CHAPTER 4

Portfolio Mean-Variance Optimization

4.1. Portfolio Selection

The fundamental goal of portfolio selection is to optimally allocate investments between different assets by considering the trade-off between risk and return. In this chapter, we discuss the implementation of a quantitative tool known as Mean-Variance Optimization (MVO) using the matrix operation in EXCEL and VBA. Consider a portfolio consisting of n assets with prices $\{ S_1 , \dots , S_n \}$ and quantities $\{ q_1 , \dots , q_n \}$. If it is to be kept for a period of time, the portfolio value at end of period will be subjected to uncertain asset price changes $\{ \Delta S_1 , \dots , \Delta S_n \}$. The potential gain or loss of the portfolio in this period can be summed up as

$$\Delta S_P = \sum_{i=1}^{n} q_i \Delta S_i \tag{4.1}$$

The objective of MVO is to determine the optimal portfolio content within a budget so as to minimize the risk exposure in this period under an expected growth in value. The idea relies on the correlation among asset price changes under a stochastic regime for which the statistics can be inferred from historical data.

It is convenient to rewrite equation (4.1) in terms of asset price returns $r_i = \Delta S_i / S_i$ over the investment horizon for which portfolio return r_P in this period can be written as a weighted sum of asset returns in the basket. This gives

$$r_P = \sum_{i=1}^{n} w_i r_i \quad , \quad \sum_{i=1}^{n} w_i = 1 \tag{4.2}$$

where the weight factor w_i represents the fraction of the total portfolio budget that will be invested on the i-th asset. In the stochastic model, uncertain asset returns in (4.2) can all be considered as random variables parameterized by historical means and variances. Written as their linear combination, portfolio return can also be considered as a random variable with mean and variance defined through (4.2) as

$$\mu_P = \sum_{i=1}^{n} w_i \mu_i \tag{4.3}$$

$$\sigma_P^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij} \tag{4.4}$$

where μ_i and $\sigma_i^2 = \sigma_{ii}$ denote respectively the mean and variance of individual asset return r_i, and σ_{ij} for $i \neq j$ denotes the covariance between two different returns r_i and r_j. Under this framework, the task of MVO is to determine the optimal portfolio content that minimizes the random variation of the portfolio return in (4.4) given an expected return in (4.3) and should also be feasible under the portfolio budget. The variance terms $w_i^2 \sigma_i^2$ in (4.4) are strictly positive and they are adding up. The idea of MVO relies on the fact that the covariance terms $w_i w_j \sigma_{ij}$ could possibly be negative and thus diversify the total portfolio variance.

It is efficient to express equations (4.3) and (4.4) in matrix multiplication and formulate MVO as the determination of the column vector $\mathbf{w}$ that

$$\textit{minimize} \quad \sigma_P^2 = \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} \tag{4.5}$$

$$\textit{subject to} \quad \mathbf{w}^T \boldsymbol{\mu} = \mu_P \quad \text{and} \quad \mathbf{u}^T \mathbf{w} = 1$$

where

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix}$$

The entries in the mean vector $\boldsymbol{\mu}$ and the variance-covariance matrix $\boldsymbol{\Sigma}$ can presumably be estimated using historical asset returns over the same horizon as the portfolio. The optimization in (4.5) allows both long and short positions for which w_i can be positive or negative. It can be solved very easily using the method of Lagrange multipliers and the optimal solution was first worked out by Merton[1] as

$$\mathbf{w} = \frac{(C\mu_P - A)(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}) + (B - A\mu_P)(\boldsymbol{\Sigma}^{-1}\mathbf{u})}{BC - A^2} \tag{4.6}$$

$$\sigma_P^2 = \frac{C\mu_P^2 - 2A\mu_P + B}{BC - A^2} \tag{4.7}$$

where $A = \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$, $B = \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{u}$, and $C = \mathbf{u}^T\boldsymbol{\Sigma}^{-1}\mathbf{u}$. Since the variance-covariance matrix $\boldsymbol{\Sigma}$ is symmetric, it can be shown that the two quadratic forms, B and C, and the denominator $BC - A^2$ are all positive. The optimal portfolio mean-variance relation in (4.7) is then strictly convex as shown in Figure 4.1(a). It is usual to invert (4.7) and present the risk-adverse domain ($\mu_P \geq A/C$) of the mean-variance relation in term of the so-called portfolio efficient frontier as shown in Figure 4.1(b) with the following optimal relation.

$$\mu_P = \frac{A}{C} + \frac{1}{C}\sqrt{(BC - A^2)(C\sigma_P^2 - 1)} \tag{4.8}$$

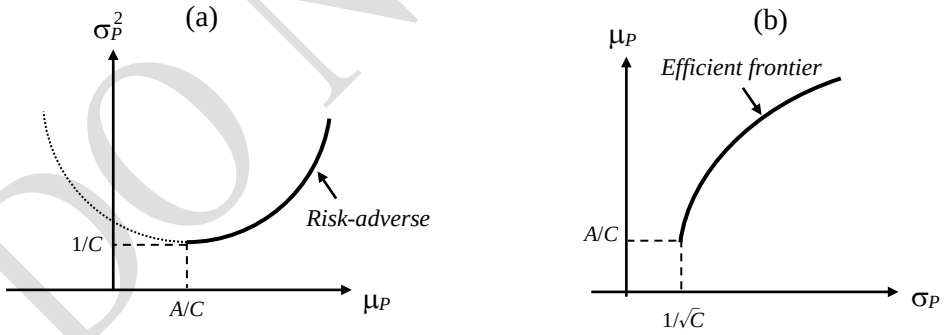


Figure 4.1: (a) Optimal portfolio mean-variance relation with risky assets only, and (b) Portfolio efficient frontier.

[1] See Robert C. Merton, "An analytic deviation of the efficient portfolio frontier", *Journal of Financial and Quantitative Analysis*, Vol. **7**, No. 3 (1972) p1851-1872. We first define the Lagrange function L that incorporates the objective function and the constraints in (4.5) with two multipliers λ_1 and λ_2 as

$$L = \tfrac{1}{2}\,\mathbf{w}^T\boldsymbol{\Sigma}\,\mathbf{w} - \lambda_1(\mathbf{w}^T\boldsymbol{\mu} - \mu_P) - \lambda_2(\mathbf{u}^T\mathbf{w} - 1)$$

The optimal $\mathbf{w}$ can be determined through the first order stationary conditions : $\{ \partial L/\partial w_1 = 0, \ldots, \partial L/\partial w_n = 0 \}$ in conjunction with the original constraints.

