Eric Shi

1003062552

APS 502

April 14, 2022

Computational Project Report

Refer to the Appendix for the entire MATLAB scripts for each problem in this computational assignment.

Problem 1 Part 1:

The goal is to optimize the cash matching problem by minimizing cost of the bond portfolio while meeting all obligations. This was done using a linear programming tool in MATLAB. Cash is allowed to be carried over. The objective function is defined in equation 1 below.

$$Optimal\ Portfolio\ Cost = \ minimize \sum_{i=1}^{n} P_i * x_i \tag{1}$$

Where n represents the number of available bonds to be invested in (given as 13), P_i represents the current price of one unit of bond i, and x_i represents the number of units of bond i to invest in. Table 1 below summarizes all the bond options available for this problem.

Table 1 – Bonds Available to be Purchased

Bond	Price ($)	Coupon ($/Year)	Face Value ($)	Maturity (Years)	Rating
1	108	10	100	6	B
2	94	7	100	6	B
3	99	8	100	6	B
4	92.7	6	100	5	B
5	96.6	7	100	5	B
6	95.9	6	100	4	B
7	92.9	5	100	4	A
8	110	10	100	3	A
9	104	8	100	3	A
10	101	6	100	3	A
11	107	10	100	2	A
12	102	7	100	2	A
13	95.2	0	100	1	A

The optimal portfolio cost is minimized by adjusting all x_i variables subject to the following constraints:

Table 2 – Summary of Cash Matching Constraints

Time (Years)	Constraint
1	$\sum_{i=1}^{12} C_i * x_i + 100 * x_{13} - z_1 \geq 500$
2	$\sum_{i=1}^{12} C_i * x_i + 100 * (x_{12} + x_{11}) + (1 + f_{1,2}) * z_1 - z_2 \geq 200$
3	$\sum_{i=1}^{10} C_i * x_i + 100 * (x_{10} + x_9 + x_8) + (1 + f_{2,3}) * z_2 - z_3 \geq 800$
4	$\sum_{i=1}^{7} C_i * x_i + 100 * (x_7 + x_6) + (1 + f_{3,4}) * z_3 - z_4 \geq 400$
5	$\sum_{i=1}^{5} C_i * x_i + 100 * (x_5 + x_4) + (1 + f_{4,5}) * z_4 - z_5 \geq 700$
6	$\sum_{i=1}^{3} C_i * x_i + 100 * (x_3 + x_2 + x_1) + (1 + f_{5,6}) * z_5 \geq 900$

Where C_i represents the coupon payment of bond i, z_t represents the excess cash to be reinvested and carried forward at time t, where t = 1, 2, 3, 4, 5, 6. $f_{t-1,t}$ represents the short forward rate for all t values. Where the short forward rates were determined using the given spot rates with equation 2 below.

$$f_{i,j} = \frac{(1 + s_j)^j}{(1 + s_i)^i} - 1 \qquad (2)$$

Where s_j is the spot rate for time period j, s_i is the spot rate for time period i, and in the case of short forward rates, $j = i + 1$.

The lower and upper bounds for each variable in the linear programming model are defined in equations 3 and 4 below.

$$0 \leq x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13} \leq \infty \qquad (3)$$

$$0 \leq z_1, z_2, z_3, z_4, z_5 \leq \infty \qquad (4)$$

Solving the above optimization problem with MATLAB's linprog function, the results are summarized in MATLAB's command window as follows.

Figure 1 – Optimal Cash Matching Bond Portfolio (No Restriction on B-rated Bonds)

```
Problem 1 Part 1
Investing in:
     8.1818 units of bond 1
     0.0000 units of bond 2
     0.0000 units of bond 3
     0.0000 units of bond 4
     5.7774 units of bond 5
     2.6202 units of bond 6
     0.0000 units of bond 7
     0.0000 units of bond 8
     6.1298 units of bond 9
     0.0000 units of bond 10
     0.1180 units of bond 11
     0.0000 units of bond 12
     3.1180 units of bond 13

Amount reinvested at each time period
     Date 1: $0.00
     Date 2: $0.00
     Date 3: $0.00
     Date 4: $0.00
     Date 5: $0.00

Total cost of the bond portfolio: $2639.97

Weight of the portfolio in B-rated bonds: 64.13%
```

Problem 1 Part 2:

Part 2 has near identical linear programming inputs as part 1 except there is one extra constraint to account for the upper limit of 50% on B-rated bonds. Note that no changes were made on the upper and lower bounds as those define limits for individual bonds, not for a sum of multiple bonds. The new inequality constraint is defined in equation 5 below.

$$\sum_{i=1}^{6} P_i * x_i - \sum_{i=7}^{13} P_i * x_i \leq 0 \tag{5}$$

Since bonds 1 to 6 are B-rated bonds, and bonds 7 to 13 are A-rated bonds, the total value invested in bonds 1 to 6 must be less than or equal to the total value invested in bonds 7 to 13. Equation 5 rearranges this explanation into a convenient form for MATLAB's linprog function. Inputting equation 5 into the Part 1 MATLAB script results in the following optimal portfolio.

Figure 2 – Optimal Cash Matching Bond Portfolio (50% Maximum in B-rated Bonds)

```
Problem 1 Part 2
Investing in:
     0.0000 units of bond 1
     8.4112 units of bond 2
     0.0000 units of bond 3
     0.0000 units of bond 4
     5.5027 units of bond 5
     0.0000 units of bond 6
     3.3565 units of bond 7
     0.0000 units of bond 8
     6.3502 units of bond 9
     0.0000 units of bond 10
     0.3184 units of bond 11
     0.0000 units of bond 12
     3.3184 units of bond 13

Amount reinvested at each time period
     Date 1: $0.00
     Date 2: $0.00
     Date 3: $0.00
     Date 4: $49.83
     Date 5: $0.00

Total cost of the bond portfolio: $2644.42

Weight of the portfolio in B-rated bonds: 50.00%
```

Note that the weight of the portfolio in B-rated bonds is exactly at the upper limit. This implies that the overall optimal portfolio without the constraint is above 50% invested in B-rated bonds, which makes sense because B-rated bonds often have a more attractive reward than A-rated bonds due to a higher risk of default. The solution in Part 1 also reiterates the above logic as the weight of the portfolio in B-rated bonds is greater than 50%, and the cost of the overall bond portfolio is less in Part 1 than in Part 2.

