**Problem 1**

- Current Stock Price $165
- Strike Price $165
- Current Date 03/13/2022
- Options Expiration Date 04/15/2022
- Risk Free Rate of 4.25%
- Continuously Compounding Coupon of 0.53%

Implement the closed form greeks for GBSM. Implement a finite difference derivative calculation. Compare the values between the two methods for both a call and a put.

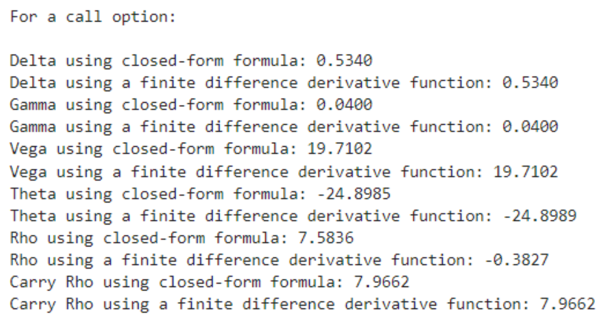
Implement the binomial tree valuation for American options with and without discrete dividends. Assume the stock above:

- Pays dividend on 4/11/2022 of $0.88

Calculate the value of the call and the put. Calculate the Greeks of each.

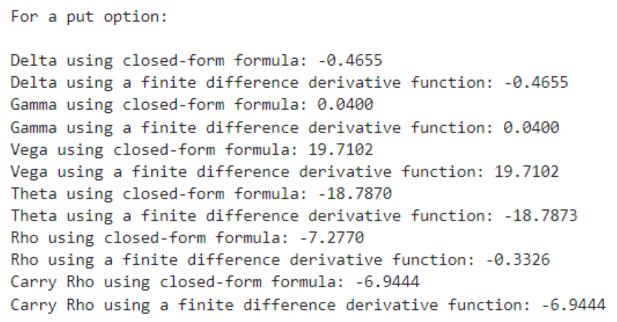
What is the sensitivity of the put and call to a change in the dividend amount?

In this question, I first defined several functions in my library to used closed-form formulas to calculate greeks for GBSM, and defined another function used to calculate (partial) derivatives (at most 2nd order) of given functions. Then, I ran defined functions both using closed-form formulas and finite difference derivative methods to derive greeks for a European call option:



```
For a call option:

Delta using closed-form formula: 0.5340
Delta using a finite difference derivative function: 0.5340
Gamma using closed-form formula: 0.0400
Gamma using a finite difference derivative function: 0.0400
Vega using closed-form formula: 19.7102
Vega using a finite difference derivative function: 19.7102
Theta using closed-form formula: -24.8985
Theta using a finite difference derivative function: -24.8989
Rho using closed-form formula: 7.5836
Rho using a finite difference derivative function: -0.3827
Carry Rho using closed-form formula: 7.9662
Carry Rho using a finite difference derivative function: 7.9662
```

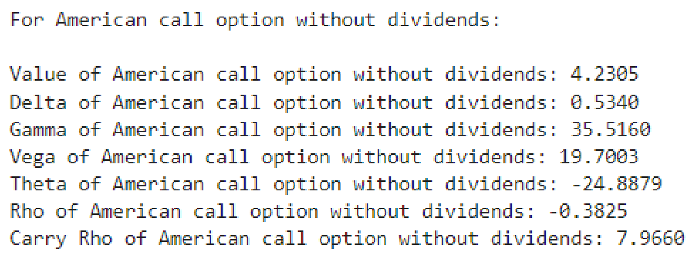
As seen from above, values of greeks are nearly the same using closed-form formulas and finite derivative functions except for Rho, measuring how values of options change with respect to risk free rates. Difference between results of Rho using two different methods is due to the fact that closed-form formulas for Rho are for GBSM when risk free rates equal to b, costs of carry. In this case, the risk free rate differs from b, so results using closed-form formulas are incorrect. Similarly, I did the same thing to derive greeks for a European put option:



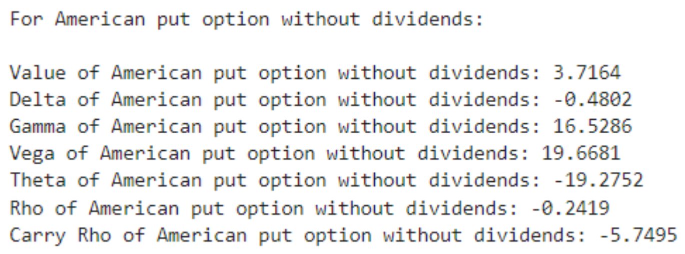
```
For a put option:

Delta using closed-form formula: -0.4655
Delta using a finite difference derivative function: -0.4655
Gamma using closed-form formula: 0.0400
Gamma using a finite difference derivative function: 0.0400
Vega using closed-form formula: 19.7102
Vega using a finite difference derivative function: 19.7102
Theta using closed-form formula: -18.7870
Theta using a finite difference derivative function: -18.7873
Rho using closed-form formula: -7.2770
Rho using a finite difference derivative function: -0.3326
Carry Rho using closed-form formula: -6.9444
Carry Rho using a finite difference derivative function: -6.9444
```

And, as shown above, results are all the same except for Rho. Again, values of Rho are different using two different methods, since the risk free rate is different from b in this case. Note that when using finite derivatives methods, I am calculating the central difference for accuracy at cost of speed.

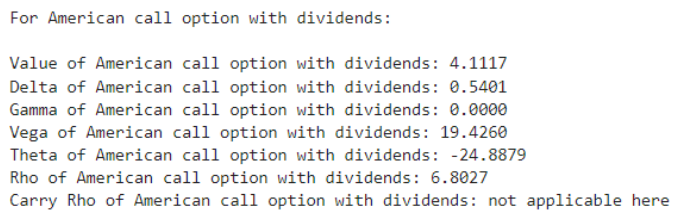
Followingly, I defined two new functions in my library, calculating values of American options without or with dividends. Applying those functions and functions using finite derivative methods to calculate greeks to this problem, I have:



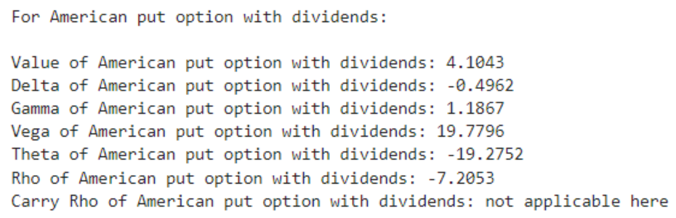
```
For American call option without dividends:

Value of American call option without dividends: 4.2305
Delta of American call option without dividends: 0.5340
Gamma of American call option without dividends: 35.5160
Vega of American call option without dividends: 19.7003
Theta of American call option without dividends: -24.8879
Rho of American call option without dividends: -0.3825
Carry Rho of American call option without dividends: 7.9660
```



```
For American put option without dividends:

Value of American put option without dividends: 3.7164
Delta of American put option without dividends: -0.4802
Gamma of American put option without dividends: 16.5286
Vega of American put option without dividends: 19.6681
Theta of American put option without dividends: -19.2752
Rho of American put option without dividends: -0.2419
Carry Rho of American put option without dividends: -5.7495
```



