Optimization of Data Requests Timing by Working with Matrixes under MSAccess Environment

Alexandru ATOMEI
Accountancy and Management Information Systems Faculty
Academy of Economic Studies, Bucharest, Romania

Abstract: This paper is going to emphasize an optimised code in order to manage matrix calculus under MSAccess. The economic impact of using such a method is the optimal cost-benefit solution, and optimised timing for data management. As well, matrix calculus is the base of Variance-Covariance method used by financial corporations as an advanced method for estimation of market risk movements with direct impact over the capital required by prudential bodies.

Keywords: Visual Basic, DAO (Data Access Objects) Recordset, System DSN (Data Source Name) driver, Variance-Covariance Matrix, Value at Risk.

1 Optimization of data requests timing by working with matrixes under MSAccess Environment

A. Current stage of matrix calculus facilities

In order to use the matrixes for economic purposes, there are a series of software solutions more or less integrated in executive management of the financial corporations.

The proposed method for optimised management of data requests by working with matrixes is a similar method to the UNIX concept, which deals with its platform as a core base, and whose interaction with other applications is performed through shells. Similarly, in this paper it will be presented this facility developed under Windows environment.

The econometric softwares (Matlab, SAS, Eviews, SPSS) offer some flexible solutions, but the complete integrated solutions require, in initial phase, licences, implementation projects and testing procedure, which exceed the forecasted budgets of most of up-to-medium financial corporations, and on a daily basis, a professional team permanently available.

The development of integrated solutions require a business study, whose implementation could not be done fastly, in order to generate advanced calculus for executive management.

A database permits the transfer of data through a XML (Extensible Markup Language) file, but there is needed a transformation of XML file into an Standardised XML file. Nowadays, last versions of MSAccess and Oracle offers a good interaction with XML files, so that, XML files could be an alternative to the present solution presented in this paper.

The matrixes could be easily developed, also, in MSExcel programs. The MSAccess offers good database facilities, but, in terms of calculus is not so powerful compared with MSExcel. In this paper, we shall present how the MsAccess and MSExcel could be linked dinamically for serving a Variance-Covariance matrix determination.

B. The role of matrixes in accountancy

The fair value concept is often invoked in International Accountancy Standards. The fair value (market value in most of the cases) generates volatility for a certain type of asset. This volatility represents the base for a specific modeling requiring a matrix calculus.

A simple covariance between two series of data could be calculated as follows (formula 1):

\[ \text{Formula 1- Covariance of two series of data} \]
Optimization of Data Requests Timing by Working with Matrixes under MSAccess Environment

Database Access through Java Technologies

\[
cov(X, Y) = \sum_{i=1}^{N} \frac{(x_i - x_a)(y_i - y_a)}{N}
\]

Where:
- \( Cov(X, Y) \) = covariance between two series of data (‘x’ and ‘y’);
- \( x_a \) = average of ‘x’ seria;
- \( y_a \) = average of ‘y’ seria;
- \( N \) = number of observations.

A Variance-Covariance matrix is formalised as follows (formula 2):

**Formula 2 – Variance-Covariance matrix**

\[
\begin{bmatrix}
\overline{x_1}^2 / N & \overline{x_1} \overline{x_2} / N & \cdots & \overline{x_1} \overline{x_c} / N \\
\overline{x_2} \overline{x_1} / N & \overline{x_2}^2 / N & \cdots & \overline{x_2} \overline{x_c} / N \\
\vdots & \vdots & \ddots & \vdots \\
\overline{x_c} \overline{x_1} / N & \overline{x_c} \overline{x_2} / N & \cdots & \overline{x_c}^2 / N
\end{bmatrix}
\]

Where:
- \( \text{COV} \) = Variance-Covariance matrix
- \( x_i \) = deviation from the \( i \)th data set
- \( \overline{x_i}^2 / N \) = variance of elements from the \( i \)th data set
- \( \overline{x_i} \overline{x_j} / N \) = covariance of elements from the \( i \)th and \( j \)th data sets
- \( N \) = number of observations for each of the \( c \) data sets

The capital adequacy process of financial corporations is directly influenced by the type of models used for potential losses determination. The usage of matrixes will introduce a better estimation of potential losses rather than roughly applying a pre-determined coefficient which is often too high, in order to be conservative.

### C. Case study

Is is supposed a MSAccess application storing data about financial assets, eg. equities / indexes listed on Stock Exchange. The database have daily recordings for one year period for 8 types of stock-exchange indexes.

The Variance-Covariance matrix requires two series of data. In our case, the series of data are represented by one seria of daily yields against the average of daily yields and second seria consisting from transposed data of previous seria.

The reason of transposed seria introduction is given by the necessity for introduction of an random factor in daily yields seria, having the same average, standard deviation and normal distribution. The random factor is founded on the supposition that a certain evolution could be contrary to the trend.

The first step in our calculation consist of using the DAO recordset facilities in order to transform the daily prices in daily yields. The second step deals with average yields calculation. The third step is performing the deduction of average yield from each daily yield.

