La finance a pu se développer grâce au calcul financier. Ce calcul est aujourd'hui facile à mettre en œuvre avec les outils omniprésents comme Excel.

De nombreux environnement permettent ces calculs comme les langages de programmation de type C, C#, C++, Python, VBA, ou bien R. Nous retenons Excel, avec sa capacité à charger des librairies VBA prête à l'emploi. Nous préférons ici la mise en œuvre de ces outils plutôt que leur fabrication. L'usage va nous permettre ainsi d'accéder aux concepts financiers.

Ces calculs seront d'autant plus simples, qu'Excel permet l'usage de tables.

1–Tables et Matrices

Une table est un tableau de données à deux dimensions, avec lignes et colonnes. Disons que c'est l'équivalent informatique de la notion mathématique de matrice.

Les traitements informatiques des données vont du plus simple au plus compliqué :

- Manipulation des données avec les tableaux croisés dynamiques (TCD).
- Calcul sur tableau, identique ici au calcul matriciel, objet de cet article
- Méthodes d'analyse des données, à l'aide de méthodes factorielles, objet d'un autre article.

Nous abordons ici, le calcul matriciel à travers un ensemble de fonction de base, utilisées de manière récurrente en finance.

2–Fonctions Matricielles

Une fonction permet d'obtenir un résultat (output), à partir d'une ou plusieurs variables d'entrées (input).

Dans Excel, nous sommes habitués aux variables dites scalaires, qui représente un nombre et qui sont placées dans une cellule. Nous allons nous intéresser aux variables de type tableau : toujours des nombres, mais placés en tableaux. Nos tableaux pourront aussi n'avoir qu'une colonne ou qu'une ligne : ce seront alors des vecteurs.

Il existe peu de fonctions Excel ayant un tableau comme argument d'entrée et de sortie. Il en existe trois portant sur du traitement matriciel :

- La transposition : changement des lignes en colonnes et vice versa,
- L'inversion : calcul de la matrice inverse d'une matrice carrée, lorsque le déterminant de la matrice n'est pas nul,
- Le produit matriciel de deux matrices.

Fonctions Matricielles d'Excel	
Français :	Anglais :
TRANSPOSE(M)	TRANSPOSE
INVERSEMAT(M)	MINVERSE
PRODUITMAT(M ; N)	MMULT

L'emploi de ces fonctions est délicat, non pas dans l'identification des variables d'entrée, mais plutôt dans les variables de sortie. En effet, il est indispensable de définir par avance, la zone et donc la taille de la matrice de sortie. Une fois cette plage sélectionnée, l'utilisation d'une fonction matricielle implique l'usage simultané de 3 touches : Ctrl/Maj/Entrée. La formule devient affichée entre accolades : {}. Les cellules en sortie deviennent ainsi liées : il n'est plus possible d'introduire des lignes ou des colonnes à l'intérieur de ces plages. Il est dommage qu'il ne soit pas possible de visualiser ces plages directement sur la feuille Excel.

Il faut aussi noter, que l'usage de fonctions matricielles n'est pas limité à ces trois

fonctions. Il est ainsi possible d'utiliser des formules matricielles directement sur des plages : on peut ainsi définir une addition et une multiplication scalaire à l'aide des opérations algébriques usuelles :

Addition matricielle :
'=(A1 :C3)+(E1 :G3)
Multiplication scalaire :
'=(A1 :C3) * E1

Remarque important : la seconde formule traite deux variables de taille différente : 3x3, et 1. Dans ces cas, Excel étend automatiquement la plus petite variable à la taille de la plus grande : Cette fonction va s'avérer des plus utile pour nos calculs.

3–Réplication des Fonctions

Nous allons fabriquer trois fonctions fondamentales en plus des fonctions précédentes, fonctions qui nous semblent fort utiles pour effectuer des calculs financiers.

Tout d'abord, reprenons nos trois fonctions précédentes, pour leur donner d'autres noms et obtenir une libraire homogène : nos fonctions matricielles seront préfixée par la lettre M.
Une seule ligne de commande dans une fonction VBA, permet de redéfinir une fonction Excel, tout en utilisant le nom anglais de la fonction en question :

MTRAN(Mx) : matrice transposée de la matrice Mx,
MINVERSE(Mx) : matrice inverse de la matrice Mx,
MPROD(Mx,My) : produit matriciel entre Mx et My.

```
Function MTran(MX As Variant) As Variant
MTran = Application.WorksheetFunction.Transpose(MX)
End Function
Function MInverse(MX As Varian
MInverse = Application.WorksheetFunction.MInverse(MX)
End Function
Function MProd(MX As Variant, MY As Variant) As Variant
MProd = Application.WorksheetFunction.MMult(MX, MY)
End Function
```

Fonction Excel/Fonction VBA
Toutes les fonctions Excel sont accessibles à travers la librairie WorksheetFunction. Il existe aussi les fonctions VBA, directement accessibles.

Range/Array
Il est important de définir des fonctions sous VBA, compatibles en Excel et en VBA. Pour cela, nous devons pouvoir manipuler des plages Excel (range) de la même façon que les tableaux (array), à travers des variables de type VARIANT. Cependant, les commandes d'accès ne sont pas les mêmes :
- L'identification passe par les fonctions ISARRAY et ISRANGE – cette dernière n'existant pas, le code est le suivant :

```
Function IsRange(ByRef M As Variant) As Boolean
IsRange = (VBA.TypeName(M) = "Range")
End Function
```

- L'identification de la taille utilise respectivement UBOUND et M.rows.count,
- la copie utilise respectivement les commandes : 'V=M' et 'V=M.value'.