The above discussion analyzes the case in which all the available assets are risky. We can extend the analysis to include risk-free cash that provides a guaranteed return μ_0 over the investment period. Consider w_0 to be the fraction of the total portfolio budget that will be held as cash. Reformulate the MVO in (4.5) as the determination of the column vector $\mathbf{w}$ and w_0 that

$$\textit{minimize} \quad \sigma_P^2 = \mathbf{w}^T \boldsymbol{\Sigma}\, \mathbf{w} \tag{4.9}$$

$$\textit{subject to} \quad \mathbf{w}^T \boldsymbol{\mu} + w_0 \mu_0 = \mu_P \quad \text{and} \quad \mathbf{u}^T \mathbf{w} + w_0 = 1$$

Again, it can be solved very easily using the method of Lagrange multipliers[2] and the optimal portfolio content is given by

$$w_0 = 1 - \frac{(\mu_P - \mu_0)(A - \mu_0 C)}{C\mu_0^2 - 2A\mu_0 + B} \quad , \quad \mathbf{w} = \frac{(\mu_P - \mu_0)(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \mu_0 \boldsymbol{\Sigma}^{-1}\mathbf{u})}{C\mu_0^2 - 2A\mu_0 + B} \tag{4.10}$$

that generates the efficient frontier as shown in Figure (4.2) with the following linear optimal relation[3].

$$\mu_P = \mu_0 + \sigma_P \sqrt{C\mu_0^2 - 2A\mu_0 + B} \tag{4.11}$$

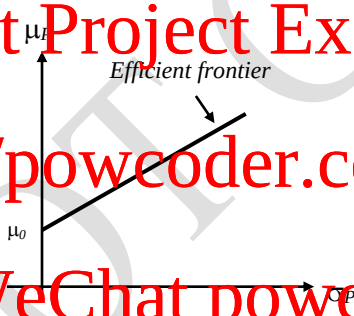


Figure 4.2: Efficient frontier of a portfolio with cash and risky assets.

We have considered so far MVO that allows both long and short positions on the risky assets for which w_i can be positive or negative. Suppose there are short-selling restrictions on the risky assets due to market constraint or internal policy. In such long-only portfolio, the asset positions w_i are all limited to be non-negative in the optimization. Accordingly, the MVO in (4.9) should be appended with additional constraints as

$$\textit{minimize} \quad \sigma_P^2 = \mathbf{w}^T \boldsymbol{\Sigma}\, \mathbf{w} \tag{4.12}$$

$$\textit{subject to} \quad \mathbf{w}^T \boldsymbol{\mu} + w_0 \mu_0 = \mu_P \;\; , \;\; \mathbf{u}^T \mathbf{w} + w_0 = 1 \;\; , \text{and} \;\; w_1 \, , \, \ldots \, , \, w_n \geq 0$$

---

[2] Similarly, we can define the Lagrange function L with two multipliers λ_1 and λ_2 as

$$L = \tfrac{1}{2}\, \mathbf{w}^T \boldsymbol{\Sigma}\, \mathbf{w} - \lambda_1 (\, \mathbf{w}^T \boldsymbol{\mu} + w_0 \mu_0 - \mu_P \,) - \lambda_2 (\, \mathbf{u}^T \mathbf{w} + w_0 - 1 \,)$$

The optimal $\mathbf{w}$ and w_0 can be determined through the stationary conditions : $\{\, \partial L/\partial w_0 = 0, \partial L/\partial w_1 = 0, \ldots , \partial L/\partial w_n = 0 \,\}$.

[3] We can rewrite the optimal risky content in (4.10) as $\mathbf{w} = (1 - w_0)\, \hat{\mathbf{w}}$ where $\hat{\mathbf{w}} = (\, \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \mu_0 \boldsymbol{\Sigma}^{-1}\mathbf{u} \,)/(A - \mu_0 C)$. Along the efficient frontier, we are actually buying or short-selling various units of the so-called market portfolio $\{\, \hat{w}_1, \ldots , \hat{w}_n \,\}$ together with cash holding. For $\mu_0 < A/C$, the cash position w_0 can be positive, zero, or negative through borrowing. For $\mu_0 \geq A/C$, the cash position is however strictly positive.

It should be noted that the cash position w_0 remains unconstrained in (4.12) where it can be positive, zero, or negative. In principle, (4.12) can be solved using again the method of Lagrange multipliers[4]. However, the evaluation of the optimal portfolio content would be non-trivial because of the inequality constraints in the Kuhn-Tucker conditions.

Markowitz has developed an efficient algorithm[5] that allows us to solve the MVO in (4.12) using simply the optimal result in (4.10). Consider another MVO related to (4.12) for which we delete the non-negative constraints $w_{1,\,2,\,\ldots\,,\,n} \geq 0$ and instead constrain certain subset of $\{ w_1, \ldots, w_n \}$, called the *OUT* subset, to be zero. The basic idea in the Markowitz algorithm is that the optimal solution of this related problem could possibly be a particular solution of the original problem for specific segment of the efficient frontier. The optimal solution can simply be obtained from (4.10) by modifying the array entries in $\{ \boldsymbol{\Sigma}, \boldsymbol{\mu}, \mathbf{u} \}$ associated with the *OUT* subset. If w_i is in the *OUT* subset, we set the i-th row of both the vectors $\boldsymbol{\mu}$ and $\mathbf{u}$ to zero, and also the i-th row and i-th column of the matrix $\boldsymbol{\Sigma}$ to zero except the diagonal entry which set to be one.

$$\mu_i \to 0 \;\; , \;\; u_i \to 0 \tag{4.13}$$

$$\Sigma_{i1}\,, \ldots, \Sigma_{in} \to 0 \;\; , \;\; \Sigma_{1i}\,, \ldots, \Sigma_{ni} \to 0 \;\; , \;\; \text{except} \;\; \Sigma_{ii} \to 1$$

Suppose $\{ \boldsymbol{\Sigma}_m , \boldsymbol{\mu}_m , \mathbf{u}_m \}$ are the modified arrays according to the *OUT* subset. The optimal solution of the related problem is then given by

$$w_0 = 1 - \frac{\mu_P^2 - 2\mu_0 A_m + \mu_0^2 C_m}{C_m \mu_0^2 - 2A_m \mu_0 + B_m} \;\; , \;\; \mathbf{w} = \frac{(\mu_P - \mu_0)(\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\mu}_m - \mu_0 \boldsymbol{\Sigma}_m^{-1}\mathbf{u}_m)}{C_m \mu_0^2 - 2A_m \mu_0 + B_m} \tag{4.14}$$

with multipliers

$$\lambda_1 = \frac{(\mu_P - \mu_0)}{C_m \mu_0^2 - 2A_m \mu_0 + B_m} \;\; , \;\; \lambda_2 = \frac{-\mu_0 (\mu_P - \mu_0)}{C_m \mu_0^2 - 2A_m \mu_0 + B_m}$$

where $A_m = \mathbf{u}_m^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\mu}_m$, $B_m = \boldsymbol{\mu}_m^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\mu}_m$, and $C_m = \mathbf{u}_m^T \boldsymbol{\Sigma}_m^{-1} \mathbf{u}_m$. It is clear in (4.14) that $w_i = 0$ inside the *OUT* subset. If all w_i outside the *OUT* subset are non-negative for particular value of μ_P, (4.14) could possibly be a solution of the MVO in (4.12) that also satisfies the Kuhn-Tucker conditions as

$$\partial L/\partial w_0 = -\lambda_1 \mu_0 - \lambda_2 = 0 \tag{4.15}$$

$$\partial L/\partial w_i = (\boldsymbol{\Sigma}\,\mathbf{w} - \lambda_1 \boldsymbol{\mu} - \lambda_2 \mathbf{u})_i = 0 \;\; \text{when} \;\; w_i \geq 0 \tag{4.16}$$
$$> 0 \;\; \text{when} \;\; w_i = 0 \;\; , \;\; \text{for } i = 1, \ldots, n$$