Another key difference between Part 1 and Part 2 is that Part 1's solution could exactly match the cash obligations at every time interval, while Part 2 needed to reinvest excess money with the bank at the short forward rate from Date 4 to Date 5.

Problem 1 Part 3:

Similar to Part 2, Part 3 has near identical linear programming inputs to Part 1, except for one extra constraint to account for the upper limit on investing in B-rated bonds. The new inequality constraint is defined in equation 6 below.

$$3 * \sum_{i=1}^{6} P_i * x_i - \sum_{i=7}^{13} P_i * x_i \leq 0 \tag{6}$$

This is similar to equation 5 except the ratio of cash invested in A-rated bonds to B-rated bonds is at least 3 to 1 instead of at least 1 to 1. This accounts for the 25% upper limit on B-rated bonds for the bond portfolio. Inputting equation 6 into the Part 1 MATLAB script results in the following optimal portfolio.

Figure 3 – Optimal Cash Matching Bond Portfolio (25% Maximum in B-rated Bonds)

```
Problem 1 Part 3
Investing in:
     0.0000 units of bond 1
     7.1267 units of bond 2
     0.0000 units of bond 3
     0.0000 units of bond 4
     0.0000 units of bond 5
     0.0000 units of bond 6
     10.4052 units of bond 7
     0.0000 units of bond 8
     6.4638 units of bond 9
     0.0000 units of bond 10
     0.4216 units of bond 11
     0.0000 units of bond 12
     3.4216 units of bond 13

Amount reinvested at each time period
     Date 1: $0.00
     Date 2: $0.00
     Date 3: $0.00
     Date 4: $742.43
     Date 5: $129.62

Total cost of the bond portfolio: $2679.63

Weight of the portfolio in B-rated bonds: 25.00%
```

The trend noticed in Part 2 still holds true for Part 3. The weight of the portfolio in B-rated bonds is at the upper limit of 25%, and the excess money reinvested has increased and diffused over into Date 5 to Date 6 as well. Additionally, the total cost of the bond portfolio for Part 3 is the most expensive of the three portfolios. This makes sense because a smaller weighting in B-rated bonds and a heavier weighting in A-rated bonds results in a portfolio with less expected return.

Table 3 below ranks the three optimal portfolios according to the cost of the portfolios.

Table 3 – Bond Portfolio Ranking According to Cost

Portfolio	B-rated Weighting	Cost of Portfolio	Rank
Part 1	64.13%	$2639.97	1
Part 2	50%	$2644.42	2
Part 3	25%	$2679.63	3

The portfolio in Part 1 costs the least, and the portfolio in Part 3 costs the most. However, this does not necessarily mean that the Part 1 portfolio is the best portfolio, and the Part 3 portfolio is the worst portfolio. In reality, B-rated bonds have a greater risk of default compared to A-rated bonds. The weakness of this linear programming analysis is that it did not account for default risk.

Problem 2 Part 1a:

The expected returns, standard deviations, and covariances between $SPY, $GOVT, and $EEMV from Jan 2014 to Jan 2022 are summarized in Figure 4 below. The values in Figure 4 were calculated using MATLAB and the figure is a screenshot of MATLAB's command window. Refer to the Appendix for the script used to calculate these values. Note that these values are based on monthly averages.

Figure 4 – Monthly Expected Returns, Deviations, Covariances: Jan 2014 – Jan 2022

```
Problem 2 Part 1a) Answers (Monthly Values):
For SPY:
     Expected Return: 0.01210149
     Standard Deviation: 0.04097272
     Variance: 0.00167876
     Covariance with GOVT: -0.00010883
     Covariance with EEMV: 0.00102356

For GOVT:
     Expected Return: 0.00190433
     Standard Deviation: 0.01143140
     Variance: 0.00013068
     Covariance with SPY: -0.00010883
     Covariance with EEMV: -0.00003431

For EEMV:
     Expected Return: 0.00435231
     Standard Deviation: 0.03626960
     Variance: 0.00131548
     Covariance with SPY: 0.00102356
     Covariance with GOVT: -0.00003431
```

Problem 2 Part 1b:

The mean-variance optimization (MVO) model was generated using MATLAB's quadprog function. This takes the covariance matrix and all the MVO constraints as arguments. The function returns both the optimal weights of each asset for each desired expected return and half the portfolio variance.

Figure 5 below is a table generated by MATLAB that summarizes the optimal weights of $SPY, $GOVT, and $EEMV for return goals ranging from the minimum asset expected return to the maximum asset expected return, split into ten equal portions. The last column in the table shows each generated portfolio's variance.

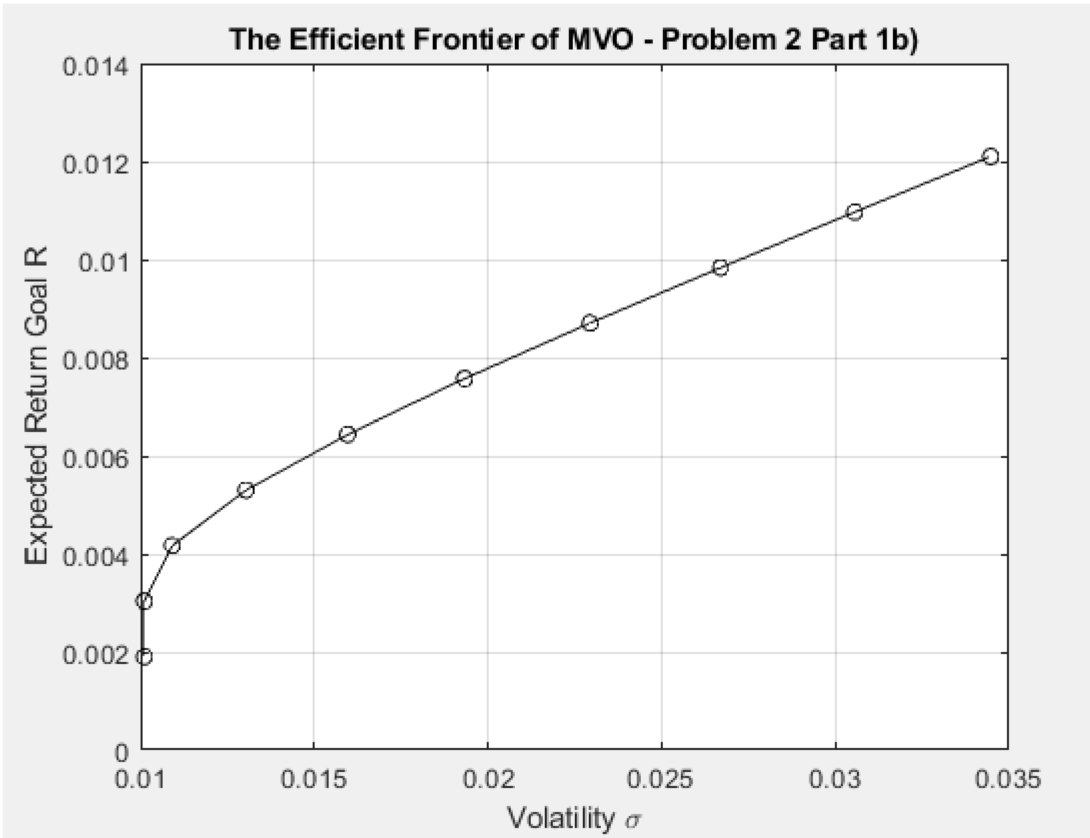
Figure 5 – Table of Optimal Portfolio Weights and Portfolio Variance (3 Assets)

```
Problem 2 Part 1b) Answers:
Table of Optimal Weights and Portfolio Variance
Return goal R   SPY Weight   GOVT Weight   EEMV Weight   Portfolio Variance
        0.002        0.107         0.876         0.017             0.000102
        0.003        0.109         0.875         0.016             0.000102
        0.004        0.240         0.833        -0.072             0.000119
        0.005        0.372         0.789        -0.161             0.000170
        0.006        0.505         0.746        -0.251             0.000255
        0.008        0.637         0.703        -0.340             0.000374
        0.009        0.770         0.660        -0.429             0.000526
        0.010        0.902         0.616        -0.519             0.000713
        0.011        1.035         0.573        -0.608             0.000934
        0.012        1.167         0.530        -0.697             0.001188
```