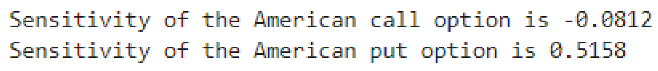
```
For American call option with dividends:

Value of American call option with dividends: 4.1117
Delta of American call option with dividends: 0.5401
Gamma of American call option with dividends: 0.0000
Vega of American call option with dividends: 19.4260
Theta of American call option with dividends: -24.8879
Rho of American call option with dividends: 6.8027
Carry Rho of American call option with dividends: not applicable here
```



```
For American put option with dividends:

Value of American put option with dividends: 4.1043
Delta of American put option with dividends: -0.4962
Gamma of American put option with dividends: 1.1867
Vega of American put option with dividends: 19.7796
Theta of American put option with dividends: -19.2752
Rho of American put option with dividends: -7.2053
Carry Rho of American put option with dividends: not applicable here
```

Note that Carry Rhos are not applicable to American options with dividends, since we are not using b, costs of carry, as a parameter while pricing these options. Therefore, partial derivatives do not exist.

Eventually, I am able to calculate sensitivity of both of those American options to changes in dividends with applications of finite difference derivative methods using central differences, and results are:



```
Sensitivity of the American call option is -0.0812
Sensitivity of the American put option is 0.5158
```

We may conclude from these numbers that as dividends increase, values of American call options decrease, while values of American put options increase.

**Problem 2**

Using the options portfolios from Problem3 last week (named problem2.csv in this week's repo) and assuming :

- American Options
- Current Date 03/03/2023
- Current AAPL price is 165
- Risk Free Rate of 4.25%
- Dividend Payment of $1.00 on 3/15/2023

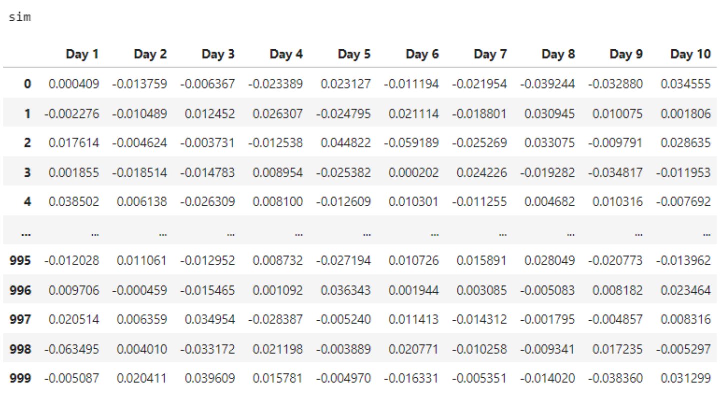
Using DailyPrices.csv. Fit a Normal distribution to AAPL returns – assume 0 mean return. Simulate AAPL returns 10 days ahead and apply those returns to the current AAPL price (above). Calculate Mean, VaR and ES.

Calculate VaR and ES using Delta-Normal.

Present all VaR and ES values a $ loss, not percentages.

Compare these results to last week's results.

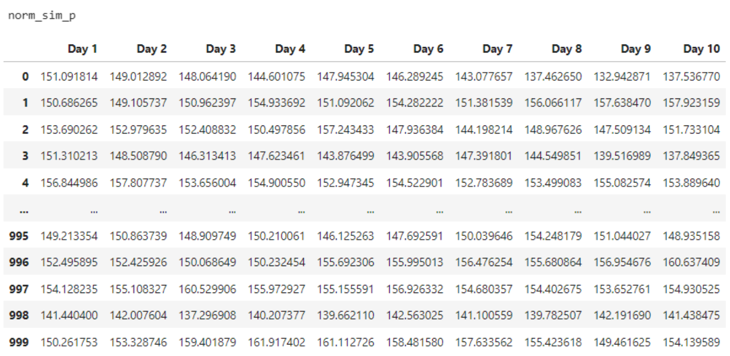
In this question, I first read data from DailyPrices.csv, and used defined function in my function library to calculate arithmetic returns of AAPL and demeaned those returns. Then, I fitted those returns with a normal distribution, and used the fitted distribution to simulate returns for 10 days ahead (1000 returns for each day) (if simulating more than 1000, the system becomes too slow):



```
sim
```

| | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Day 8 | Day 9 | Day 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000409 | -0.013759 | -0.006367 | -0.023389 | 0.023127 | -0.011194 | -0.021954 | -0.039244 | -0.032880 | 0.034555 |
| **1** | -0.002276 | -0.010489 | 0.012452 | 0.026307 | -0.024795 | 0.021114 | -0.018801 | 0.030945 | 0.010075 | 0.001806 |
| **2** | 0.017614 | -0.004624 | -0.003731 | -0.012538 | 0.044822 | -0.059189 | -0.025269 | 0.033075 | -0.009791 | 0.028635 |
| **3** | 0.001855 | -0.018514 | -0.014783 | 0.008954 | -0.025382 | 0.000202 | 0.024226 | -0.019282 | -0.034817 | -0.011953 |
| **4** | 0.038502 | 0.006138 | -0.026309 | 0.008100 | -0.012609 | 0.010301 | -0.011255 | 0.004682 | 0.010316 | -0.007692 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **995** | -0.012028 | 0.011061 | -0.012952 | 0.008732 | -0.027194 | 0.010726 | 0.015891 | 0.028049 | -0.020773 | -0.013962 |
| **996** | 0.009706 | -0.000459 | -0.015465 | 0.001092 | 0.036343 | 0.001944 | 0.003085 | -0.005083 | 0.008182 | 0.023464 |
| **997** | 0.020514 | 0.006359 | 0.034954 | -0.028387 | -0.005240 | 0.011413 | -0.014312 | -0.001795 | -0.004857 | 0.008316 |
| **998** | -0.063495 | 0.004010 | -0.033172 | 0.021198 | -0.003889 | 0.020771 | -0.010258 | -0.009341 | 0.017235 | -0.005297 |
| **999** | -0.005087 | 0.020411 | 0.039609 | 0.015781 | -0.004970 | -0.016331 | -0.005351 | -0.014020 | -0.038360 | 0.031299 |