---

[4] Define the Lagrange function H with multipliers $\{ \lambda_1, \lambda_2, \alpha_1, \ldots, \alpha_n \}$ as

$$H = L - \alpha_1 w_1 - \ldots - \alpha_n w_n \;\; , \;\; \text{where} \;\; L = \tfrac{1}{2}\, \mathbf{w}^T \boldsymbol{\Sigma}\, \mathbf{w} - \lambda_1 (\, \mathbf{w}^T \boldsymbol{\mu} + w_0 \mu_0 - \mu_P) - \lambda_2 (\, \mathbf{u}^T \mathbf{w} + w_0 - 1)$$

With inequality constraints, the optimal $\mathbf{w}$ and w_0 can be determined through the Kuhn-Tucker conditions :

$$\partial L/\partial w_0 = 0 \,, \;\; \partial L/\partial w_i = \alpha_i \,, \;\; \alpha_i \geq 0 \,, \;\; w_i \geq 0 \,, \;\; \text{and} \;\; \alpha_i w_i = 0 \;\; \text{for } i = 1, \ldots, n$$

that can be rewritten as $\partial L/\partial w_0 = 0$, $\partial L/\partial w_i = 0$ when $w_i \geq 0$, and $\partial L/\partial w_i > 0$ when $w_i = 0$ for $i = 1, \ldots, n$.

[5] See H.M. Markowitz, G.P. Todd, and W.F. Sharpe, *Mean-Variance Analysis in Portfolio Choice and Capital Markets*, Frank J. Fabozzi Associates, 1987.

It should be noted that $\partial L/\partial w_0 = 0$ in (4.15) would automatically be satisfied with the multipliers defined in (4.14). Given portfolio return $\mu_P > \mu_0$, we can determine the optimal portfolio content by solving the MVO in (4.12) through the Markowitz algorithm as

(1) Define an *OUT* subset and construct the modified arrays { $\mathbf{\Sigma}_m$, $\boldsymbol{\mu}_m$, $\mathbf{u}_m$ } according to (4.13).

(2) Check that all the entries of **w** in (4.14) are non-negative. If so, proceed to step (3). If this is not the case, return to step (1) and try another *OUT* subset.

(3) Check that condition (4.16) has been satisfied. If so, w_0 and **w** defined in (4.14) will be an optimal solution given portfolio return μ_P. Otherwise, return to step (1) and try another *OUT* subset.

In step (1), the size of the *OUT* subset can be chosen from $N_{out} = 0$ to $N_{out} = n - 1$. When $N_{out} = n$, there is only one *OUT* subset namely the entire risky content { $w_1, \ldots, w_n$ }. The algorithm will not work as well in this case as the denominator in (4.14) vanishes with $\boldsymbol{\mu}_m$ and $\mathbf{u}_m$ being zero vectors. However, this corresponds to the trivial portfolio content with $\mathbf{w} = \mathbf{0}$ and $w_0 = 1$. The algorithm will guarantee to find a solution before we exhaust the list of all *OUT* subsets. Also, the optimal portfolio content is unique given return. We should quit the routine once we obtain a solution.

4.2. EXCEL Implementation

It is always convenient to separate the raw data from the actual calculations using different worksheets defined as[6]

dayclose – Historical daily closing prices of all risky assets to be considered.
return – Historical price returns generated through 'dayclose' for specific time horizon.
MVO – Actual calculations of the mean-variance optimization based on "return".

Figure 4.3 depicts the layout of the worksheet "dayclose" with historical daily closing prices of 35 commonly traded commodities from year 1998 to 2005. Sequences of closing prices are recorded one column per asset starting from column B onward with the corresponding timestamp given by column A. In each column, the top cell records the asset name with its ticker symbol in the following cell. For example, column B records the daily closing prices of Crude Oil with ticker symbol being CL.

	A	B	C	D	E	F	G	H	I	J	K
1		CRUDE OIL	HEATING OIL	UNL GASOLINE	NATURAL GAS	ALUMINIUM	COPPER	GOLD	ZINC	SILVER	LEAD
2	Date	CL	HO	HU	NG	LMAH	HG	GC	LMZS	SI	LMPB
3	19980101	17.64	49.08	52.81	2.264	1552	76.9	289.9	1107	5.933	564
4	19980102	17.43	49.41	53.26	2.153	1552	76.2	289.4	1107	5.9	564
5	19980105	16.89	47.92	51.86	2.207	1502	75.35	282.7	1082	5.873	559
6	19980106	16.91	47.76	52.35	2.182	1504	75.4	282.3	1084	6.015	560
7	19980107	16.82	47.33	51.94	2.145	1504	75	284.7	1089	5.973	562
8	19980108	16.97	47.75	52.62	2.046	1492	74.6	281.8	1120	5.728	580
9	19980109	16.63	46.7	52.81	2.046	1490	74.15	279.1	1087	5.593	570
10	19980112	16.47	46.21	52.03	2.002	1472	74	278.9	1077	5.44	546
11	19980113	16.43	46.09	51.39	2.014	1504	76.1	283.8	1124	5.653	547
12	19980114	16.45	46.28	51.05	2.016	1512	76.5	282.4	1142	5.658	542
13	19980115	16.34	46.23	50.39	2.094	1507	77.2	286.4	1132	5.783	527
14	19980116	16.51	46.75	50.99	2.176	1498	76.3	291.1	1152	5.758	528
15	19980119	16.51	46.75	50.99	2.176	1503	76.3	291.1	1116	5.758	513

Figure 4.3: The layout of the worksheet "dayclose" with daily closing prices.

Figure 4.4 displays the worksheet "return" that contains the historical price returns of the assets. They are generated dynamically using the raw data in "dayclose" and according to the investment horizon defined in cell K16 (named as horizon) of worksheet "MVO". Again, return sequences are

[6] Refer to mean_variance_opt.xls.

arranged one column per asset starting from column A onward with the corresponding ticker symbol defined in the top cell. Such labeling row can be constructed through a direct mapping from "dayclose" using the common expression A1 = dayclose!B2 along the row. The investment horizon should be defined in number of days. Suppose, for example, it is taken to be five days (horizon = 5). The cell A2 should correspond to the price return of CL with opening price dayclose!B3 and closing price dayclose!B8 five days later. In the same way, the following cell A3 should correspond to the price return from dayclose!B8 to dayclose!B13 in the subsequent five days. In general, this can be done very easily using the **OFFSET** function with the leftmost corner cell dayclose!$A$3 being the reference location in worksheet "dayclose". In each column of worksheet "return", price returns in each cell can be calculated by identifying the opening and closing prices through row offset from the reference location according to its row number **ROW**() as

opening price = **OFFSET**(dayclose!$A$3, (**ROW**() – 2) ∗ horizon , **COLUMN**())

closing price = **OFFSET**(dayclose!$A$3, (**ROW**() – 1) ∗ horizon , **COLUMN**())

price return = (closing price – opening price) / opening price

The column number **COLUMN**() will provide a column offset from the reference location to match up the ticker symbol in the two worksheets. Running down the row of "return", price returns should be calculated until the location of the closing price has exceeded the data range of "dayclose". It happens when closing price is pointing to a blank cell thereafter, and where we simply insert a blank cell in "return". The following expression can be applied to every column of "return" with ticker symbol:

IF(**OFFSET**(dayclose!$A$3, (**ROW**() – 1) ∗ horizon , **COLUMN**()) = "" , "" ,
(**OFFSET**(dayclose!$A$3, (**ROW**() – 1) ∗ horizon , **COLUMN**())
– **OFFSET**(dayclose!$A$3, (**ROW**() – 2) ∗ horizon , **COLUMN**()))
/ **OFFSET**(dayclose!$A$3, (**ROW**() – 2) ∗ horizon , **COLUMN**()))

	A	B	C	D	E	F	G	H	I	J
1	CL	HO	HU	NG	LMAH	HG	GC	LMZS	SI	LMPB
2	-0.073696	-0.058068	-0.045825	-0.075088	-0.028995	0.003901	-0.012073	0.022584	-0.025282	-0.065603
3	0.090575	0.065974	0.090693	0.003343	0.021234	0.036269	0.057263	0.007067	0.042711	0.001898
4	-0.104377	-0.095576	-0.099709	0.089005	-0.019168	-0.043125	-0.011559	-0.051754	0.160862	-0.007576
5	-0.038221	-0.031860	-0.038197	-0.001748	-0.036767	-0.007185	-0.013030	-0.034228	-0.122143	0.041985
6	-0.074919	-0.066976	-0.012398	-0.065674	0.013067	0.086842	-0.001016	0.012452	0.000488	0.023810
7	0.185211	0.135867	0.145106	0.095595	-0.008826	-0.046610	0.022704	0.042573	0.032531	0.035778
8	-0.075460	-0.052482	-0.065775	0.136441	-0.014726	0.026032	0.020543	0.039927	0.008192	0.010363
9	-0.023779	-0.011309	0.000199	-0.123824	0.025374	0.046411	0.021104	-0.013962	-0.017969	-0.008547
10	0.003292	0.017274	0.037781	-0.072595	-0.056271	-0.068007	-0.047377	-0.016814	-0.055211	-0.034483
11	-0.040026	-0.086966	-0.063422	-0.042612	0.010776	-0.037437	0.005007	-0.034203	-0.115022	0.005357
12	0.033493	-0.007288	0.020663	-0.022738	-0.034115	-0.008570	-0.026237	-0.020503	-0.006660	-0.035524
13	-0.221561	-0.065570	-0.081179	0.061386	-0.010670	-0.016622	-0.001364	-0.012369	0.007663	-0.009208
14	0.231946	0.066920	0.061300	0.137593	-0.034213	-0.019608	0.007514	-0.017341	0.008365	0.016729
15	0.001379	-0.022092	-0.064543	-0.125871	0.025029	0.046897	-0.003390	0.051961	0.000000	-0.007313

Figure 4.4: The layout of the worksheet "return" with price returns over specific investment horizon.

In worksheet "MVO", we first construct the variance-covariance matrix **Σ** using the asset price returns in "return". Figure 4.5 depicts the layout of the matrix with user-defined portfolio contents. As shown in the figure, the number of assets to be included in the portfolio is arbitrary with the maximum size limited to ten. The choice of asset can be specified through the cells J3:S3 by entering their ticker symbols. The total number of assets in the portfolio is given by K15 = **COUNTA**(J3:S3) that counts the number of non-empty cells in this array. The adjacent cells J4:S4 will identify the corresponding column location of data in worksheet "return" to facilitate calculation of the matrix. For an unused blank cell in J3:S3, the adjacent cell will also be blank in J4:S4. The following expression can be used to define J4 and apply to all other cells in J4:S4 :

J4 = **IF**(J3 = "" , "" , **MATCH**(J3 , return!1:1 , 0))

In Figure 4.5, the ticker symbol of J3 is defined as HG, the above function will search for this item in array return!1:1 (row 1 of "return") and report in J4 its relative position (the sixth column) in this array. We can use the column locations acquired in cells J4:S4 to construct the variance-covariance matrix defined in cells J5:S14 (named as vcmatrix). We first repeat the same sequence vertically in cells I5:I14 based on the common expression I5 = **IF**(J4 = "" , "" , J4). Thus, each matrix entry Σ_{ij} will associate with the target column locations of two assets reading off from I5:I14 (for $i = 1, 2, \ldots$) and J4:S4 (for $j = 1, 2, \ldots$). For example, the entry in J5 has target locations given by I5 and J4. The corresponding return data to be included in the covariance calculation can be identified using the **OFFSET** function with the cell return!$A$2 in leftmost column of "return" being the reference location.

COVAR(**OFFSET**(return!$A$2 , 0 , $I5 – 1 , nsample) ,
OFFSET(return!$A$2 , 0 , J$4 – 1 , nsample)) (4.17)

For both target locations, the row and column offsets from the reference location aim at the leading entry of the data set, the height parameter (named as nsample) then defines the downward range to be included in the calculation. The height parameter is simply the size of the return data given an investment horizon. It is defined in N16 = **COUNT**(return!A:A) by counting the number of numeric cells in column A of "return". If there are blank cells in J4:S4 (and so I5:I14), the variance-covariance matrix will have redundant columns (and rows) relative to the unused cells. With respect to such redundancy, the matrix should be augmented with zero columns (and zero rows) except an one at diagonal entries. We can use the following formula to define J5 and all other matrix entries:

J5 = **IF**(**OR**($I5 = "" , J$4 = "") ,
IF(**ROW**() – **ROW**($J$5) = **COLUMN**() – **COLUMN**($J$5) , 1 , 0) ,
COVAR(**OFFSET**(return!$A$2 , 0 , $I5 – 1 , nsample) ,
OFFSET(return!$A$2 , 0 , J$4 – 1 , nsample)))

	H	I	J	K	L	M	N	O	P	Q	R	S	T
1													
2			Var-Covar										
3	Tickers :		HG	LMAH	CL	GC	W						
4			6	5	1	7	15						
5	HG	6	0.00188739	0.00089565	0.00038696	0.00030496	0.00017046	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
6	LMAH	5	0.00089565	0.00086633	0.00037601	0.00019351	0.00006943	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
7	CL	1	0.00038696	0.00037601	0.00567581	0.00012155	0.00059209	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
8	GC	7	0.00030496	0.00019351	0.00012155	0.00100378	0.00019257	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
9	W	15	0.00017046	0.00006943	0.00059209	0.00019257	0.00273249	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
10			0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000	0.00000000	0.00000000	0.00000000	
11			0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000	0.00000000	0.00000000	
12			0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000	0.00000000	
13			0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000	
14			0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	
15			Nasset :	5									
16			Horizon :	10	(days)	Nsample :	197						
17			A :	3.494774989									
18			B :	0.030080053									
19			C :	2052.936879									
20													

Figure 4.5: The layout of the variance-covariance matrix in worksheet "MVO".