Figure 6 below is a plot of the efficient frontier for these three assets.

Figure 6 – Plot of the Efficient Frontier:

Assets: $SPY, $GOVT, $EEMV



Problem 2 Part 1c:

Three portfolios were considered for comparison using monthly returns from Feb 2022. Table 4 below summarizes the portfolio weights to be compared.

Table 4 – Portfolio Weights to be Compared for Feb 2022 Monthly Returns

Portfolio Name	Weight in SPY	Weight in GOVT	Weight in EEMV	Feb '22 Return
Min. Variance	10.7%	87.6%	1.7%	-1.06%
Equal Weight	33.3%	33.3%	33.3%	-1.30%
70/20/10	70%	20%	10%	-2.25%

Figure 7 is a printed MATLAB output that summarizes the ranks of the three portfolios in terms of Feb 2022 monthly returns.

Figure 7 – Ranking of the Three Portfolios in Terms of Return

```
Problem 2 Part 1c) Answers:
Rank the 3 Portfolios by Return
    Rank 1: Minimum Variance Portfolio. Feb 2022 Return = -1.06%
    Rank 2: Equal Weighted Portfolio. Feb 2022 Return = -1.30%
    Rank 3: 70% SPY, 20% GOVT, 10% EEMV Portfolio. Feb 2022 Return = -2.25%
```

These portfolio performances in Feb 2022 highlights the general relationship between risk and return. Of the three assets, the asset with the least variance and lowest expected return is GOVT, and the asset with the greatest variance and highest expected return is SPY. Therefore, portfolios with low variance have a high weighting in GOVT, and portfolios with high variance have a high weighting in SPY.

In general, higher risk portfolios tend to have higher expected returns, but Table 4 and Figure 7 show the opposite. This is because in a time period where the markets are going through a correction, lower risk portfolios tend to drop less than higher risk portfolios. In other words, portfolios with higher returns are also generally more volatile, meaning that portfolio values swing larger in both the upward and downward directions. If Feb 2022 ended up being a good month for the equities market, it would be likely that the rankings would flip, with the minimum variance optimal portfolio underperforming the other two portfolios.

Problem 2 Part 2:

By adding the 5 new assets into the MATLAB script for Problem 2 Part 1b, a new table of optimal portfolio weights and portfolio variance, as well as a new plot of the efficient frontier was generated. Similar to Part 1b, the return goals ranged from the minimum asset expected return to the maximum asset expected return, split into ten equal portions. Figures 8 and 9 below show the updated table of optimal portfolio weights and portfolio variance, and the updated plot of the efficient frontier including all eight assets.

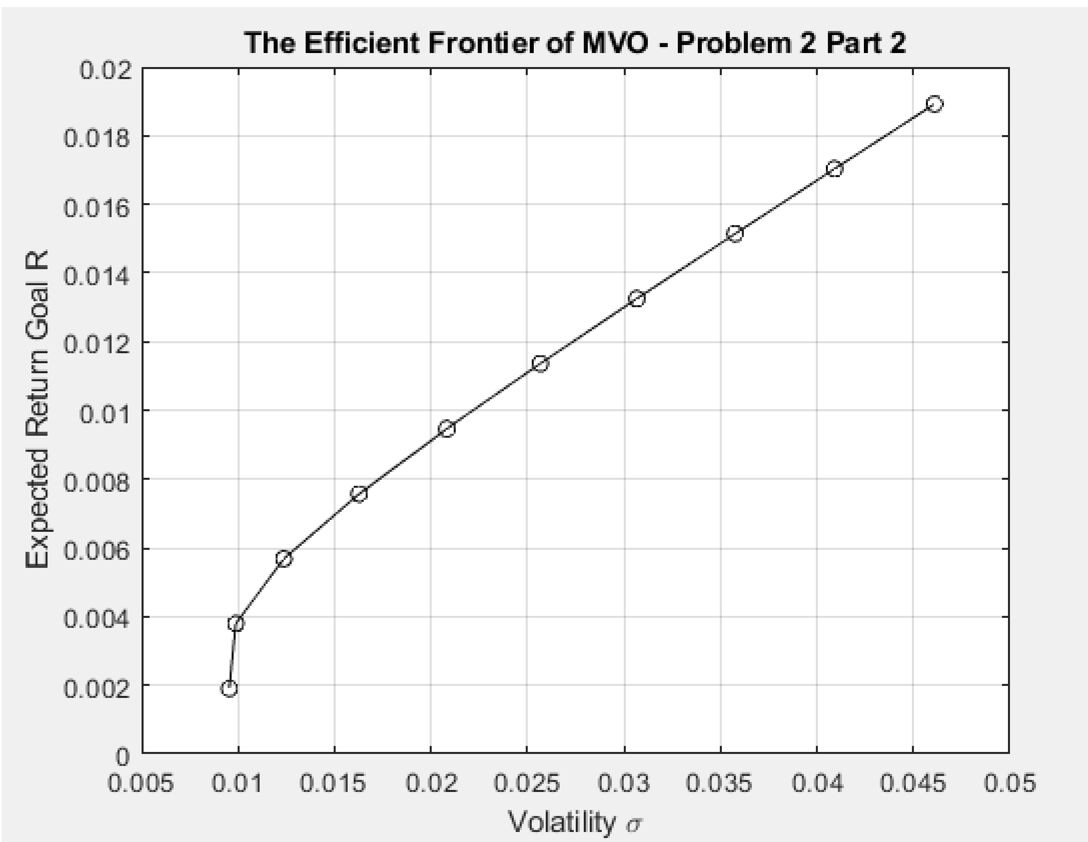
Figure 8 – Table of Optimal Portfolio Weights and Portfolio Variance (8 Assets)