With returns above, I am able to calculate prices of 10 days ahead of AAPL, and values of portfolios of 10 days ahead could be derived from those prices:



```
norm_sim_p
```

| | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Day 8 | Day 9 | Day 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 151.091814 | 149.012892 | 148.064190 | 144.601075 | 147.945304 | 146.289245 | 143.077657 | 137.462650 | 132.942871 | 137.536770 |
| **1** | 150.686265 | 149.105737 | 150.962397 | 154.933692 | 151.092062 | 154.282222 | 151.381539 | 156.066117 | 157.638470 | 157.923159 |
| **2** | 153.690262 | 152.979635 | 152.408832 | 150.497856 | 157.243433 | 147.936384 | 144.198214 | 148.967626 | 147.509134 | 151.733104 |
| **3** | 151.310213 | 148.508790 | 146.313413 | 147.623461 | 143.876499 | 143.905568 | 147.391801 | 144.549851 | 139.516989 | 137.849365 |
| **4** | 156.844986 | 157.807737 | 153.656004 | 154.900550 | 152.947345 | 154.522901 | 152.783689 | 153.499083 | 155.082574 | 153.889640 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **995** | 149.213354 | 150.863739 | 148.909749 | 150.210061 | 146.125263 | 147.692591 | 150.039646 | 154.248179 | 151.044027 | 148.935158 |
| **996** | 152.495895 | 152.425926 | 150.068649 | 150.232454 | 155.692306 | 155.995013 | 156.476254 | 155.680864 | 156.954676 | 160.637409 |
| **997** | 154.128235 | 155.108327 | 160.529906 | 155.972927 | 155.155591 | 156.926332 | 154.680357 | 154.402675 | 153.652761 | 154.930525 |
| **998** | 141.440400 | 142.007604 | 137.296908 | 140.207377 | 139.662110 | 142.563025 | 141.100559 | 139.782507 | 142.191690 | 141.438475 |
| **999** | 150.261753 | 153.328746 | 159.401879 | 161.917402 | 161.112726 | 158.481580 | 157.633562 | 155.423618 | 149.461625 | 154.139589 |

```
norm_sim_10port
```

| Portfolio | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 990 | 991 | 992 | 993 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Call** | 1.279925 | 10.408580 | 6.438092 | 1.337940 | 7.718334 | 13.286582 | 14.653486 | 9.162637 | 0.926025 | 12.961337 | ... | 13.573441 | 4.391655 | 4.766921 | 9.150402 |
| **CallSpread** | 1.156839 | 6.399884 | 4.578454 | 1.202315 | 5.269723 | 7.359688 | 7.696949 | 5.877381 | 0.847097 | 7.279439 | ... | 7.430466 | 3.404353 | 3.634823 | 5.871664 |
| **CoveredCall** | 137.103150 | 151.135751 | 148.085579 | 137.393342 | 149.261653 | 152.482575 | 153.005164 | 150.411037 | 135.309348 | 152.358229 | ... | 152.592246 | 145.426129 | 145.993961 | 150.403920 |
| **ProtectedPut** | 145.973482 | 157.923159 | 151.733104 | 145.973482 | 153.889640 | 161.730305 | 163.463240 | 156.138822 | 145.973482 | 161.317965 | ... | 162.093980 | 147.656405 | 148.435142 | 156.121299 |
| **Put** | 13.517120 | 1.340532 | 3.331993 | 13.221450 | 2.456647 | 0.687928 | 0.492126 | 1.772376 | 15.387702 | 0.744799 | ... | 0.637827 | 5.462221 | 4.985664 | 1.777650 |
| **PutSpread** | 10.079499 | 1.340532 | 3.331993 | 10.096424 | 2.456647 | 0.687928 | 0.492126 | 1.772376 | 10.017791 | 0.744799 | ... | 0.637827 | 5.462221 | 4.985664 | 1.777650 |
| **Stock** | 137.536770 | 157.923159 | 151.733104 | 137.849365 | 153.889640 | 161.730305 | 163.463240 | 156.138822 | 135.604480 | 161.317965 | ... | 162.093980 | 147.656405 | 148.435142 | 156.121299 |
| **Straddle** | 14.797046 | 11.749111 | 9.770085 | 14.559390 | 10.174981 | 13.974511 | 15.145611 | 10.935014 | 16.313727 | 13.706136 | ... | 14.211269 | 9.853876 | 9.752585 | 10.928052 |
| **SynLong** | -12.237195 | 9.068048 | 3.106099 | -11.883510 | 5.261686 | 12.598654 | 14.161360 | 7.390261 | -14.461677 | 12.216538 | ... | 12.935614 | -1.070566 | -0.218743 | 7.372752 |

Eventually, I could sort these values of portfolios and calculate 5% VaR and ES:

```
resulting_mat
```

| Portfolio | Current Value (on 2023/3/3) | Mean of Portfolio Value($) | Mean of Losses/Gains($) | VaR($) | ES($) |
|---|---|---|---|---|---|
| **Call** | 6.80 | 7.612283 | -0.812283 | 6.018437 | 6.330911 |
| **CallSpread** | 9.01 | 4.506905 | 4.503095 | 8.282189 | 8.567421 |
| **CoveredCall** | 155.08 | 146.279029 | 8.800971 | 20.911491 | 24.319383 |
| **ProtectedPut** | 154.04 | 153.391203 | 0.648797 | 8.066518 | 8.066518 |
| **Put** | 4.85 | 5.491430 | -0.641430 | 4.722136 | 4.788129 |
| **PutSpread** | 6.69 | 4.580030 | 2.109970 | 6.562136 | 6.628129 |
| **Stock** | 151.03 | 151.262608 | -0.232608 | 16.636580 | 20.147335 |
| **Straddle** | 11.65 | 13.103713 | -1.453713 | 1.963359 | 1.983951 |
| **SynLong** | 11.65 | 2.120853 | 9.529147 | 27.453671 | 31.271717 |

Note that in this matrix (excluding column "Mean of Portfolio Value($)" and "Current Value (on 2023/3/3)"), positive values represent losses, and negative values represent gains, since I subtracted simulated portfolio values from portfolio values on current date before displaying results in the matrix. Finishing with Monte Carlo Normal method, I switched to Delta Normal method to calculate VaR and ES. I first derived the first derivative of values of American options with respect to underlying prices:

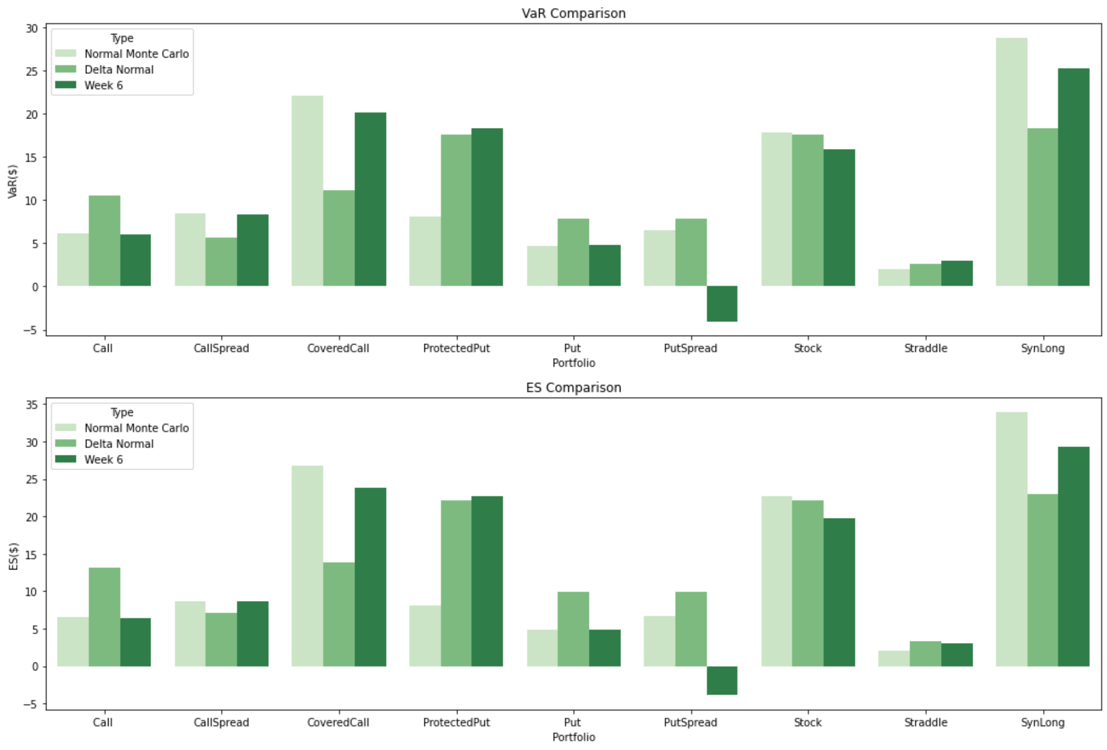
```
deltas

array([ 0.59365644, -0.44571621,  0.59365644, -0.44571621,  0.59365644,
        0.27109813, -0.44571621,  0.        ,  1.        ,  0.59365644,
       -0.44571621,  1.        ,  0.37167166,  1.        ,  0.        ])
```

which were used to compute gradients used in Delta Normal VaR and ES calculation. With derived gradients, I am able to calculate 5% VaR and ES for the portfolio together with the standard deviations of returns of AAPL:

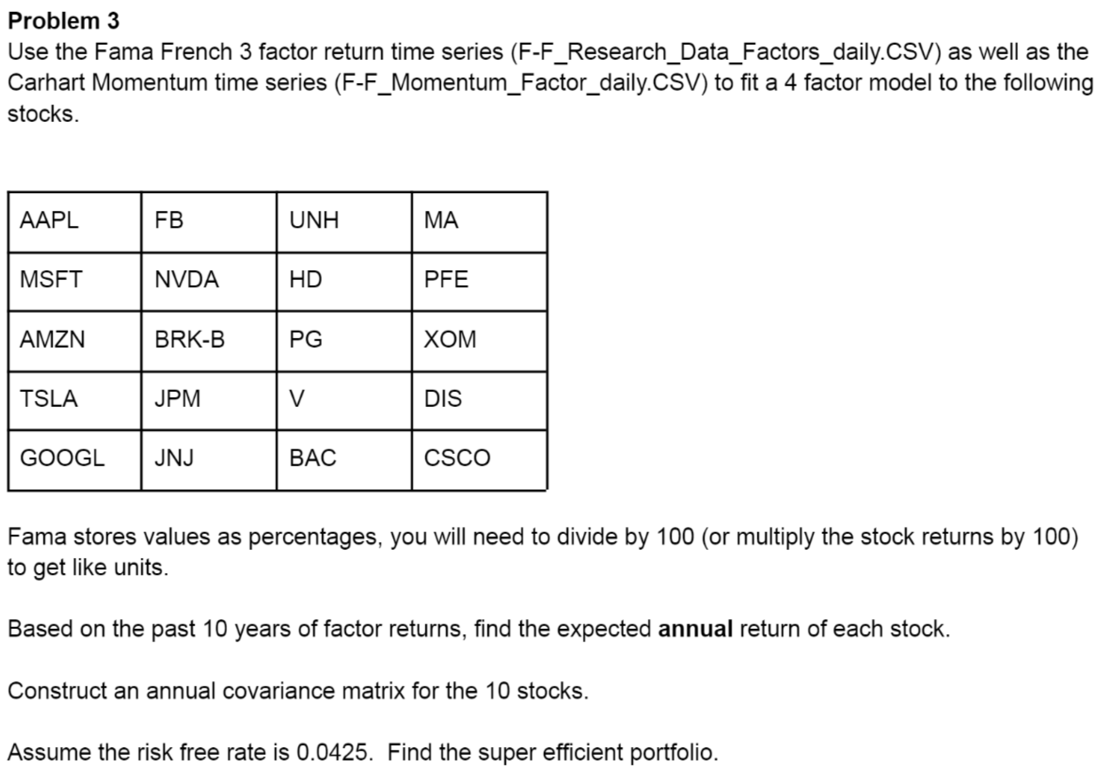
```
resulting_mat_del
```

| Portfolio | Mean($) | VaR($) | ES($) |
|---|---|---|---|
| **Call** | 0 | 10.464460 | 13.122855 |
| **CallSpread** | 0 | 5.685778 | 7.130194 |
| **CoveredCall** | 0 | 11.075626 | 13.889282 |
| **ProtectedPut** | 0 | 17.627131 | 22.105134 |
| **Put** | 0 | 7.856698 | 9.852616 |
| **PutSpread** | 0 | 7.856698 | 9.852616 |
| **Stock** | 0 | 17.627131 | 22.105134 |
| **Straddle** | 0 | 2.607762 | 3.270239 |
| **SynLong** | 0 | 18.321158 | 22.975471 |

Note that here I set all means to zero, as shown in the table , since means do not make sense here. With these results, a comparison could be done. A graph is drawn to put different VaRs and ES for each portfolio using different methods together, as shown below:



As seen above, values of American options, calculated in this week's problem, are typically higher than European options, derived from last week's assignment (there are exceptions - puts that are highly likely to not be available for exercise are having lower values). High returns/values are usually associated with higher risks, explaining why VaRs and ES are typically higher for American options (similarly, there are exceptions).