As shown in Figure 4.6, the mean vector **μ** is defined in C5:C14 (named as mvec) of worksheet "MVO". Similar to (4.17), it can be constructed with reference to the target locations given by I5:I14. For example, the mean return in C5 can be calculated according to the target location in I5 as

AVERAGE(**OFFSET**(return!$A$2 , 0 , $I5 – 1 , nsample))

Again, the mean vector will have zero redundant entries if there are blank cells in I5:I14. This is also true for the **u** vector defined in B5:B14 (named as uvec). The following expressions can be used to define the mean vector and the **u** vector, respectively, as

C5 = **IF**($I5 = "" , 0 , **AVERAGE**(**OFFSET**(return!$A$2 , 0 , $I5 – 1 , nsample)))

B5 = **IF**($I5 = "" , 0 , 1)

	u	Mean	(Long/Short) w	(Long/Short with cash) w & w0	(Long with cash) w & w0	(Solver) w & w0	Tickers :	
5	1	0.00475002	0.54865170	0.58967960	0.42523893	0.42523891	HG	6
6	1	0.00122930	-0.32003669	-0.83602452	0.00000000	0.00000000	LMAH	5
7	1	0.00885418	0.26791066	0.21825540	0.38147271	0.38147270	CL	1
8	1	0.00243828	0.46379599	0.05626889	0.00000000	0.00000000	GC	7
9	1	0.00134965	0.03967834	-0.10011259	0.00000000	0.00000000	W	15
10	0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000		
11	0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000		
12	0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000		
13	0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000		
14	0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000		
15				1.07193323	0.19328836	0.19328839		
16		(daily)			Markowitz			
17	Risk-free rate :	0.00019231	0.00192308					
18	Portfolio mean :	0.00057692	0.00576923			0.00576923		
19	Portfolio variance :	0.00017253	0.00023060		0.00023279	0.00023279		

Figure 4.6: The layout of the optimal output in worksheet "MVO".

Consider first the MVO problem in (4.5) that allows both long and short positions in risky assets. As shown in Figure 4.5, we have evaluated the factors *A*, *B*, and *C* in cells K17 (named as Avalue), K18 (named as Bvalue), and K19 (named as Cvalue), respectively, through EXCEL matrix operations as

K17 = **MMULT**(**TRANSPOSE**(uvec) , **MMULT**(**MINVERSE**(vcmatrix) , mvec))

K18 = **MMULT**(**TRANSPOSE**(mvec) , **MMULT**(**MINVERSE**(vcmatrix) , mvec))

K19 = **MMULT**(**TRANSPOSE**(uvec) , **MMULT**(**MINVERSE**(vcmatrix) , uvec))

In Figure 4.6, the expected portfolio return μ_P is defined in D18 (named as mport) by scaling the choice of daily rate in C18 with the investment horizon as D18 = C18∗horizon. The optimal portfolio content **w** in (4.6) can be determined in cells D5:D14 using the formula

{ D5:D14 = ((Cvalue ∗ mport – Avalue) ∗ **MMULT**(**MINVERSE**(vcmatrix) , mvec)
+ (Bvalue – Avalue ∗ mport) ∗ **MMULT**(**MINVERSE**(vcmatrix) , uvec))
/ (Bvalue ∗ Cvalue – Avalue^2) }

The minimized portfolio variance σ_P^2 can be calculated in D19 based on the optimal content as

D19 = **MMULT**(**TRANSPOSE**(D5:D14) , **MMULT**(vcmatrix , D5:D14))

Consider the MVO problem in (4.9) with the inclusion of cash. In Figure 4.6, the risk-free return μ_0 is defined in D17 = C17∗horizon (named as riskfree) by scaling the daily rate in C17. The optimal portfolio content **w** and cash w_0 in (4.10) can be determined in cells E5:E14 and E15, respectively, using the formula

{ E5:E14 = (mport – riskfree) ∗ (**MMULT**(**MINVERSE**(vcmatrix) , mvec)
– riskfree ∗ **MMULT**(**MINVERSE**(vcmatrix) , uvec))
/ (Cvalue ∗ riskfree^2 – 2 ∗ Avalue ∗ riskfree + Bvalue) }

E15 = 1 – (mport – riskfree) ∗ (Avalue – riskfree ∗ Cvalue)
/ (Cvalue ∗ riskfree^2 – 2 ∗ Avalue ∗ riskfree + Bvalue)

As before, the minimized portfolio variance can also be calculated in E19 based on the optimal content.

Consider now the MVO problem in (4.12) with short-selling restrictions on risky assets. We defer our discussion on the implementation of the Markowitz algorithm to section 4.3 with the use of VBA. Here, we consider solving this problem using EXCEL SOLVER despite there are critical issues on initializing the algorithm. In Figure 4.6, the portfolio content **w** to be optimized is defined in cells G5:G14. The corresponding cash position w_0 in G15 can be related to this content through the budget constraint $w_0 = 1 - \mathbf{u}^T \mathbf{w}$ in (4.12) as

G15 = 1 – **MMULT**(**TRANSPOSE**(uvec) , G5:G14)

In G18 and G19, we explicitly evaluate the expected return $\mathbf{w}^T \boldsymbol{\mu} + w_0 \mu_0$ and variance $\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}$ of the portfolio, respectively, relative to this content as

G18 = **MMULT**(**TRANSPOSE**(G5:G14) , mvec) + G15 ∗ riskfree

G19 = **MMULT**(**TRANSPOSE**(G5:G14) , **MMULT**(vcmatrix , G5:G14))

In the **Solver Parameters** screen as shown in Figure 4.7, we set the **Target Cells** to be the portfolio variance in G19 and check **Equal To** as **Min** for minimizing. We take in the **By Changing Cells** the portfolio content in cells G5:G14 and include in the **Subject to the Constraints** field the condition that the so-evaluated portfolio return in G18 must equal the prescribed value mport in D18. In the **Solver Options** screen as shown in Figure 4.8, we check **Assume Non-Negative** that imposes the non-negative constraints in (4.12) on the portfolio content specified in the **By Changing Cells**. It is sufficient to consider in **Precision** the required precision of 10^{-8} for the algorithm and limits the number of iterations within 20. To start off the search algorithm, we need to provide initial values for G5:G14 as close to the solution as possible. A proper choice would be a cash portfolio with no risky assets $\{ w_1 = 0, \ldots , w_n = 0 \}$. However, there is no guarantee of finding the correct optimal solution.

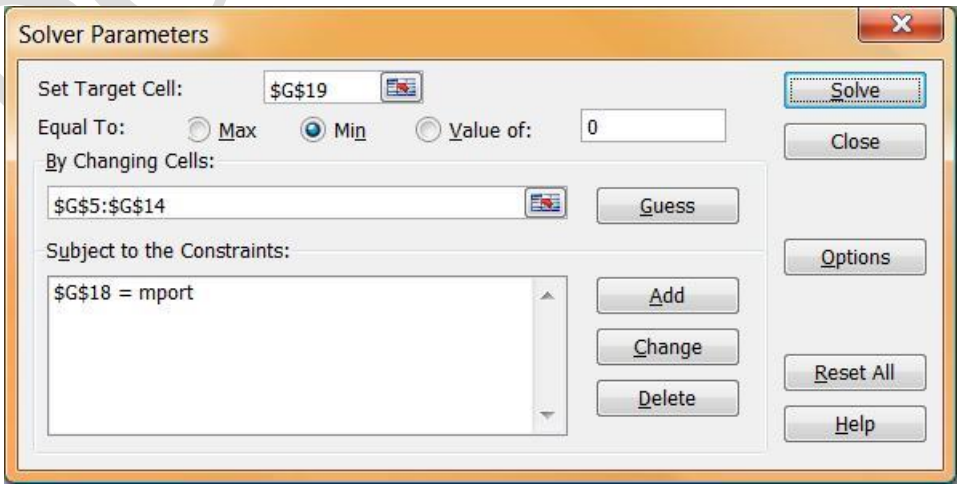


Figure 4.7: Solver Parameters screen.

