財務數值方法 HW6

Q1.

首先先上課的範例程式加上宣告 y[MATRIX_SIZE]這個 dictionary

```
7    {
8        int i,j;
9        double X[MATRIX_SIZE];
10       double y[MATRIX_SIZE] = { 0 };
```

接著再輸入矩陣的所有條件後(與範例程式一樣)，加上計算 y = S 乘以 X 矩陣，算法就是用矩陣的乘法先將 y 算出來之後再宣告一個 Var_Rp 去計算 y 和 X 的內積

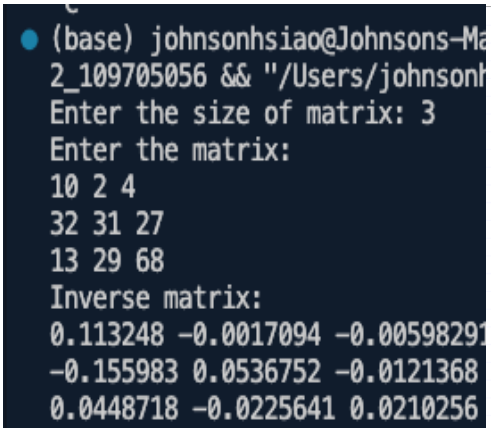
```
// 計算 y = S*X
for (i = 0; i < MATRIX_SIZE; i++)
{
    for (j = 0; j < MATRIX_SIZE; j++)
    {
        y[i] += S[i][j] * X[j];
    }
}
// 計算 Var(R_p) = X^T * y
double Var_Rp = 0;
for (i = 0; i < MATRIX_SIZE; i++)
{
    Var_Rp += X[i] * y[i];
}
printf("投資組合的變異度為 %lf \n", Var_Rp);
printf("95%信心水準下的VaR %lf", -1.645 * sqrt(Var_Rp));
return 0;
```

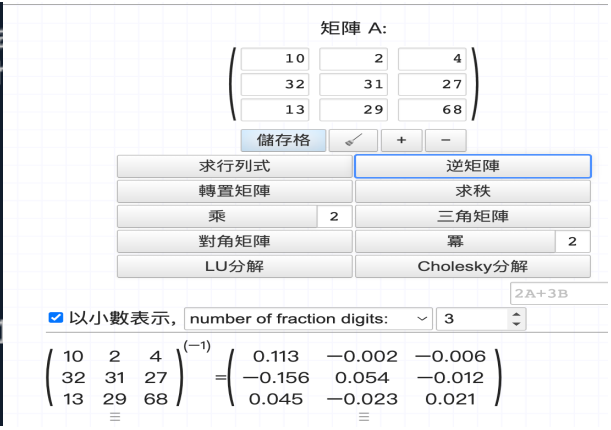
此圖為 demo 的結果

```
輸入資產配置
第1個資產:0.3
第2個資產:0.2
第3個資產:0.5
輸入共變異數矩陣
第1和第1個資產的共變異度=1
第2和第1個資產的共變異度=0.5
第2和第2個資產的共變異度=1
第3和第1個資產的共變異度=0.3
第3和第2個資產的共變異度=0.6
第3和第3個資產的共變異度=1
投資組合的變異度為 0.650000
95信心水準下的VaR -1.326241%
(base) johnsonhsiao@Johnsons-MacB
```

Q2.

以下兩圖是我用自己的 terminal 和網路上提供的線上計算機驗證，結果是一致的，提供助教參考～～～程式碼說明在下方





一開市向老師上課說的因為 C++需要效能的運算，所以要一開始就將 size 指定好，之後因為之後要輸出整個矩陣，所以先定義好 print 要怎麼 print，輸出二維陣列 arr 中的元素，大小為 n*n，印出格式為每一列數字以空格分隔，每一列後換行，就跟矩陣的定一一樣。

```
4   const int MAX_SIZE = 50;
5
6   void print(double arr[MAX_SIZE][MAX_SIZE], int n) {
7       for (int i = 0; i < n; i++) {
8           for (int j = 0; j < n; j++) {
9               cout << arr[i][j] << " ";
10          }
11          cout << endl;
12      }
13  }
```

而在主要的 gauss-jordon function 中的參數為接受兩個二維陣列 a 和 b 以及整數 n 和 m。a 為原矩陣，大小為 n*n，b 為單位矩陣，大小為 m*m。對 a 和 b 做高斯-約旦消去法，求出 a 的反矩陣，並儲存於 b 中。詳細邏輯為將矩陣 a 變為上三角矩陣，並同時將矩陣 b 也進行相應的變換。接著，利用約旦消去法將矩陣 a 變為對角矩陣，同時將矩陣 b 也進行相應的變換，得到的矩陣 b 即為矩陣 a 的反矩陣

```
15  void gauss_jordan(double a[MAX_SIZE][MAX_SIZE], int n, double b[MAX_SIZE][MAX_SIZE], int m) {
16      double temp;
17      for (int i = 0; i < n; i++) {
18          for (int j = 0; j < n; j++) {
19              if (i != j) {
20                  temp = a[j][i] / a[i][i];
21                  for (int k = 0; k < n; k++) {
22                      a[j][k] -= temp * a[i][k];
23                      b[j][k] -= temp * b[i][k];
24                  }
25              }
26          }
27      }
28      for (int i = 0; i < n; i++) {
29          temp = a[i][i];
30          for (int j = 0; j < n; j++) {
31              a[i][j] /= temp;
32              b[i][j] /= temp;
33          }
34      }
35  }
```

最後主程式的參數為接受使用者輸入矩陣的大小 n，以及 n*n 個元素，儲存在陣列 a 中並初始化陣列 b 為單位矩陣以提供去高斯約旦的 function 計算。呼叫 gauss_jordan 函式求出 a 的反矩陣，並儲存於 b 中。輸出 b，即為 a 的反矩陣，並使用 print 函式印出。

```
37  int main() {
38      int n;
39      double a[MAX_SIZE][MAX_SIZE], b[MAX_SIZE][MAX_SIZE];
40      cout << "Enter the size of matrix: ";
41      cin >> n;
42      cout << "Enter the matrix:" << endl;
43      for (int i = 0; i < n; i++) {
44          for (int j = 0; j < n; j++) {
45              cin >> a[i][j];
46              if (i == j) {
47                  b[i][j] = 1;
48              }
49              else {
50                  b[i][j] = 0;
51              }
52          }
53      }
54      gauss_jordan(a, n, b, n);
55      cout << "Inverse matrix:" << endl;
56      print(b, n);
57      return 0;
58  }
59
```