There is presented hereinafter, the Visual Basic Code from MSAccess in order to generate the calculation of daily yields against average:

```vba
Function eq_delta() 'Calculation of daily yields against average'
    Dim r As DAO.Recordset, t As DAO.Recordset, q As DAO.Recordset
    Dim strSQL As String
    'Local variables
    Dim data As Date
```
Dim deltaBETI, deltaBETC, deltaBETFI, deltaVAB, deltaROTXUSD, deltaROTXEUR, deltaRSQALL As Double
Dim BETI, BETC, BETFI, VAB, ROTXUSD, ROTXEUR, RSQALL As Double
Set r = CurrentDb.OpenRecordset("SELECT * FROM information_indexes, local_time WHERE (((information_indexes.data)<>#12/31/2008#))")
Set t = CurrentDb.OpenRecordset("local_eq_delta")
Set q = CurrentDb.OpenRecordset("information_indexes")
DoCmd.SetWarnings False
DoCmd.RunSQL "Delete * From local_eq_delta"
DoCmd.SetWarnings True
' First step - Daily yields determination
Do While Not r.EOF
    t.AddNew
    t!data = r!data
    t!deltaBETI = r!BETI / q!BETI - 1
    t!deltaBETC = r!BETC / q!BETC - 1
    t!deltaBETFI = r!BETFI / q!BETFI
    t!deltaVAB = r!VAB / q!VAB - 1
    t!deltaROTXUSD = r!ROTXUSD / q!ROTXUSD - 1
    t!deltaROTXEUR = r!ROTXEUR / q!ROTXEUR - 1
    t!deltaRSQALL = r!RSQALL / q!RSQALL - 1
    t.Update
    q.MoveNext
    r.MoveNext
Loop
m.Close
t.Close q.Close r.Close
Set t = Nothing
Dim m As DAO.Recordset
Set t = CurrentDb.OpenRecordset("local_eq_delta")
Set m = CurrentDb.OpenRecordset("local_eq_variance")
' Second step - average daily yields
DoCmd.SetWarnings False
DoCmd.RunSQL "Delete * From local_eq_variance"
DoCmd.SetWarnings True
DoCmd.RunSQL "SELECT AvgOfdeltaBETI, Avg(local_eq_delta.deltaBETI) AS AvgOfdeltaBETI, AvgOfdeltaBETC, Avg(local_eq_delta.deltaBETC) AS AvgOfdeltaBETC, AvgOfdeltaBETFI AS AvgOfdeltaBETFI, Avg(local_eq_delta.deltaBETFI) AS AvgOfdeltaBETFI, Avg(local_eq_delta.deltaVAB) AS AvgOfdeltaVAB, Avg(local_eq_delta.deltaROTXUSD) AS AvgOfdeltaROTXUSD, Avg(local_eq_delta.deltaROTXEUR) AS AvgOfdeltaROTXEUR, Avg(local_eq_delta.deltaRSQALL) AS AvgOfdeltaRSQALL INTO local_eq_delta_avg"
DoCmd.SetWarnings True
' Third step - daily yields against average
Do While Not t.EOF
    m.AddNew
    m!data = t!data
    m!deltaBETI = t!deltaBETI - AvgOfdeltaBETI
    m!deltaBETC = t!deltaBETC - AvgOfdeltaBETC
    m!deltaBETFI = t!deltaBETFI - AvgOfdeltaBETFI
    m!deltaVAB = t!deltaVAB - AvgOfdeltaVAB
    m!deltaROTXUSD = t!deltaROTXUSD - AvgOfdeltaROTXUSD
    m!deltaROTXEUR = t!deltaROTXEUR - AvgOfdeltaROTXEUR
    m!deltaRSQALL = t!deltaRSQALL - AvgOfdeltaRSQALL
    m.Update
t.MoveNext
Loop
m.Close
t.Close
Set t = Nothing
End Function

The user of MSAccess application is performing a visualization of a specific report in order to see the final figures of Value of Risk. When clicking to open a specific file, the previous code is run as a macro (Table 2 – first command) and is stored in local tables.

Secondly, after the macro is run, there is generated a new procedure as follows:

Private Sub Command58_Click() 'Transfer of daily yields ag. Average in Excel, refresh driver, report visualisation'
On Error GoTo Err_Command58_Click
Dim stDocName As String
stDocName = "create_eq_variance"
DoCmd.RunMacro stDocName
End Sub

' MSAccess macro launch
Dim stDocName As String
stDocName = "create_eq_variance"
DoCmd.RunMacro stDocName

' MSExcel launch
Dim xlApp As Excel.Application
Dim xlWb As Excel.Workbook
Dim xlWk As Excel.Worksheet
DoCmd.SetWarnings False
Set xlApp = New Excel.Application
This operation of dynamically transfer of data between MSAccess and MSExcel could create an integrated environment of exploiting the database facilities by MSExcel or other MSOffice application and, on other hand, of calculus facilities by the MSAccess.

The data requests when working with matrixes is crucial. This solution of complex calculus generation in real-time is a real benefit for the end user, basically a financial modeller, who needs to solve and adjust frequently market data and formulas.

I. References


[***] IASB (2009) Standardele Internaţionale de Raportare Financiară, CECCAR.