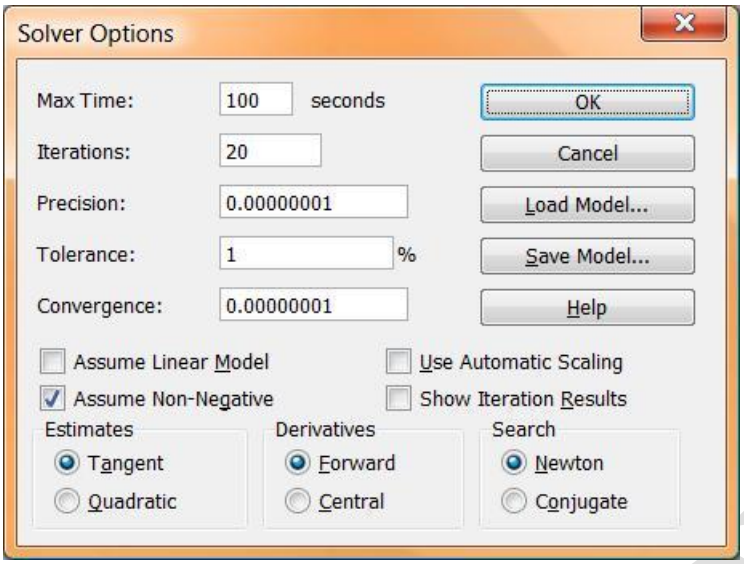


Figure 4.8: Solver Options screen.

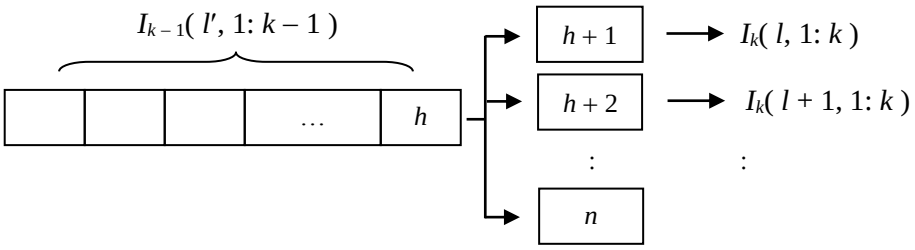
4.3. EXCEL Plus VBA Implementation

An effective and reliable way to solve the MVO problem in (4.12) is to implement the Markowitz algorithm as discussed in section 4.1 using VBA. It will guarantee to find the correct optimal solution that is unique given portfolio return. The idea is to examine all possible *OUT* subsets and identify the particular case that generates the optimal solution of the original problem. The size of the *OUT* subset can run from $N_{out} = 0$ to $N_{out} = n - 1$. For each N_{out}, there are N_c (equals n choose N_{out}) *OUT* subsets with different combinations:

N_{out}	*OUT* subsets	N_c
0	$\{ \phi \}$	1
1	$\{ w_1 \}, \{ w_2 \}, \ldots, \{ w_n \}$	n
2	$\{ w_1 , w_2 \}, \{ w_1 , w_3 \} \ldots \{ w_1 , w_n \}, \{ w_2 , w_3 \}, \ldots \{ w_2 , w_n \}, \ldots \{ w_{n-1} , w_n \}$	$\frac{1}{2}n(n-1)$
:		
$n - 1$	$\{ w_2 , w_3 , \ldots , w_n \}, \{ w_1 , w_3 , \ldots , w_n \}, \ldots, \{ w_1 , w_2 , \ldots , w_{n-1} \}$	n

Among all possible *OUT* subsets above, there is a unique combination for which the corresponding optimal content **w** given by (4.14) will be non-negative and satisfies the Kuhn-Tucker conditions in (4.16).

We first develop a routine called GetOutSubset( ) capable of generating all *OUT* subsets given its size $N_{out} = k$. Consider the array $I_k(L, 1{:}\ k)$ that provides in ascending order the pointers to all elements in different *OUT* subsets labeling through $L = 1, \ldots, N_c(k)$. When $k = 2$, for example, we have $I_2(1, 1) = 1$ and $I_2(1, 2) = 2$ that define the first *OUT* subset $\{ w_1 , w_2 \}$. Pointer arrays of size k can be generated by appending every pointer array of size $k - 1$ with an additional entry of higher value. Given pointer array $I_{k-1}(l', 1{:}\ k - 1)$ of size $k - 1$, we can generate several pointer arrays of size k (labeled consecutively as $L = l, l + 1, \ldots$); each with additional entry of separate value greater than the last entry in the given array:



In this way, the entire set of pointer arrays $I_k(L, 1{:}\,k)$ for $L = 1, \ldots, N_c(k)$ can be generated iteratively by considering every $I_{k-1}(L', 1{:}\,k-1)$ for $L' = 1, \ldots, N_c(k-1)$. The pseudo code of GetOutSubset( ) is given by Code 4.1 which performs such tasks. As input, the routine reads in all pointer arrays for size $k-1$ and the number of combinations. As output, it generates the arrays for size k and updates the number of combinations. The iteration should start off from $k = 1$ with $N_c(1) = n$ pointer arrays, namely, $I_1(1, 1) = 1$, $I_1(2, 1) = 2$, … , and $I_1(n, 1) = n$. The VBA code of GetOutSubset( ) is given by Code 4.2. Note that "nmax" and "Ncmax" are parameters that configure the maximum possible size of n and N_c, respectively, in the module. In worksheet "MVO", the number of assets to be included in the portfolio is limited below ten. We should set nmax = 10, and the maximum number of *OUT* subsets should correspondingly be Ncmax = 252 (10 choose 5).

GetOutSubset(n , k , N_c , $I(1 : N_c \ , 1 : k)$)

$l = 0$

# input N_c and I for size $k-1$, and consider every input array
For($L' = 1$ to N_c) {

# for particular array I, generate several I_{new} for size k with consecutive labeling
For($j = I(L', k-1) + 1$ to n) { $l = l + 1$
For($i = 1$ to $k-1$){ $I_{new}(l, i) = I(L', i)$ }
$I_{new}(l, k) = j$ }
}

# return I_{new} as I and update N_c
For($L = 1$ to l) { For($i = 1$ to k){ $I(L, i) = I_{new}(L, i)$ } }
$N_c = l$

Code 4.1: Pseudo code of the GetOutSubset( ) routine.

```
Sub GetOutSubset(n As Integer, k As Integer, ByRef Nc As Integer, ByRef Iout() As Integer)
   Dim lcount As Integer, L As Integer, Lprime As Integer
   Dim i As Integer, j As Integer
   Dim Inew(1 To Ncmax, 1 To nmax)  As Integer

   lcount = 0
   For Lprime = 1 To Nc
      For j = Iout(Lprime, k - 1) + 1 To n
         lcount = lcount + 1
         For i = 1 To k - 1
            Inew(lcount, i) = Iout(Lprime, i)
         Next i
         Inew(lcount, k) = j
      Next j
   Next Lprime

   For L = 1 To lcount
      For i = 1 To k
         Iout(L, i) = Inew(L, i)
      Next i
   Next L

   Nc = lcount
End Sub
```

Code 4.2: VBA code of the GetOutSubset( ) routine.