Problem 2 Part 2 Answers:
Table of Optimal Weights and Portfolio Variance

Return goal R	SPY Weight	GOVT Weight	EEMV Weight	CME Weight	BR Weight	CBOE Weight	ICE Weight	ACN Weight	Portfolio Variance
0.002	0.160	0.870	0.009	0.049	-0.051	-0.016	-0.000	-0.022	0.000091
0.004	0.174	0.836	-0.019	0.073	-0.031	-0.019	-0.010	-0.004	0.000098
0.006	0.203	0.765	-0.078	0.124	0.010	-0.026	-0.031	0.033	0.000153
0.008	0.232	0.693	-0.136	0.175	0.051	-0.034	-0.052	0.070	0.000265
0.009	0.261	0.621	-0.194	0.227	0.093	-0.041	-0.073	0.107	0.000433
0.011	0.290	0.550	-0.253	0.278	0.134	-0.048	-0.094	0.144	0.000659
0.013	0.319	0.478	-0.311	0.329	0.175	-0.056	-0.115	0.181	0.000940
0.015	0.348	0.406	-0.369	0.380	0.216	-0.063	-0.136	0.217	0.001279
0.017	0.377	0.335	-0.427	0.431	0.257	-0.071	-0.157	0.254	0.001673
0.019	0.406	0.263	-0.486	0.482	0.298	-0.078	-0.178	0.291	0.002125

Figure 9 – Plot of the Efficient Frontier:

Assets: $SPY, $GOVT, $EEMV, $CME, $BR, $CBOE, $ICE, $ACN



Appendix:

Problem 1 Part 1 MATLAB Script

```
clc, clear, close all
%% Problem 1 Part 1: Formulate a linear program
%% Given information
date = [1, 2, 3, 4, 5, 6]; % Given due dates of liabilities
req_amt = [500, 200, 800, 400, 700, 900]; % Amounts owed at each date
spot = [0.01, 0.015, 0.02, 0.025, 0.03, 0.035]; % Given spot rates

% Bond prices in chronological order
bond_prices = [108, 94, 99, 92.7, 96.6, 95.9, 92.9, 110, 104, 101,...
    107, 102, 95.2]; % Unit bond prices for each bond
reinvestment_cost = [0, 0, 0, 0, 0]; % No extra cost to reinvesting
remainders at each time period

% Find short forward rates
f = zeros(1, length(date)); % Initialize the short forward rate vector

% Populate short forward rate vector
for i = 1:length(date)
    if i == 1
        f(i) = spot(i);
    else
        f(i) = (1+spot(i))^i/(1+spot(i-1))^(i-1)-1;
    end
end

% Unit bond payments at each date
coupons = [
    10, 7, 8, 6, 7, 6, 5, 10, 8, 6, 10, 7, 100;     % Each unit bond payment
at date 1
    10, 7, 8, 6, 7, 6, 5, 10, 8, 6, 110, 107, 0;    % Each unit bond payment
at date 2
    10, 7, 8, 6, 7, 6, 5, 110, 108, 106, 0, 0, 0;   % Each unit bond payment
at date 3
    10, 7, 8, 6, 7, 106, 105, 0, 0, 0, 0, 0, 0;     % Each unit bond payment
at date 4
    10, 7, 8, 106, 107, 0, 0, 0, 0, 0, 0, 0, 0;     % Each unit bond payment
at date 5
    110, 107, 108, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0     % Each unit bond payment
at date 6
    ];

reinvestment = [
    -1, 0, 0, 0, 0;       % Remainder at date 1 to be reinvested
    1+f(2), -1, 0, 0, 0; % Remainder at date 1 reinvested at f12, remainder
at date 2 to be reinvested
    0, 1+f(3), -1, 0, 0; % Remainder at date 2 reinvested at f23, remainder
at date 3 to be reinvested
    0, 0, 1+f(4), -1, 0; % Remainder at date 3 reinvested at f34, remainder
at date 4 to be reinvested
    0, 0, 0, 1+f(5), -1; % Remainder at date 4 reinvested at f45, remainder
at date 5 to be reinvested
```

```
    0, 0, 0, 0, 1+f(6)   % Remainder at date 5 reinvested at f56, any
remainder at the end is profit
    ];

%% Define the Objective Coefficients and Perform Linprog
c = [bond_prices, reinvestment_cost]'; % Minimize cost of bond portfolio

% Inequality constraints (Negative to adjust for inequality for linprog)
A = -[coupons, reinvestment]; % Coupon payments with remainders after
obligations reinvested
b = -req_amt'; % Each obligation must be at least met

% Equality constraints
Aeq = [];
beq = [];

% Variable bounds
ub = []';
lb = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]';

[x, fval] = linprog(c, A, b, Aeq, beq, lb, ub);
%% Print linprog results
fprintf('Problem 1 Part 1\n')
fprintf('Investing in:\n')
for i = 1:size(coupons, 2)
    fprintf('\t %.4f units of bond %d\n', x(i), i)
end
fprintf('\nAmount reinvested at each time period\n')
for i = 1:size(reinvestment, 2)
    fprintf('\t Date %d: $%.2f\n', i, x(i+size(coupons, 2)))
end
fprintf('\nTotal cost of the bond portfolio: $%.2f\n\n', fval)

B_value = 0;
for i = 1:6
    B_value = B_value + x(i)*bond_prices(i);
end
fprintf('Weight of the portfolio in B-rated bonds: %.2f%%\n\n',
B_value/fval*100)
```

