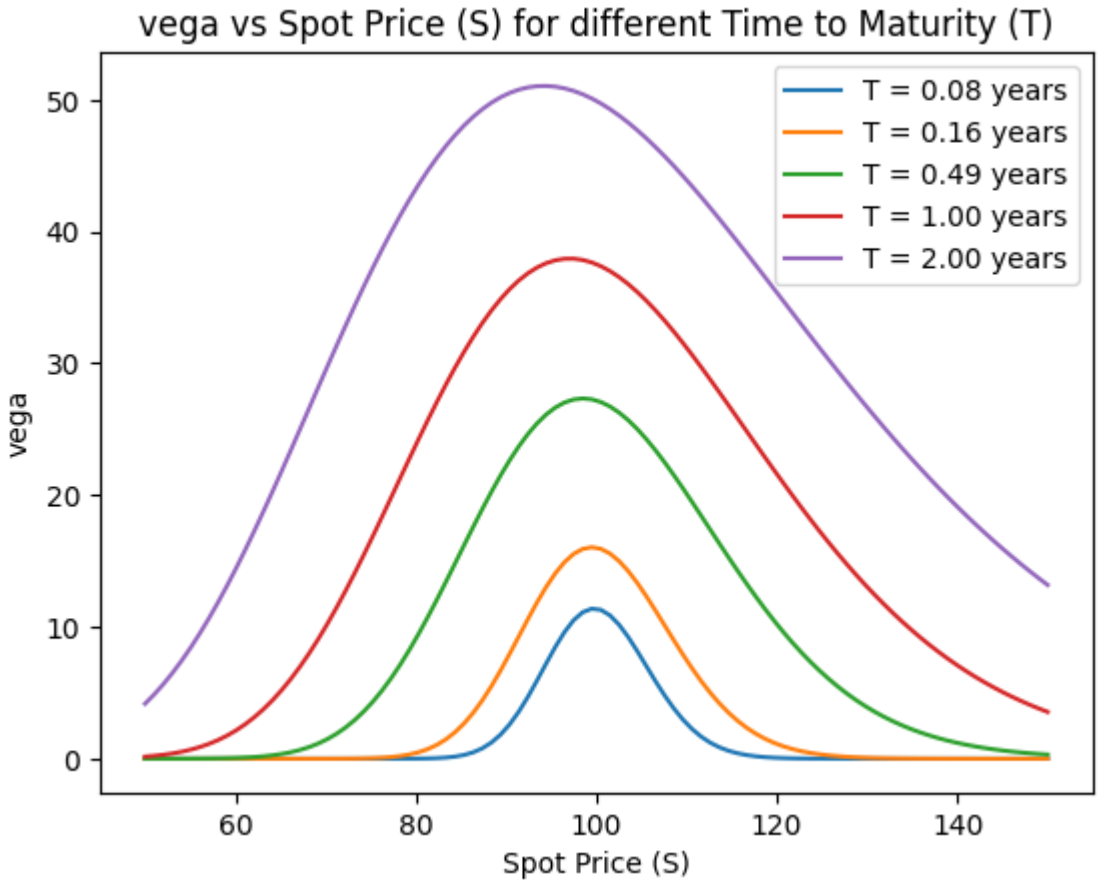
Question 2

Use Black-Scholes formula and calculus to derive Gamma(d^2 C/dS^2) and Vega(dC/dsigma). Plot each as a function of spot price(S) and time to maturity(T), i.e. Each plot should have multiple curves each corresponding to different time to maturity but sharing the same x-axis(spot level).
You should have two plots, one for Gamma and one for Vega.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

def black_scholes(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return d1, d2

def gamma(S, K, T, r, sigma):
    d1, _ = black_scholes(S, K, T, r, sigma)
    return norm.pdf(d1) / (S * sigma * np.sqrt(T))

def vega(S, K, T, r, sigma):
    d1, _ = black_scholes(S, K, T, r, sigma)
    return S * norm.pdf(d1) * np.sqrt(T)

def plot_greeks(greek_func, time_to_maturities, S, K, r, sigma):
    plt.figure()
    for T in time_to_maturities:
        greek_values = [greek_func(s, K, T, r, sigma) for s in S]
        plt.plot(S, greek_values, label=f'T = {T:.2f} years')

    plt.xlabel('Spot Price (S)')
    plt.ylabel(greek_func.__name__)
    plt.legend()
    plt.title(f'{greek_func.__name__} vs Spot Price (S) for different Time t
    plt.show()

if __name__ == "__main__":
    S = np.linspace(50, 150, 100)
    K = 100
```

```
    r = 0.05
    sigma = 0.2
    time_to_maturities = [30 / 365, 60 / 365, 180 / 365, 365 / 365, 730 / 36

    plot_greeks(gamma, time_to_maturities, S, K, r, sigma)
    plot_greeks(vega, time_to_maturities, S, K, r, sigma)
```




vega vs Spot Price (S) for different Time to Maturity (T)
T = 0.08 years
T = 0.16 years
T = 0.49 years
T = 1.00 years
T = 2.00 years
vega
Spot Price (S)
0
10
20
30
40
50
60
80
100
120
140

ILLINOIS