We now want to develop a routine called Markowitz( ) that considers every *OUT* subset generated from GetOutSubset( ) and performs the checking in steps 1-3 as stated at end of section 4.1. The pseudo code of Markowitz( ) is given by Code 4.3. As input, it reads in the total number of risky assets n, the data arrays { $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$, $\mathbf{u}$ }, the expected portfolio return μ_P, and the risk-free return μ_0. As output, it returns the optimal portfolio content $\mathbf{w}$ and cash position w_0. To examine all possible *OUT* subsets, we have considered in the outer loop the values of N_{out} from 0 to $n - 1$. For each N_{out}, we generate the entire *OUT* subsets and perform the checking on every combination in the inner loop of L. The checking will proceed to higher values of N_{out} until an optimal solution has been identified where the procedure will be stopped immediately.

For $N_{out} = 0$, there is $N_c(0) = 1$ *OUT* subset { ϕ } with no modification on the entries in { $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$, $\mathbf{u}$ }. For $N_{out} \geq 1$, the *OUT* subsets are generated iteratively through GetOutSubset( ) starting off from $N_{out} = 1$ with $N_c(1) = n$ subsets as defined above. For each *OUT* subset, the modified arrays { $\boldsymbol{\Sigma}_m$, $\boldsymbol{\mu}_m$, $\mathbf{u}_m$ } will be constructed according to the generated pointer array, and the portfolio content defined in (4.14) will also be calculated. The checking of non-negative and Kuhn-Tucker conditions for such portfolio content will subsequently be conducted.

It should be noted that the denominator $D_m = C_m \mu_0^2 - 2A_m \mu_0 + B_m$ in (4.14) is strictly positive. However, it could possibly be negative due to floating point precision that renders the sign of (4.14) to be misidentified. Under double precision, a real number will only be quoted up to 15 decimal places, it is therefore essential to adjust the denominator by a floating precision of $\varepsilon = 10^{-14}$. The same factor should also be used in checking whether a real number x is negative ($x < -\varepsilon$), positive ($x > \varepsilon$), or zero ($|x| \leq \varepsilon$). Using the first "Next L", we skip those *OUT* subsets with negative portfolio content in (4.14) if $w_i < -\varepsilon$ for either one component of $\mathbf{w}$. Then, for the Kuhn-Tucker conditions in (4.16), we skip again using the second "Next L" if neither ($\eta_i > \varepsilon$) $\cap$ ($|w_i| \leq \varepsilon$) or ($|\eta_i| \leq \varepsilon$) $\cap$ ($w_i \geq -\varepsilon$) is true (*i.e.* *testflag* = .False.) for either one component of $\mathbf{w}$ and $\boldsymbol{\eta} = (\Sigma \mathbf{w} - \lambda_1 \boldsymbol{\mu} - \lambda_2 \mathbf{u})$. It runs through all L and proceeds to higher values of N_{out}. If an *OUT* subset has not been filtered out by these two exit conditions, the corresponding portfolio content in (4.14) will be the optimal solution of the MVO in (4.12). The entire procedure will be stopped immediately by "Exit N_{out}" that exits two nested loops.

Markowitz(n , $\mu(1:n)$, $u(1:n)$, $\Sigma(1:n, 1:n)$, μ_P , μ_0 , $w(1:n)$, w_0)

$\varepsilon = 1 \times 10^{-14}$

For ($N_{out} = 0$ to $n - 1$) {

generate the *OUT* subsets

If($N_{out} = 0$) then

$N_c = 1$

elseif($N_{out} = 1$) then

$N_c = n$

For ($L = 1$ to N_c) { $I(L, 1) = L$ }

else

Call *GetOutSubset*(n , N_{out} , N_c , $I(1:N_c, 1:N_{out})$)

endif

For ($L = 1$ to N_c) {

construct the modified arrays according to (4.13)

For ($i = 1$ to n) { $\mu_m(i) = \mu(i)$, $u_m(i) = u(i)$

For ($j = 1$ to n) do{ $\Sigma_m(i, j) = \Sigma(i, j)$ } }

For ($k = 1$ to N_{out}) { $i = I(L, k)$
$\mu_m(i) = 0$, $u_m(i) = 0$
For ($j = 1$ to n) { $\Sigma_m(i, j) = 0$, $\Sigma_m(j, i) = 0$ }
$\Sigma_m(i, i) = 1$ }

calculate A_m, B_m, and C_m

$A_m = \boldsymbol{u}_m^T(1:n)\, \boldsymbol{\Sigma}_m^{-1}(1:n, 1:n)\, \boldsymbol{\mu}_m(1:n)$
$B_m = \boldsymbol{\mu}_m^T(1:n)\, \boldsymbol{\Sigma}_m^{-1}(1:n, 1:n)\, \boldsymbol{\mu}_m(1:n)$
$C_m = \boldsymbol{u}_m^T(1:n)\, \boldsymbol{\Sigma}_m^{-1}(1:n, 1:n)\, \boldsymbol{u}_m(1:n)$

calculate the portfolio content defined in (4.14)

$D_m = C_m \mu_0^2 - 2A_m \mu_0 + B_m + \varepsilon$
$\lambda_1 = (\mu_P - \mu_0) / D_m$, $\lambda_2 = -\mu_0 (\mu_P - \mu_0) / D_m$
$\mathbf{w}(1:n) = \lambda_1 \boldsymbol{\Sigma}_m^{-1}(1:n, 1:n)\, \boldsymbol{\mu}_m(1:n) + \lambda_2 \boldsymbol{\Sigma}_m^{-1}(1:n, 1:n) \cdot \boldsymbol{u}_m(1:n)$
$w_0 = 1 - \lambda_1 (A_m - \mu_0 C_m)$

check that all the entries of **w** in (4.14) are non-negative

For ($i = 1$ to n) { If($w(i) < -\varepsilon$) { Next L } }

check that the KKT condition (4.16) has been satisfied

$\boldsymbol{\eta}(1:n) = \boldsymbol{\Sigma}(1:n, 1:n)\, \mathbf{w}(1:n) - \lambda_1 \boldsymbol{\mu}(1:n) - \lambda_2 \mathbf{u}(1:n)$

For ($i = 1$ to n) { *testFlag* = **OR**(**AND**(**ABS**($\eta(i)$) $\le \varepsilon$, $w(i) \ge -\varepsilon$) ,
AND($\eta(i) > \varepsilon$, **ABS**($w(i)$) $\le \varepsilon$))
If (.NOT. *testFlag*) { Next L } }
Exit N_{out}
} }

Code 4.3: Pseudo code of the Markowitz( ) routine.