Problem 1 Part 2 MATLAB Script

```
clc, clear, close all
%% Problem 1 Part 2: Formulate a linear program with Maximum 50% Invested in
B-rated Bonds
%% Given information
date = [1, 2, 3, 4, 5, 6]; % Given due dates of liabilities
req_amt = [500, 200, 800, 400, 700, 900]; % Amounts owed at each date
spot = [0.01, 0.015, 0.02, 0.025, 0.03, 0.035]; % Given spot rates

% Bond prices in chronological order
bond_prices = [108, 94, 99, 92.7, 96.6, 95.9, 92.9, 110, 104, 101,...
    107, 102, 95.2]; % Unit bond prices for each bond
reinvestment_cost = [0, 0, 0, 0, 0]; % No extra cost to reinvesting
remainders at each time period

% Find short forward rates
f = zeros(1, length(date)); % Initialize the short forward rate vector

% Populate short forward rate vector
for i = 1:length(date)
    if i == 1
        f(i) = spot(i);
    else
        f(i) = (1+spot(i))^i/(1+spot(i-1))^(i-1)-1;
    end
end

% Unit bond payments at each date
coupons = [
    10, 7, 8, 6, 7, 6, 5, 10, 8, 6, 10, 7, 100;     % Each unit bond payment
at date 1
    10, 7, 8, 6, 7, 6, 5, 10, 8, 6, 110, 107, 0;    % Each unit bond payment
at date 2
    10, 7, 8, 6, 7, 6, 5, 110, 108, 106, 0, 0, 0;  % Each unit bond payment
at date 3
    10, 7, 8, 6, 7, 106, 105, 0, 0, 0, 0, 0, 0;     % Each unit bond payment
at date 4
    10, 7, 8, 106, 107, 0, 0, 0, 0, 0, 0, 0, 0;     % Each unit bond payment
at date 5
    110, 107, 108, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0     % Each unit bond payment
at date 6
    ];

reinvestment = [
    -1, 0, 0, 0, 0;       % Remainder at date 1 to be reinvested
    1+f(2), -1, 0, 0, 0; % Remainder at date 1 reinvested at f12, remainder
at date 2 to be reinvested
    0, 1+f(3), -1, 0, 0; % Remainder at date 2 reinvested at f23, remainder
at date 3 to be reinvested
    0, 0, 1+f(4), -1, 0; % Remainder at date 3 reinvested at f34, remainder
at date 4 to be reinvested
    0, 0, 0, 1+f(5), -1; % Remainder at date 4 reinvested at f45, remainder
at date 5 to be reinvested
    0, 0, 0, 0, 1+f(6)   % Remainder at date 5 reinvested at f56, any
remainder at the end is profit
```

```
    ];

B_rated_limits = [
    108, 94, 99, 92.7, 96.6, 95.9,...              % Positive on all B-rated
bonds
    -92.9, -110, -104, -101, -107, -102, -95.2,... % Negative on all A-rated
bonds
    0, 0, 0, 0, 0                                  % No cost for reinvesting
    ];
B_rated_inequality = 0;                            % B rated bond value - A
rated bond value <= 0

%% Define the Objective Coefficients and Perform Linprog
c = [bond_prices, reinvestment_cost]'; % Minimize cost of bond portfolio

% Inequality constraints (Negative to adjust for inequality for linprog)
A = [
    -coupons, -reinvestment; % Coupon payments with remainders after
obligations reinvested
    B_rated_limits           % Maximum 50 percent B-rated bonds
    ];
b = [-req_amt, B_rated_inequality]'; % Each obligation must be at least met

% Equality constraints
Aeq = [];
beq = [];

% Variable bounds
ub = []';
lb = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]';

%% Print linprog results
[x, fval] = linprog(c, A, b, Aeq, beq, lb, ub);
fprintf('Problem 1 Part 2\n')
fprintf('Investing in:\n')
for i = 1:size(coupons, 2)
    fprintf('\t %.4f units of bond %d\n', x(i), i)
end
fprintf('\nAmount reinvested at each time period\n')
for i = 1:size(reinvestment, 2)
    fprintf('\t Date %d: $%.2f\n', i, x(i+size(coupons, 2)))
end
fprintf('\nTotal cost of the bond portfolio: $%.2f\n\n', fval)

B_value = 0;
for i = 1:6
    B_value = B_value + x(i)*bond_prices(i);
end
fprintf('Weight of the portfolio in B-rated bonds: %.2f%%\n\n',
B_value/fval*100)
```