**Problem 3**
Use the Fama French 3 factor return time series (F-F_Research_Data_Factors_daily.CSV) as well as the Carhart Momentum time series (F-F_Momentum_Factor_daily.CSV) to fit a 4 factor model to the following stocks.

| | | | |
|---|---|---|---|
| AAPL | FB | UNH | MA |
| MSFT | NVDA | HD | PFE |
| AMZN | BRK-B | PG | XOM |
| TSLA | JPM | V | DIS |
| GOOGL | JNJ | BAC | CSCO |

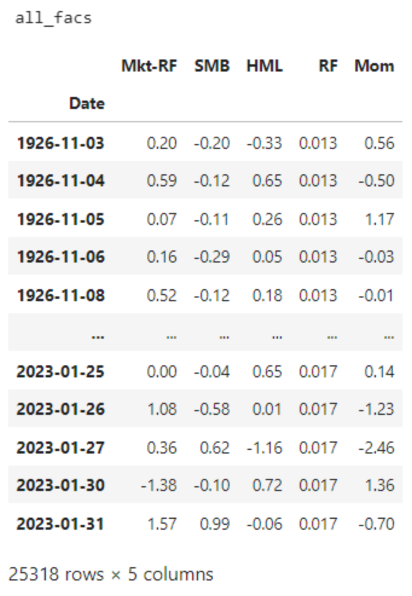
Fama stores values as percentages, you will need to divide by 100 (or multiply the stock returns by 100) to get like units.

Based on the past 10 years of factor returns, find the expected **annual** return of each stock.

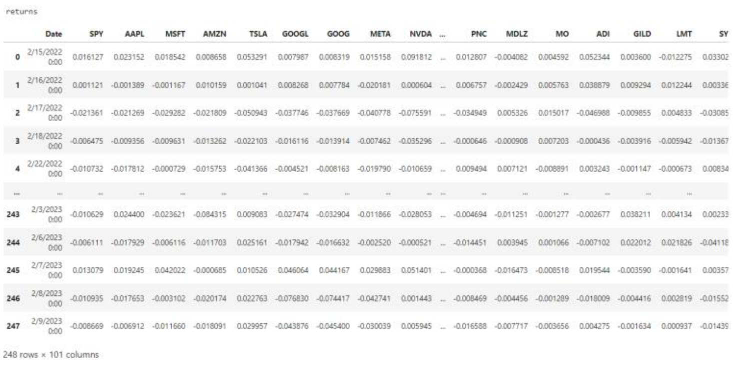
Construct an annual covariance matrix for the 10 stocks.

Assume the risk free rate is 0.0425. Find the super efficient portfolio.

In this question, I first read provided data, and combined those factor data in a single dataframe for later use (note that F-F_Research_Data_Factors_daily.CSV and F-F_Momentum_Factor_daily.CSV have different spectrums of time – we only keep overlapping data):


```
all_facs

            Mkt-RF   SMB    HML     RF    Mom
      Date
1926-11-03    0.20  -0.20  -0.33  0.013   0.56
1926-11-04    0.59  -0.12   0.65  0.013  -0.50
1926-11-05    0.07  -0.11   0.26  0.013   1.17
1926-11-06    0.16  -0.29   0.05  0.013  -0.03
1926-11-08    0.52  -0.12   0.18  0.013  -0.01
       ...     ...    ...    ...    ...    ...
2023-01-25    0.00  -0.04   0.65  0.017   0.14
2023-01-26    1.08  -0.58   0.01  0.017  -1.23
2023-01-27    0.36   0.62  -1.16  0.017  -2.46
2023-01-30   -1.38  -0.10   0.72  0.017   1.36
2023-01-31    1.57   0.99  -0.06  0.017  -0.70

25318 rows × 5 columns
```

Then, I used a defined function in my functions library to calculate returns of stocks with prices provided in DailyPrices.csv (part of the resulting matrix due to too many columns):



```
returns
```

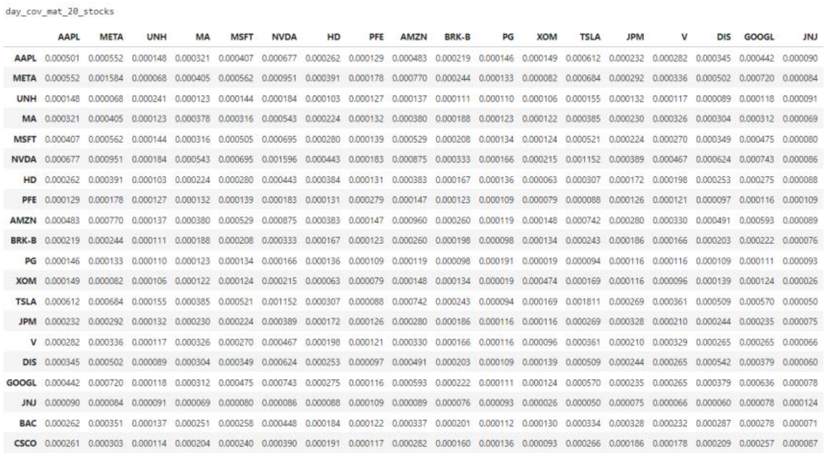
| | Date | SPY | AAPL | MSFT | AMZN | TSLA | GOOGL | GOOG | META | NVDA | ... | PNC | MDLZ | MO | ADI | GILD | LMT | SY |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2/15/2022 0:00 | 0.016127 | 0.023152 | 0.018542 | 0.008658 | 0.053291 | 0.007987 | 0.008319 | 0.015158 | 0.091812 | ... | 0.012807 | -0.004082 | 0.004592 | 0.052344 | 0.003600 | -0.012275 | 0.03302 |
| **1** | 2/16/2022 0:00 | 0.001121 | -0.001389 | -0.001167 | 0.010159 | 0.001041 | 0.008268 | 0.007784 | -0.020181 | 0.000604 | ... | 0.006757 | -0.002429 | 0.005763 | 0.038879 | 0.009294 | 0.012244 | 0.00336 |
| **2** | 2/17/2022 0:00 | -0.021361 | -0.021269 | -0.029282 | -0.021809 | -0.050943 | -0.037746 | -0.037669 | -0.040778 | -0.075591 | ... | -0.034949 | 0.005326 | 0.015017 | -0.046988 | -0.009855 | 0.004833 | -0.03085 |
| **3** | 2/18/2022 0:00 | -0.006475 | -0.009356 | -0.009631 | -0.013262 | -0.022103 | -0.016116 | -0.013914 | -0.007462 | -0.035296 | ... | -0.000646 | -0.000908 | 0.007203 | -0.000436 | -0.003916 | -0.005942 | -0.01367 |
| **4** | 2/22/2022 0:00 | -0.010732 | -0.017812 | -0.000729 | -0.015753 | -0.041366 | -0.004521 | -0.008163 | -0.019790 | -0.010659 | ... | 0.009494 | 0.007121 | -0.008891 | 0.003243 | -0.001147 | -0.000673 | 0.00834 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **243** | 2/3/2023 0:00 | -0.010629 | 0.024400 | -0.023621 | -0.084315 | 0.009083 | -0.027474 | -0.032904 | -0.011866 | -0.028053 | ... | -0.004694 | -0.011251 | -0.001277 | -0.002677 | 0.038211 | 0.004134 | 0.00233 |
| **244** | 2/6/2023 0:00 | -0.006111 | -0.017929 | -0.006116 | -0.011703 | 0.025161 | -0.017942 | -0.016632 | -0.002520 | -0.000521 | ... | -0.014451 | 0.003945 | 0.001066 | -0.007102 | 0.022012 | 0.021826 | -0.04118 |
| **245** | 2/7/2023 0:00 | 0.013079 | 0.019245 | 0.042022 | -0.000685 | 0.010526 | 0.046064 | 0.044167 | 0.029883 | 0.051401 | ... | -0.000368 | -0.016473 | -0.008518 | 0.019544 | -0.003590 | -0.001641 | 0.00357 |
| **246** | 2/8/2023 0:00 | -0.010935 | -0.017653 | -0.003102 | -0.020174 | 0.022763 | -0.076830 | -0.074417 | -0.042741 | 0.001443 | ... | -0.008469 | -0.004456 | -0.001289 | -0.018009 | -0.004416 | 0.002819 | -0.01552 |
| **247** | 2/9/2023 0:00 | -0.008669 | -0.006912 | -0.011660 | -0.018091 | 0.029957 | -0.043876 | -0.045400 | -0.030039 | 0.005945 | ... | -0.016588 | -0.007717 | -0.003656 | 0.004275 | -0.001634 | 0.000937 | -0.01439 |