The VBA code of Markowitz( ) is given by Code 4.4. For the calculations of A_m, B_m and C_m , we have used the routine SolveAxb( ) (see Code 3.4) to first calculate the vectors $\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m^{-1}\boldsymbol{u}_m$. The matrix multiplications with the transposed vectors $\boldsymbol{u}_m^T$ and $\boldsymbol{\mu}_m^T$ can then be performed very easily through the rule $\boldsymbol{x}^T\boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$ for two ($n \times 1$) vectors. In addition, the portfolio content **w** in (4.14) can also be calculated immediately using the same results. Notice that the vector $\boldsymbol{\eta} = \partial L/\partial \mathbf{w}$ in (4.16) has been determined by directly applying the rule for matrix multiplication and addition as

$$\eta_i = \sum_{k=1}^{n} \Sigma_{ik} w_k - \lambda_1 \mu_i - \lambda_2 u_i \quad , \quad \text{for } i = 1, 2, \ldots, n$$

To exit a nested loop during the checking of non-negative **w** and the KKT condition, we use "GoTo nextL" and label the "Next L" statement in the coding as "nextL". To terminate the entire procedure once the optimal solution has been found, we exit two nested loops using "GoTo exitNout" for which we label the line immediately after "Next Nout" as "exitNout".

We will use the same spreadsheet design in Figures 4.5 and 4.6 for this VBA implementation. The main VBA routine MVO( ) can be invoked through the "Markowitz" button in the spreadsheet as shown in Figure 4.6. In Code 4.5, we have divided the MVO( ) routine into three parts handling the data input, the core Markowitz algorithm, and the output. As input, it reads in the total number of risky assets n in cell K15, the expected portfolio return μ_P in cell D18, and the risk-free return μ_0 in

cell D17. It also reads in the data arrays { $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$, **u** } according to the size of *n* with the use of the **OFFSET** function relative to the cells J5, C5, and B5, respectively. As output, it returns the optimal portfolio content and cash position that display in cells F5:F14 and F15, respectively. It should be noted that if *n* is less than nmax = 10, the additional portfolio contents in F5:F14 will always be zero in the output.

```
Sub Markowitz(n As Integer, mvec() As Double, uvec() As Double, vcmatrix() As Double, mport As Double, _
              riskfree As Double, ByRef wvec() As Double, ByRef w0 As Double)

    Dim Nout As Integer, Nc As Integer
    Dim Iout(1 To Ncmax, 1 To nmax) As Integer
    Dim L As Integer, i As Integer, j As Integer, k As Integer

    Dim mvecm(1 To nmax) As Double
    Dim uvecm(1 To nmax) As Double
    Dim vcmatrixm(1 To nmax, 1 To nmax) As Double
    Dim etavec(1 To nmax) As Double
    Dim tempvec1(1 To nmax) As Double
    Dim tempvec2(1 To nmax) As Double

    Dim Am As Double, Bm As Double, Cm As Double, Dm As Double
    Dim lambda1 As Double, lambda2 As Double
    Dim testFlag As Boolean

    For Nout = 0 To n - 1

        'generate the OUT subsets
        If (Nout = 0) Then
            Nc = 1
        ElseIf (Nout = 1) Then
            Nc = n
            For L = 1 To Nc: Iout(L, 1) = L: Next L
        Else
            Call GetOutSubset(n, Nout, Nc, Iout)
        End If

        For L = 1 To Nc
            'construct the modified arrays
            For i = 1 To n
                mvecm(i) = mvec(i)
                uvecm(i) = uvec(i)
                For j = 1 To n: vcmatrixm(i, j) = vcmatrix(i, j): Next j
            Next i

            For k = 1 To Nout
                i = Iout(L, k)
                mvecm(i) = 0
                uvecm(i) = 0
                For j = 1 To n
                    vcmatrixm(i, j) = 0
                    vcmatrixm(j, i) = 0
                Next j
                vcmatrixm(i, i) = 1
            Next k

            'calculate Am, Bm, and Cm
            Call SolveAxb(vcmatrixm, mvecm, tempvec1, n, 1, 1, 1)
            Call SolveAxb(vcmatrixm, uvecm, tempvec2, n, 1, 1, 1)
            Am = 0
            Bm = 0
            Cm = 0
            For i = 1 To n
                Am = Am + uvecm(i) * tempvec1(i)
                Bm = Bm + mvecm(i) * tempvec1(i)
                Cm = Cm + uvecm(i) * tempvec2(i)
            Next i

            'calculate the portfolio content
            Dm = Cm * riskfree ^ 2 - 2 * Am * riskfree + Bm + eps
            lambda1 = (mport - riskfree) / Dm
            lambda2 = -riskfree * (mport - riskfree) / Dm
            For i = 1 To n: wvec(i) = lambda1 * tempvec1(i) + lambda2 * tempvec2(i): Next i
```

```
        w0 = 1 - lambda1 * (Am - riskfree * Cm)

        'check that the portfolio content are non-negative
        For i = 1 To n
           If (wvec(i) < -eps) Then GoTo nextL
        Next i

        'checking the KKT condition
        For i = 1 To n
           tempvec1(i) = 0
           For j = 1 To n: tempvec1(i) = tempvec1(i) + vcmatrix(i, j) * wvec(j): Next j
           etavec(i) = tempvec1(i) - lambda1 * mvec(i) - lambda2 * uvec(i)
        Next i

        For i = 1 To n
           testFlag = (Abs(etavec(i)) <= eps And wvec(i) >= - eps) Or (etavec(i) > eps And Abs(wvec(i) <= eps))
           If (Not testFlag) Then GoTo nextL
        Next i

        GoTo exitNout

nextL: Next L

    Next Nout
exitNout:

End Sub
```

Code 4.4: VBA code of the Markowitz( ) routine.


```
Option Explicit
Private Const nmax = 10, Ncmax = 252
Private Const eps = 1 * 10 ^ -14
```



```
Sub MVO()

    Dim n As Integer
    Dim mvec(1 To nmax) As Double
    Dim uvec(1 To nmax) As Double
    Dim vcmatrix(1 To nmax, 1 To nmax) As Double
    Dim mport As Double
    Dim riskfree As Double
    Dim wvec(1 To nmax) As Double
    Dim w0 As Double
    Dim i As Integer, j As Integer

    n = Range("K15").Value
    mport = Range("D18").Value
    riskfree = Range("D17").Value

    For i = 1 To n
       mvec(i) = Range("C5").Offset(i - 1)
       uvec(i) = Range("B5").Offset(i - 1)
       For j = 1 To n: vcmatrix(i, j) = Range("J5").Offset(j - 1, i - 1): Next j
    Next i

    Call Markowitz(n, mvec, uvec, vcmatrix, mport, riskfree, wvec, w0)

    For i = 1 To nmax: Range("F5").Offset(i - 1) = wvec(i): Next i
    Range("F15").Value = w0

End Sub
```

Code 4.5: VBA code of the MVO( ) routine.