Problem 1 Part 3 MATLAB Script

```
clc, clear, close all
%% Problem 1 Part 3: Formulate a linear program with Maximum 25% Invested in
B-rated Bonds
%% Given information
date = [1, 2, 3, 4, 5, 6]; % Given due dates of liabilities
req_amt = [500, 200, 800, 400, 700, 900]; % Amounts owed at each date
spot = [0.01, 0.015, 0.02, 0.025, 0.03, 0.035]; % Given spot rates

% Bond prices in chronological order
bond_prices = [108, 94, 99, 92.7, 96.6, 95.9, 92.9, 110, 104, 101,...
    107, 102, 95.2]; % Unit bond prices for each bond
reinvestment_cost = [0, 0, 0, 0, 0]; % No extra cost to reinvesting
remainders at each time period

% Find short forward rates
f = zeros(1, length(date)); % Initialize the short forward rate vector

% Populate short forward rate vector
for i = 1:length(date)
    if i == 1
        f(i) = spot(i);
    else
        f(i) = (1+spot(i))^i/(1+spot(i-1))^(i-1)-1;
    end
end

% Unit bond payments at each date
coupons = [
    10, 7, 8, 6, 7, 6, 5, 10, 8, 6, 10, 7, 100;     % Each unit bond payment
at date 1
    10, 7, 8, 6, 7, 6, 5, 10, 8, 6, 110, 107, 0;    % Each unit bond payment
at date 2
    10, 7, 8, 6, 7, 6, 5, 110, 108, 106, 0, 0, 0;  % Each unit bond payment
at date 3
    10, 7, 8, 6, 7, 106, 105, 0, 0, 0, 0, 0, 0;     % Each unit bond payment
at date 4
    10, 7, 8, 106, 107, 0, 0, 0, 0, 0, 0, 0, 0;     % Each unit bond payment
at date 5
    110, 107, 108, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0     % Each unit bond payment
at date 6
    ];

reinvestment = [
    -1, 0, 0, 0, 0;       % Remainder at date 1 to be reinvested
    1+f(2), -1, 0, 0, 0; % Remainder at date 1 reinvested at f12, remainder
at date 2 to be reinvested
    0, 1+f(3), -1, 0, 0; % Remainder at date 2 reinvested at f23, remainder
at date 3 to be reinvested
    0, 0, 1+f(4), -1, 0; % Remainder at date 3 reinvested at f34, remainder
at date 4 to be reinvested
    0, 0, 0, 1+f(5), -1; % Remainder at date 4 reinvested at f45, remainder
at date 5 to be reinvested
    0, 0, 0, 0, 1+f(6)   % Remainder at date 5 reinvested at f56, any
remainder at the end is profit
```

```
    ];

B_rated_limits = [
    3*108, 3*94, 3*99, 3*92.7, 3*96.6, 3*95.9,...  % Positive on all B-rated
bonds, for max 25% B-rated bonds, 3*B-rated bonds <= 1*A-rated bonds
    -92.9, -110, -104, -101, -107, -102, -95.2,... % Negative on all A-rated
bonds
    0, 0, 0, 0, 0                                   % No cost for reinvesting
    ];
B_rated_inequality = 0;                             % 3*B-rated bond value -
A-rated bond value <= 0

%% Define the Objective Coefficients and Perform Linprog
c = [bond_prices, reinvestment_cost]'; % Minimize cost of bond portfolio

% Inequality constraints (Negative to adjust for inequality for linprog)
A = [
    -coupons, -reinvestment; % Coupon payments with remainders after
obligations reinvested
    B_rated_limits           % Maximum 50 percent B-rated bonds
    ];
b = [-req_amt, B_rated_inequality]'; % Each obligation must be at least met

% Equality constraints
Aeq = [];
beq = [];

% Variable bounds
ub = []';
lb = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]';

%% Print linprog results
[x, fval] = linprog(c, A, b, Aeq, beq, lb, ub);
fprintf('Problem 1 Part 3\n')
fprintf('Investing in:\n')
for i = 1:size(coupons, 2)
    fprintf('\t %.4f units of bond %d\n', x(i), i)
end
fprintf('\nAmount reinvested at each time period\n')
for i = 1:size(reinvestment, 2)
    fprintf('\t Date %d: $%.2f\n', i, x(i+size(coupons, 2)))
end
fprintf('\nTotal cost of the bond portfolio: $%.2f\n\n', fval)

B_value = 0;
for i = 1:6
    B_value = B_value + x(i)*bond_prices(i);
end
fprintf('Weight of the portfolio in B-rated bonds: %.2f%%\n\n',
B_value/fval*100)
```