248 rows × 101 columns

With these input data read, I filtered and processed these data so that they could be applicable later to fit the model as I would like to – preparing all factor data as Xs and (stock returns – risk free rates) as Ys that would be used in regression. Note that I multiply returns by 100 so that they are in the format of percentage, since factor data are all stored as percentages (in fact, returns in the screenshot above have already been multiplied by 100). Then, I employed "LinearRegression" function included in sklearn, a statistical package in Python, to run multivariate linear regression over my Xs and Ys. With estimated betas, I am able to calculate daily expected returns of each stock based on average factor returns of the past 10 years, and annualize them by multiplying daily returns by the number of trading days; therefore, resulting returns are (note that returns are expressed in percentages here):

```
avg_stock_returns

array([ 7.26661346,  1.89670079, 10.71677668,  9.58514253,  7.1843535 ,
       13.52537887,  5.25911198,  3.37573392, -0.45663866,  5.79973827,
        3.56429296, 21.91349254,  0.13493969,  4.97885092, 10.12018843,
       -5.12721612,  0.17313769,  5.08874871, -3.07512551,  6.4660114 ])
```

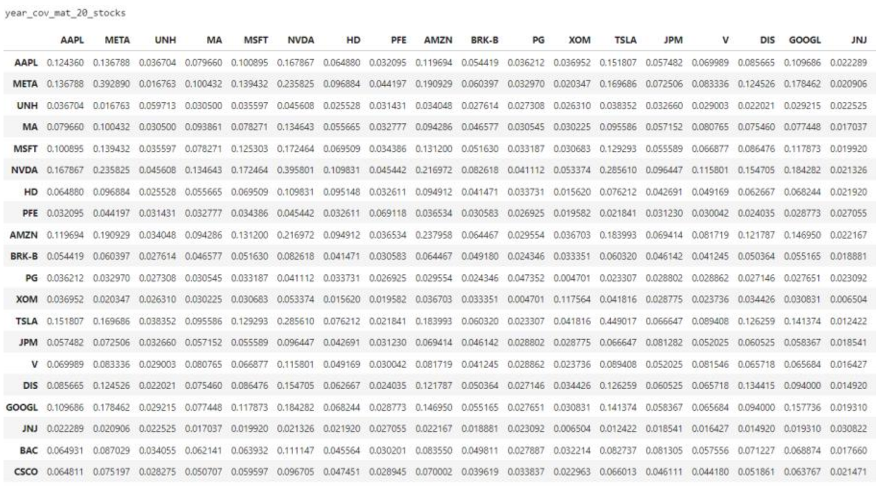
Then, I took advantage of another function "exwCovMat" defined in my library to calculate covariance matrix. With equally weighted returns, I derived the daily covariance matrix (part of the matrix):



```
day_cov_mat_20_stocks
```

