Name: Long Zhang
Fintech545 hw6
Date:3/24/2023

Problem1

Implement the closed form greeks for GBSM, we first define the gbsm function and then apply the closed form formula of each greeks in the lecture. Next, for finite difference derivative calculation, we need to use partial derivative with respect to different parameters to find the greeks' formula. For all greeks' call and put value, the output are shown below:

```
# delta
delta_call = gbsm_delta("Call", S, X, ttm, ivol, r, b)
delta_put = gbsm_delta("Put", S, X, ttm, ivol, r, b)
gbsm_delta_num = cal_partial_derivative(gbsm, 1, 'underlying')
delta_call_num = gbsm_delta_num("Call", S, X, ttm, ivol, r, b)
delta_put_num = gbsm_delta_num("Put", S, X, ttm, ivol, r, b)
print(delta_call, delta_put)
print(delta_call_num, delta_put_num)
```

```
0.08297130333914773 -0.9165496333661425
1.986922887836227e-14 -0.9858611793589489
```

```
# vega
vega_call = gbsm_vega("Call", S, X, ttm, ivol, r, b)
vega_put = gbsm_vega("Put", S, X, ttm, ivol, r, b)
gbsm_vega_num = cal_partial_derivative(gbsm, 1, 'ivol')
vega_call_num = gbsm_vega_num("Call", S, X, ttm, ivol, r, b)
vega_put_num = gbsm_vega_num("Put", S, X, ttm, ivol, r, b)
print(vega_call, vega_put)
print(vega_call_num, vega_put_num)
```

```
6.942036604441163 6.942036604441163
9.767308509353247e-12 0.0
```

```
# gamma
gamma_call = gbsm_gamma("Call", S, X, ttm, ivol, r, b)
gamma_put = gbsm_gamma("Put", S, X, ttm, ivol, r, b)
gbsm_gamma_num = cal_partial_derivative(gbsm, 2, 'underlying')
gamma_call_num = gbsm_gamma_num("Call", S, X, ttm, ivol, r, b)
gamma_put_num = gbsm_gamma_num("Put", S, X, ttm, ivol, r, b)
print(gamma_call, gamma_put)
print(gamma_call_num, gamma_put_num)
```

```
0.016830979206204362 0.016830979206204362
9.042259959910318e-14 2.8421709430404007e-08
```

```
# theta
theta_call = gbsm_theta("Call", S, X, ttm, ivol, r, b)
theta_put = gbsm_theta("Put", S, X, ttm, ivol, r, b)
gbsm_theta_num = cal_partial_derivative(gbsm, 1, 'ttm')
theta_call_num = -gbsm_theta_num("Call", S, X, ttm, ivol, r, b)
theta_put_num = -gbsm_theta_num("Put", S, X, ttm, ivol, r, b)
print(theta_call, theta_put)
print(theta_call_num, theta_put_num)
```

```
-8.126522359668838 -1.9409914783019566
-1.5819627783921197e-12 8.957748660847642
```

```
# rho
rho_call = gbsm_rho("Call", S, X, ttm, ivol, r, b)
rho_put = gbsm_rho("Put", S, X, ttm, ivol, r, b)
gbsm_rho_num = cal_partial_derivative(gbsm, 1, 'rf')
rho_call_num = gbsm_rho_num("Call", S, X, ttm, ivol, r, b)
rho_put_num = gbsm_rho_num("Put", S, X, ttm, ivol, r, b)
print(rho_call, rho_put)
print(rho_call_num, rho_put_num)
```

```
1.1025939156368187 -13.758003122735788
-3.8791500416870894e-16 -1.1887809038029218
```

```
# carry rho
carry_rho_call = gbsm_carry_rho("Call", S, X, ttm, ivol, r, b)
carry_rho_put = gbsm_carry_rho("Put", S, X, ttm, ivol, r, b)
gbsm_carry_rho_num = cal_partial_derivative(gbsm, 1, 'b')
carry_rho_call_num = gbsm_carry_rho_num("Call", S, X, ttm, ivol, r,
carry_rho_put_num = gbsm_carry_rho_num("Put", S, X, ttm, ivol, r, b)
print(carry_rho_call, carry_rho_put)
print(carry_rho_call_num, carry_rho_put_num)
```

```
1.132953825011723 -12.515271800549371
2.7148101769271126e-13 -13.461704838221067
```

While we applying derivative method, the delta, vega, gamma, theta, carry rho values are greater than closed form. For rho, the closed form value is greater than derivative method.

For binomial tree model, we apply the form in lecture note;

$$P_{i,j} = max\left(\pi\left(Su^{i}d^{(j-i)}\right),\ e^{-r\Delta t}\left[pP_{j+i,i+1} + (1-p)P_{j+1,i}\right]\right)$$

Set there are N=200 steps, the output value for call and put for no dividend and with dividend are shown below:

```
no div call:0.28273717030825374
no div put:14.649213091283734
```

```
Binomial tree value with dividend for call: 0.27886724160603915
Binomial tree value with dividend for put: 15.152886604860912
```

For the greeks of binomial tree model, the put and call value are shown below:
Delta:
Call=0.071159 put=-0.94295
Gamma:
call=0.017208 put=0.015870
Vega:
Call=6.065369 put=8.031475
Theta:
Call=-7.149208 put=-3.03558
Rho:
Call=0.87787 put=-12.502269

For sensitivity, we set the small change delta=1e^(-3), the sensitivity output is shown below:

```
Sensitivity to dividend amount: Call: -0.013, Put: 0.945
```

Problem2

To calculate the Mean, var and es, we first define the function of implied volatility calculation and simulate portfolio values with normal distribution. To simulate the data, we use scipy.stats.norm method. The output is shown below:

| Portfolio | Mean | VaR | ES |
|---|---|---|---|
| **Call** | 2.364587 | -4.220267 | -5.159150 |
| **CallSpread** | -0.736854 | -3.201527 | -3.670549 |
| **CoveredCall** | -2.370038 | -12.486353 | -17.105050 |
| **ProtectedPut** | 6.955003 | -0.464203 | -1.933145 |
| **Put** | 7.284255 | 0.735192 | -0.879949 |
| **PutSpread** | 2.112575 | -0.029255 | -0.813042 |
| **Stock** | 0.599027 | -14.682045 | -20.050934 |
| **Straddle** | 9.648843 | 8.042447 | 8.031390 |
| **SynLong** | -4.919668 | -20.443891 | -25.697596 |

Comparing with last week's result, var and es are much smaller than last week's data, which means american options have lower risk than european options. And have lower expected loss than european options.

Problem3

We first load the data and calculate arithmetic expected returns in past 10 years.Next, using geometric mean returns and calculate the covariance of different stocks. Then applying the lecture note sharp ratio content, we find the portfolio with highest sharp ratio.