Problem 2 Part 1 MATLAB Script

```
clc, clear, close all
options = optimset('Display', 'off'); x0 = [];
%% Problem 2 Part 1a): Find Expected Return, Standard Deviation, and
Covariances of $SPY, $GOVT, and $EEMV
% Given Information
total_months = (2022-2014)*12; % Number of months elapsed in the data

SPY = readtable('SPY.csv', 'ReadVariableNames', false);   % Import the SPY
monthly dataset from Jan 2014 to Jan 2022
GOVT = readtable('GOVT.csv', 'ReadVariableNames', false); % Import the GOVT
monthly dataset from Jan 2014 to Jan 2022
EEMV = readtable('EEMV.csv', 'ReadVariableNames', false); % Import the EEMV
monthly dataset from Jan 2014 to Jan 2022
% Calculate the Expected Return of these Three ETFs
% Find the column of interest and convert table values to numerical format
SPY_adj_close = table2array(SPY(:, 6));
GOVT_adj_close = table2array(GOVT(:, 6));
EEMV_adj_close = table2array(EEMV(:, 6));

% Calculate the expected yearly returns of each ETF
r_SPY_m = zeros(total_months, 1);
r_GOVT_m = zeros(total_months, 1);
r_EEMV_m = zeros(total_months, 1);
for i = 1:total_months
    r_SPY_m(i) = SPY_adj_close(i+1)/SPY_adj_close(i)-1;
    r_GOVT_m(i) = GOVT_adj_close(i+1)/GOVT_adj_close(i)-1;
    r_EEMV_m(i) = EEMV_adj_close(i+1)/EEMV_adj_close(i)-1;
end

r_SPY = mean(r_SPY_m);   % Arithmetic average monthly return of $SPY from
2014-2022
r_GOVT = mean(r_GOVT_m); % Arithmetic average monthly return of $GOVT from
2014-2022
r_EEMV = mean(r_EEMV_m); % Arithmetic average monthly return of $EEMV from
2014-2022

% Calculate the standard deviation of each ETF
sigma_SPY = std(r_SPY_m);
sigma_GOVT = std(r_GOVT_m);
sigma_EEMV = std(r_EEMV_m);

% Calculate the covariance of each ETF
cov_SPY_GOVT_matrix = cov(r_SPY_m, r_GOVT_m);   % 2x2 Matrix [var(spy),
cov(spy, govt); cov(govt, spy), var(govt)]
cov_SPY_EEMV_matrix = cov(r_SPY_m, r_EEMV_m);   % 2x2 Matrix [var(spy),
cov(spy, eemv); cov(eemv, spy), var(eemv)]
cov_GOVT_EEMV_matrix = cov(r_GOVT_m, r_EEMV_m); % 2x2 Matrix [var(govt),
cov(govt, eemv); cov(eemv, govt), var(eemv)]

% Extract each unique individual covariance
var_SPY = cov_SPY_GOVT_matrix(1);
var_GOVT = cov_GOVT_EEMV_matrix(1);
var_EEMV = cov_GOVT_EEMV_matrix(4);
```

```
cov_SPY_GOVT = cov_SPY_GOVT_matrix(2);
cov_SPY_EEMV = cov_SPY_EEMV_matrix(2);
cov_GOVT_EEMV = cov_GOVT_EEMV_matrix(2);

% Print Answers for Part a)
fprintf('Problem 2 Part 1a) Answers (Monthly Values):\n')
fprintf('For SPY:\n\t Expected Return: %.8f\n', r_SPY)
fprintf('\t Standard Deviation: %.8f\n', sigma_SPY)
fprintf('\t Variance: %.8f\n', var_SPY)
fprintf('\t Covariance with GOVT: %.8f\n', cov_SPY_GOVT)
fprintf('\t Covariance with EEMV: %.8f\n\n', cov_SPY_EEMV)

fprintf('For GOVT:\n\t Expected Return: %.8f\n', r_GOVT)
fprintf('\t Standard Deviation: %.8f\n', sigma_GOVT)
fprintf('\t Variance: %.8f\n', var_GOVT)
fprintf('\t Covariance with SPY: %.8f\n', cov_SPY_GOVT)
fprintf('\t Covariance with EEMV: %.8f\n\n', cov_GOVT_EEMV)

fprintf('For EEMV:\n\t Expected Return: %.8f\n', r_EEMV)
fprintf('\t Standard Deviation: %.8f\n', sigma_EEMV)
fprintf('\t Variance: %.8f\n', var_EEMV)
fprintf('\t Covariance with SPY: %.8f\n', cov_SPY_EEMV)
fprintf('\t Covariance with GOVT: %.8f\n\n', cov_GOVT_EEMV)

%% Problem 2 Part 1b): Generate an Efficient Frontier of the Three Assets
% Define the goal return of the portfolio
mu = [r_SPY, r_GOVT, r_EEMV]; % Vector of expected returns of all three ETFs
n_points = 10; % Number of equally spaced out points on the efficient
frontier
goal_R = min(mu):(max(mu)-min(mu))/(n_points-1):max(mu); % Expected return
goals range from minimum asset return to maximum asset return

% Perform quadprog
Q = [sigma_SPY^2, cov_SPY_GOVT, cov_SPY_EEMV;
    cov_SPY_GOVT, sigma_GOVT^2, cov_GOVT_EEMV;
    cov_SPY_EEMV, cov_GOVT_EEMV, sigma_EEMV^2];
c = [0, 0, 0]';
A = -mu;
Aeq = [1, 1, 1]; beq = 1;
ub = [];
lb = []; % with short selling assumed

std_portfolio = zeros(n_points, 1); % Initialize a vector of standard
deviations for each goal R
optimal_weights = zeros(n_points, length(mu)); % Initialize a matrix of
weights in each ETF for each goal R
for i = 1:n_points
    b = -goal_R(i); % The Markowitz return constraint changes for each goal R
    [x, fval] = quadprog(Q, c, A, b, Aeq, beq, lb, ub, x0, options); % Find
weights and 1/2 portfolio variance (objective function)
    std_portfolio(i) = (fval*2)^0.5; % Store the portfolio standard deviation
for plotting
    optimal_weights(i, :) = x; % Store the optimal weights for table creation
end
```

```
% Plot the efficient frontier
plot(std_portfolio, goal_R, '-ko')
xlabel('Volatility \sigma')
ylabel('Expected Return Goal R')
title('The Efficient Frontier of MVO - Problem 2 Part 1b)')
grid on

% Print a Table of optimal weights and portfolio variance for each goal R
fprintf('Problem 2 Part 1b) Answers:\nTable of Optimal Weights and Portfolio
Variance\n')
fprintf('Return goal R   SPY Weight   GOVT Weight   EEMV Weight   Portfolio
Variance\n')
for i = 1:n_points
    fprintf('%13.3f%13.3f%14.3f%14.3f%21.6f\n', goal_R(i), ...
        optimal_weights(i,1), optimal_weights(i,2), optimal_weights(i,3),
std_portfolio(i)^2)
end

%% Problem 2 Part 1c): Compare Portfolios and Explain Relative Risk and
Return
% Define the portfolio weights
min_var_port = optimal_weights(1, :); % Minimum variance optimal portfolio
equal_port = [1/3, 1/3, 1/3]; % Equal weighted portfolio
given_port = [0.7, 0.2, 0.1]; % Portfolio defined in the question (70% SPY,
20% GOVT, 10% EEMV)

% Calculate the Feb 2022 return for each of the Three ETFs
EoFeb_SPY = 435.28; EoJan_SPY = 448.52; % Adjusted end of month closing
prices for SPY (Yahoo Finance USD)
EoFeb_GOVT = 25.69; EoJan_GOVT = 25.91; % Adjusted end of month closing
prices for GOVT (Yahoo Finance USD)
EoFeb_EEMV = 62.38; EoJan_EEMV = 62.45; % Adjusted end of month closing
prices for EEMV (Yahoo Finance USD)

r_Feb_SPY = EoFeb_SPY/EoJan_SPY-1;     % Feb 2022 return for SPY
r_Feb_GOVT = EoFeb_GOVT/EoJan_GOVT-1; % Feb 2022 return for GOVT
r_Feb_EEMV = EoFeb_EEMV/EoJan_EEMV-1; % Feb 2022 return for EEMV
r_Feb_ETF = [r_Feb_SPY; r_Feb_GOVT; r_Feb_EEMV]; % Feb 2022 ETF returns in
vector notation

% Calculate the Feb 2022 return for each portfolio
r_Feb_minvar = min_var_port*r_Feb_ETF;
r_Feb_equal = equal_port*r_Feb_ETF;
r_Feb_given = given_port*r_Feb_ETF;

% Print answers for Part c)
fprintf('\nProblem 2 Part 1c) Answers:\nRank the 3 Portfolios by Return\n')
fprintf('\tRank 1: Minimum Variance Portfolio. Feb 2022 Return = %.2f%%\n',
r_Feb_minvar*100)
fprintf('\tRank 2: Equal Weighted Portfolio. Feb 2022 Return = %.2f%%\n',
r_Feb_equal*100)
fprintf('\tRank 3: 70%% SPY, 20%% GOVT, 10%% EEMV Portfolio. Feb 2022 Return
= %.2f%%\n', r_Feb_given*100)
```

```
fprintf('\nExplained relative performance in terms of risk and return in
written report\n\n')
```