| | AAPL | META | UNH | MA | MSFT | NVDA | HD | PFE | AMZN | BRK-B | PG | XOM | TSLA | JPM | V | DIS | GOOGL | JNJ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AAPL** | 0.000501 | 0.000552 | 0.000148 | 0.000321 | 0.000407 | 0.000677 | 0.000262 | 0.000129 | 0.000483 | 0.000219 | 0.000146 | 0.000149 | 0.000612 | 0.000232 | 0.000282 | 0.000345 | 0.000442 | 0.000090 |
| **META** | 0.000552 | 0.001584 | 0.000068 | 0.000405 | 0.000562 | 0.000951 | 0.000391 | 0.000178 | 0.000770 | 0.000244 | 0.000133 | 0.000082 | 0.000684 | 0.000292 | 0.000336 | 0.000502 | 0.000720 | 0.000084 |
| **UNH** | 0.000148 | 0.000068 | 0.000241 | 0.000123 | 0.000144 | 0.000184 | 0.000103 | 0.000127 | 0.000137 | 0.000111 | 0.000110 | 0.000106 | 0.000155 | 0.000132 | 0.000117 | 0.000089 | 0.000118 | 0.000091 |
| **MA** | 0.000321 | 0.000405 | 0.000123 | 0.000378 | 0.000316 | 0.000543 | 0.000224 | 0.000132 | 0.000380 | 0.000188 | 0.000123 | 0.000122 | 0.000385 | 0.000230 | 0.000326 | 0.000304 | 0.000312 | 0.000069 |
| **MSFT** | 0.000407 | 0.000562 | 0.000144 | 0.000316 | 0.000505 | 0.000695 | 0.000280 | 0.000139 | 0.000529 | 0.000208 | 0.000134 | 0.000124 | 0.000521 | 0.000224 | 0.000270 | 0.000349 | 0.000475 | 0.000080 |
| **NVDA** | 0.000677 | 0.000951 | 0.000184 | 0.000543 | 0.000695 | 0.001596 | 0.000443 | 0.000183 | 0.000875 | 0.000333 | 0.000166 | 0.000215 | 0.001152 | 0.000389 | 0.000467 | 0.000624 | 0.000743 | 0.000086 |
| **HD** | 0.000262 | 0.000391 | 0.000103 | 0.000224 | 0.000280 | 0.000443 | 0.000384 | 0.000131 | 0.000383 | 0.000167 | 0.000136 | 0.000063 | 0.000307 | 0.000172 | 0.000198 | 0.000253 | 0.000275 | 0.000088 |
| **PFE** | 0.000129 | 0.000178 | 0.000127 | 0.000132 | 0.000139 | 0.000183 | 0.000131 | 0.000279 | 0.000147 | 0.000123 | 0.000109 | 0.000079 | 0.000088 | 0.000126 | 0.000121 | 0.000097 | 0.000116 | 0.000109 |
| **AMZN** | 0.000483 | 0.000770 | 0.000137 | 0.000380 | 0.000529 | 0.000875 | 0.000383 | 0.000147 | 0.000960 | 0.000260 | 0.000119 | 0.000148 | 0.000742 | 0.000280 | 0.000330 | 0.000491 | 0.000593 | 0.000089 |
| **BRK-B** | 0.000219 | 0.000244 | 0.000111 | 0.000188 | 0.000208 | 0.000333 | 0.000167 | 0.000123 | 0.000260 | 0.000198 | 0.000098 | 0.000134 | 0.000243 | 0.000186 | 0.000166 | 0.000203 | 0.000222 | 0.000076 |
| **PG** | 0.000146 | 0.000133 | 0.000110 | 0.000123 | 0.000134 | 0.000166 | 0.000136 | 0.000109 | 0.000119 | 0.000098 | 0.000191 | 0.000019 | 0.000094 | 0.000116 | 0.000116 | 0.000109 | 0.000111 | 0.000093 |
| **XOM** | 0.000149 | 0.000082 | 0.000106 | 0.000122 | 0.000124 | 0.000215 | 0.000063 | 0.000079 | 0.000148 | 0.000134 | 0.000019 | 0.000474 | 0.000169 | 0.000116 | 0.000096 | 0.000139 | 0.000124 | 0.000026 |
| **TSLA** | 0.000612 | 0.000684 | 0.000155 | 0.000385 | 0.000521 | 0.001152 | 0.000307 | 0.000088 | 0.000742 | 0.000243 | 0.000094 | 0.000169 | 0.001811 | 0.000269 | 0.000361 | 0.000509 | 0.000570 | 0.000050 |
| **JPM** | 0.000232 | 0.000292 | 0.000132 | 0.000230 | 0.000224 | 0.000389 | 0.000172 | 0.000126 | 0.000280 | 0.000186 | 0.000116 | 0.000116 | 0.000269 | 0.000328 | 0.000210 | 0.000244 | 0.000235 | 0.000075 |
| **V** | 0.000282 | 0.000336 | 0.000117 | 0.000326 | 0.000270 | 0.000467 | 0.000198 | 0.000121 | 0.000330 | 0.000166 | 0.000116 | 0.000096 | 0.000361 | 0.000210 | 0.000329 | 0.000265 | 0.000265 | 0.000066 |
| **DIS** | 0.000345 | 0.000502 | 0.000089 | 0.000304 | 0.000349 | 0.000624 | 0.000253 | 0.000097 | 0.000491 | 0.000203 | 0.000109 | 0.000139 | 0.000509 | 0.000244 | 0.000265 | 0.000542 | 0.000379 | 0.000060 |
| **GOOGL** | 0.000442 | 0.000720 | 0.000118 | 0.000312 | 0.000475 | 0.000743 | 0.000275 | 0.000116 | 0.000593 | 0.000222 | 0.000111 | 0.000124 | 0.000570 | 0.000235 | 0.000265 | 0.000379 | 0.000636 | 0.000078 |
| **JNJ** | 0.000090 | 0.000084 | 0.000091 | 0.000069 | 0.000080 | 0.000086 | 0.000088 | 0.000109 | 0.000089 | 0.000076 | 0.000093 | 0.000026 | 0.000050 | 0.000075 | 0.000066 | 0.000060 | 0.000078 | 0.000124 |
| **BAC** | 0.000262 | 0.000351 | 0.000137 | 0.000251 | 0.000258 | 0.000448 | 0.000184 | 0.000122 | 0.000337 | 0.000201 | 0.000112 | 0.000130 | 0.000334 | 0.000328 | 0.000232 | 0.000287 | 0.000278 | 0.000071 |
| **CSCO** | 0.000261 | 0.000303 | 0.000114 | 0.000204 | 0.000240 | 0.000390 | 0.000191 | 0.000117 | 0.000282 | 0.000160 | 0.000136 | 0.000093 | 0.000266 | 0.000186 | 0.000178 | 0.000209 | 0.000257 | 0.000087 |

Similarly to returns, the covariance matrix is annualized by me through multiplying each element by the number of trading days, and the annual covariance matrix is:



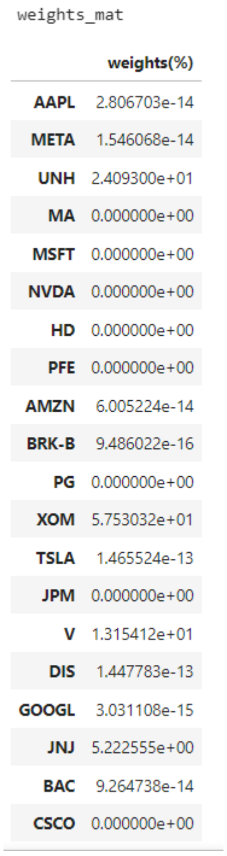
```
year_cov_mat_20_stocks
```

| | AAPL | META | UNH | MA | MSFT | NVDA | HD | PFE | AMZN | BRK-B | PG | XOM | TSLA | JPM | V | DIS | GOOGL | JNJ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AAPL** | 0.124360 | 0.136788 | 0.036704 | 0.079660 | 0.100895 | 0.167867 | 0.064880 | 0.032095 | 0.119694 | 0.054419 | 0.036212 | 0.036952 | 0.151807 | 0.057482 | 0.069989 | 0.085665 | 0.109686 | 0.022289 |
| **META** | 0.136788 | 0.392890 | 0.016763 | 0.100432 | 0.139432 | 0.235825 | 0.096884 | 0.044197 | 0.190929 | 0.060397 | 0.032970 | 0.020347 | 0.169686 | 0.072506 | 0.083336 | 0.124526 | 0.178462 | 0.020906 |
| **UNH** | 0.036704 | 0.016763 | 0.059713 | 0.030500 | 0.035597 | 0.045608 | 0.025528 | 0.031431 | 0.034048 | 0.027614 | 0.027308 | 0.026310 | 0.038352 | 0.032660 | 0.029003 | 0.022021 | 0.029215 | 0.022525 |
| **MA** | 0.079660 | 0.100432 | 0.030500 | 0.093861 | 0.078271 | 0.134643 | 0.055665 | 0.032777 | 0.094286 | 0.046577 | 0.030545 | 0.030225 | 0.095586 | 0.057152 | 0.080765 | 0.075460 | 0.077448 | 0.017037 |
| **MSFT** | 0.100895 | 0.139432 | 0.035597 | 0.078271 | 0.125303 | 0.172464 | 0.069509 | 0.034386 | 0.131200 | 0.051630 | 0.033187 | 0.030683 | 0.129293 | 0.055589 | 0.066877 | 0.086476 | 0.117873 | 0.019920 |
| **NVDA** | 0.167867 | 0.235825 | 0.045608 | 0.134643 | 0.172464 | 0.395801 | 0.109831 | 0.045442 | 0.216972 | 0.082618 | 0.041112 | 0.053374 | 0.285610 | 0.096447 | 0.115801 | 0.154705 | 0.184282 | 0.021326 |
| **HD** | 0.064880 | 0.096884 | 0.025528 | 0.055665 | 0.069509 | 0.109831 | 0.095148 | 0.032611 | 0.094912 | 0.041471 | 0.033731 | 0.015620 | 0.076212 | 0.042691 | 0.049169 | 0.062667 | 0.068244 | 0.021920 |
| **PFE** | 0.032095 | 0.044197 | 0.031431 | 0.032777 | 0.034386 | 0.045442 | 0.032611 | 0.069118 | 0.036534 | 0.030583 | 0.026925 | 0.019582 | 0.021841 | 0.031230 | 0.030042 | 0.024035 | 0.028773 | 0.027055 |
| **AMZN** | 0.119694 | 0.190929 | 0.034048 | 0.094286 | 0.131200 | 0.216972 | 0.094912 | 0.036534 | 0.237958 | 0.064467 | 0.029554 | 0.036703 | 0.183993 | 0.069414 | 0.081719 | 0.121787 | 0.146950 | 0.022167 |
| **BRK-B** | 0.054419 | 0.060397 | 0.027614 | 0.046577 | 0.051630 | 0.082618 | 0.041471 | 0.030583 | 0.064467 | 0.049180 | 0.024346 | 0.033351 | 0.060320 | 0.046142 | 0.041245 | 0.050364 | 0.055165 | 0.018881 |
| **PG** | 0.036212 | 0.032970 | 0.027308 | 0.030545 | 0.033187 | 0.041112 | 0.033731 | 0.026925 | 0.029554 | 0.024346 | 0.047352 | 0.004701 | 0.023307 | 0.028802 | 0.028862 | 0.027146 | 0.027651 | 0.023092 |
| **XOM** | 0.036952 | 0.020347 | 0.026310 | 0.030225 | 0.030683 | 0.053374 | 0.015620 | 0.019582 | 0.036703 | 0.033351 | 0.004701 | 0.117564 | 0.041816 | 0.028775 | 0.023736 | 0.034426 | 0.030831 | 0.006504 |
| **TSLA** | 0.151807 | 0.169686 | 0.038352 | 0.095586 | 0.129293 | 0.285610 | 0.076212 | 0.021841 | 0.183993 | 0.060320 | 0.023307 | 0.041816 | 0.449017 | 0.066647 | 0.089408 | 0.126259 | 0.141374 | 0.012422 |
| **JPM** | 0.057482 | 0.072506 | 0.032660 | 0.057152 | 0.055589 | 0.096447 | 0.042691 | 0.031230 | 0.069414 | 0.046142 | 0.028802 | 0.028775 | 0.066647 | 0.081282 | 0.052025 | 0.060525 | 0.058367 | 0.018541 |
| **V** | 0.069989 | 0.083336 | 0.029003 | 0.080765 | 0.066877 | 0.115801 | 0.049169 | 0.030042 | 0.081719 | 0.041245 | 0.028862 | 0.023736 | 0.089408 | 0.052025 | 0.081546 | 0.065718 | 0.065684 | 0.016427 |
| **DIS** | 0.085665 | 0.124526 | 0.022021 | 0.075460 | 0.086476 | 0.154705 | 0.062667 | 0.024035 | 0.121787 | 0.050364 | 0.027146 | 0.034426 | 0.126259 | 0.060525 | 0.065718 | 0.134415 | 0.094000 | 0.014920 |
| **GOOGL** | 0.109686 | 0.178462 | 0.029215 | 0.077448 | 0.117873 | 0.184282 | 0.068244 | 0.028773 | 0.146950 | 0.055165 | 0.027651 | 0.030831 | 0.141374 | 0.058367 | 0.065684 | 0.094000 | 0.157736 | 0.019310 |
| **JNJ** | 0.022289 | 0.020906 | 0.022525 | 0.017037 | 0.019920 | 0.021326 | 0.021920 | 0.027055 | 0.022167 | 0.018881 | 0.023092 | 0.006504 | 0.012422 | 0.018541 | 0.016427 | 0.014920 | 0.019310 | 0.030822 |
| **BAC** | 0.064931 | 0.087029 | 0.034055 | 0.062141 | 0.063932 | 0.111147 | 0.045564 | 0.030201 | 0.083550 | 0.049811 | 0.027887 | 0.032214 | 0.082737 | 0.081305 | 0.057556 | 0.071227 | 0.068874 | 0.017660 |
| **CSCO** | 0.064811 | 0.075197 | 0.028275 | 0.050707 | 0.059597 | 0.096705 | 0.047451 | 0.028945 | 0.070002 | 0.039619 | 0.033837 | 0.022963 | 0.066013 | 0.046111 | 0.044180 | 0.051861 | 0.063767 | 0.021471 |

With expected returns and covariances of those 20 stocks derived, I started to derive the optimized portfolio. I first defined a function for optimization with everything set up, including constraints, bounds of changeable items, and initial guesses, and input expected returns (expressed in percentage), covariances (expressed in percentage), and risk free rates (expressed in percentage) into the function. The optimized Sharpe Ratio is:

```
max_Sharpe

1.5550315075682315
```

And weights of each stock in the optimized portfolio is (note that I express all weights in percentages):



```
weights_mat
```

| | weights(%) |
|---|---|
| **AAPL** | 2.806703e-14 |
| **META** | 1.546068e-14 |
| **UNH** | 2.409300e+01 |
| **MA** | 0.000000e+00 |
| **MSFT** | 0.000000e+00 |
| **NVDA** | 0.000000e+00 |
| **HD** | 0.000000e+00 |
| **PFE** | 0.000000e+00 |
| **AMZN** | 6.005224e-14 |
| **BRK-B** | 9.486022e-16 |
| **PG** | 0.000000e+00 |
| **XOM** | 5.753032e+01 |
| **TSLA** | 1.465524e-13 |
| **JPM** | 0.000000e+00 |
| **V** | 1.315412e+01 |
| **DIS** | 1.447783e-13 |
| **GOOGL** | 3.031108e-15 |
| **JNJ** | 5.222555e+00 |
| **BAC** | 9.264738e-14 |
| **CSCO** | 0.000000e+00 |