Problem 2 Part 2 MATLAB Script

```
clc, clear, close all
options = optimset('Display', 'off'); x0 = [];
%% Problem 2 Part 2: Repeat Problem 2 Part 1b with More Assets
% Given Information
total_months = (2022-2014)*12; % Number of months elapsed in the data

SPY = readtable('SPY.csv', 'ReadVariableNames', false);   % Import the SPY monthly dataset from Jan 2014 to Jan 2022
GOVT = readtable('GOVT.csv', 'ReadVariableNames', false); % Import the GOVT monthly dataset from Jan 2014 to Jan 2022
EEMV = readtable('EEMV.csv', 'ReadVariableNames', false); % Import the EEMV monthly dataset from Jan 2014 to Jan 2022
CME = readtable('CME.csv', 'ReadVariableNames', false); % Import the CME monthly dataset from Jan 2014 to Jan 2022
BR = readtable('BR.csv', 'ReadVariableNames', false); % Import the BR monthly dataset from Jan 2014 to Jan 2022
CBOE = readtable('CBOE.csv', 'ReadVariableNames', false); % Import the CBOE monthly dataset from Jan 2014 to Jan 2022
ICE = readtable('ICE.csv', 'ReadVariableNames', false); % Import the ICE monthly dataset from Jan 2014 to Jan 2022
ACN = readtable('ACN.csv', 'ReadVariableNames', false); % Import the ACN monthly dataset from Jan 2014 to Jan 2022

% Calculate the Expected Return of each asset
% Find the column of interest and convert table values to numerical format
SPY_adj_close = table2array(SPY(:, 6));
GOVT_adj_close = table2array(GOVT(:, 6));
EEMV_adj_close = table2array(EEMV(:, 6));
CME_adj_close = table2array(CME(:, 6));
BR_adj_close = table2array(BR(:, 6));
CBOE_adj_close = table2array(CBOE(:, 6));
ICE_adj_close = table2array(ICE(:, 6));
ACN_adj_close = table2array(ACN(:, 6));

% Calculate the expected monthly returns of each asset
r_SPY_m = zeros(total_months, 1);
r_GOVT_m = zeros(total_months, 1);
r_EEMV_m = zeros(total_months, 1);
r_CME_m = zeros(total_months, 1);
r_BR_m = zeros(total_months, 1);
r_CBOE_m = zeros(total_months, 1);
r_ICE_m = zeros(total_months, 1);
r_ACN_m = zeros(total_months, 1);
for i = 1:total_months
    r_SPY_m(i) = SPY_adj_close(i+1)/SPY_adj_close(i)-1;
    r_GOVT_m(i) = GOVT_adj_close(i+1)/GOVT_adj_close(i)-1;
    r_EEMV_m(i) = EEMV_adj_close(i+1)/EEMV_adj_close(i)-1;
    r_CME_m(i) = CME_adj_close(i+1)/CME_adj_close(i)-1;
    r_BR_m(i) = BR_adj_close(i+1)/BR_adj_close(i)-1;
    r_CBOE_m(i) = CBOE_adj_close(i+1)/CBOE_adj_close(i)-1;
    r_ICE_m(i) = ICE_adj_close(i+1)/ICE_adj_close(i)-1;
    r_ACN_m(i) = ACN_adj_close(i+1)/ACN_adj_close(i)-1;
end
```

```
% Combined monthly returns of each asset
r_m_total = [r_SPY_m, r_GOVT_m, r_EEMV_m, r_CME_m, r_BR_m, r_CBOE_m, r_ICE_m, r_ACN_m];

r_SPY = mean(r_SPY_m);   % Arithmetic average monthly return of $SPY from 2014-2022
r_GOVT = mean(r_GOVT_m); % Arithmetic average monthly return of $GOVT from 2014-2022
r_EEMV = mean(r_EEMV_m); % Arithmetic average monthly return of $EEMV from 2014-2022
r_CME = mean(r_CME_m);   % Arithmetic average monthly return of $CME from 2014-2022
r_BR = mean(r_BR_m);     % Arithmetic average monthly return of $BR from 2014-2022
r_CBOE = mean(r_CBOE_m); % Arithmetic average monthly return of $CBOE from 2014-2022
r_ICE = mean(r_ICE_m);   % Arithmetic average monthly return of $ICE from 2014-2022
r_ACN = mean(r_ACN_m);   % Arithmetic average monthly return of $ACN from 2014-2022
mu = [r_SPY, r_GOVT, r_EEMV, r_CME, r_BR, r_CBOE, r_ICE, r_ACN]; % Vector of expected returns of all assets

% Calculate the covariance of each asset
cov_matrix = zeros(length(mu), length(mu)); % Initialize the total covariance matrix in order of: SPY, GOVT, EEMV, CME, BR, CBOE, ICE, ACN
for i = 1:length(mu)
    for j = 1:length(mu)
        cov_ij = cov(r_m_total(:, i), r_m_total(:, j));
        cov_matrix([i, j], [i, j]) = cov_ij;
    end
end

% Define the goal return of the portfolio
n_points = 10; % Number of equally spaced out points on the efficient frontier
goal_R = min(mu):(max(mu)-min(mu))/(n_points-1):max(mu); % Expected return goals range from minimum asset return to maximum asset return

% Perform quadprog
Q = cov_matrix;
c = zeros(length(mu), 1);
A = -mu;
Aeq = ones(1, length(mu)); beq = 1;
ub = [];
lb = []; % with short selling assumed

std_portfolio = zeros(n_points, 1); % Initialize a vector of standard deviations for each goal R
optimal_weights = zeros(n_points, length(mu)); % Initialize a matrix of weights in each asset for each goal R
for i = 1:n_points
    b = -goal_R(i); % The Markowitz return constraint changes for each goal R
    [x, fval] = quadprog(Q, c, A, b, Aeq, beq, lb, ub, x0, options); % Find weights and 1/2 portfolio variance (objective function)
```

```
    std_portfolio(i) = (fval*2)^0.5; % Store the portfolio standard deviation
for plotting
    optimal_weights(i, :) = x; % Store the optimal weights for table creation
end

% Plot the efficient frontier
plot(std_portfolio, goal_R, '-ko')
xlabel('Volatility \sigma')
ylabel('Expected Return Goal R')
title('The Efficient Frontier of MVO - Problem 2 Part 2')
grid on

% Print a Table of optimal weights and portfolio variance for each goal R
fprintf('Problem 2 Part 2 Answers:\nTable of Optimal Weights and Portfolio
Variance\n')
fprintf('Return goal R   SPY Weight   GOVT Weight   EEMV Weight   ')
fprintf('CME Weight   BR Weight   CBOE Weight   ICE Weight   ACN Weight   ')
fprintf('Portfolio Variance\n')
for i = 1:n_points

fprintf('%13.3f%13.3f%14.3f%14.3f%13.3f%12.3f%14.3f%13.3f%13.3f%21.6f\n'...
        , goal_R(i), optimal_weights(i,1), optimal_weights(i,2),...
        optimal_weights(i,3), optimal_weights(i,4), optimal_weights(i,5),...
        optimal_weights(i,6), optimal_weights(i,7), optimal_weights(i,8),...
        std_portfolio(i)^2)
end
```