4–Fonctions de définition de matrice.

Voici une liste de fonctions définissant des matrices particulières qui nous sont utiles pour les calculs financiers :
MZERO(l,c) : matrice nulle, toutes les cases sont à zéro.
MUN(n,n) : matrice carrée unité, matrice nulle avec diagonale égale à l'unité 1.
MSCAL(n,k) : matrice carrée scalaire, matrice nulle avec diagonale égale à k.
MDIAG(Mx) : extraction de la diagonale d'une matrice carrée Mx – les nombres hors diagonale étant nuls.
MADD(M,k) : addition du nombre k à toutes les cases de la matrice M.
MMUL(M,k) : multiplication de toutes les cases de la matrice M par le nombre k.

5–Fonctions produit

Voici une liste de fonctions modifiant plusieurs fonctions, utilisées très souvent dans les calculs financiers.

MSPROD(Mx) : produit de la matrice transposée Mx avec elle-même : $^{T}(Mx)*(Mx)$
MTPROD(Mx,My) : produit de la transposée de la matrice Mx et de la matrice My : $^{T}(Mx)*(My)$
MQPROD(Mx) : $^{T}(Mx)*(My)*(Mx)$
MQPRODG(Mx) : $(Mx)^{-1}*(My)*(Mx)$
MQPRODD(Mx) : $(Mx)*(My)*(Mx)^{-1}$

6–Fonctions statistiques

Voici une liste de fonctions statistiques utilisées pour travailler sur des séries chronologiques.
MCENTRE(Mx) : centrage des données par colonne : calcule la moyenne de la colonne et soustrait cette moyenne à toutes les données de la colonne,
MREDUIT(Mx) : centrage et réduction : division par l'écart-type de la colonne,
MCOVARIANCE(Mx) : calcule la matrice de covariance, donc sur des données centrées.
MCORRELATION(Mx) : calcule la matrice de corrélation, donc sur des données centrées réduites.
MCHOLESKY(Mx) : calcule la matrice racine carrrée au sens de Cholesky. Cette matrice étant triangulaire inférieure.
MREND(Mx,Hor) : calcule la matrice des rendements géométrique d'horizon HOR, de chaque colonne de la matrice Mx.
MVALP(Mx) : calcule les valeurs propres de la matrice symétrique Mx,
MVECP(Mx) : calcule les vecteurs propres de la matrice symétrique Mx.

Valeurs Propres et Vecteurs Propres
Ces fonctions sont très utiles. Elles n'existent pas en Excel, mais sont faciles à programmer en VBA, en utilisant l'algorithme de Jacobi, qui s'applique uniquement sur les matrices symétriques.

7–Fonctions de nombres

Fonctions de générations de séries et de tables de nombres, utilisés pour les simulations financières :
TPASCAL(n,k) : coefficient du binôme, du triangle de Pascal,
TCATALAN(n,k) : coefficient du triangle de Catalan,
NCATALAN(k) : k ème nombre de Catalan,
NHALTON(p,k) : k ème nombre de Halton, généré à partir du nombre premier p. Ce nombre est utilisé pour simuler la méthode de quasi Monte Carlo.

8–Table des fonctions

MTran(M)	Transpose la matrice
MInverse(M)	Inverse la matrice
MProd(Mx, My)	Produit matriciel
MZero(n,m)	Matrice Zero
MUn(n)	Matrice Identité
MScal(n,k)	Matrice Scalaire de diagonale k
MDiagVP(P)	Matrice diagonale quelconque
MAdd(M,k)	Addition d'un nombre à une matrice
MMul(M,k)	Multiplication d'un nombre par une matrice
MTProd(Mx, My)	Produit transposée x matrice: tMx My
MSProd(Mx)	Produit transposée x matrice: tMx Mx
MQProd(Mx, My)	Produit quadratique: tMx My Mx
MQProdD(Mx, My)	Produit quadratique: Mx My inv(Mx)
MQProdG(Mx, My)	Produit quadratique: inv(Mx) My Mx
MCentre(M)	Matrice Centrée
MReduit(M)	Matrice Centrée Réduite
MCovariance(M)	Matrice de Covariance de M
MCorrelation(M)	Matrice de Corrélation de M
MCholesky(M)	Matrice "Racine Carrée" de Cholesky
ValP(M)	Valeurs Propres de M
VecP(M)	Vecteurs Propres de M
NHalton(p,k)	Nombres de Halton
TPascal(n,k)	Triangle de Pascal
TCatalan(n,k)	Triangle de Catalan
NCatalan(k)	Nombres de Catalan

9–Code

Fonction Cholesky

```
Function MCholesky(MX As Variant) As Variant
Dim n As Integer, k As Integer, i As Integer, j
As Integer
Dim x As Double
Dim M() As Variant
n = MLig(MX)
ReDim M(1 To n, 1 To n)
M = MZero(n, n)
For j = 1 To n
    x = 0
    For k = 1 To j - 1
        x = x + M(j, k) * M(j, k)
    Next k
    M(j, j) = MX(j, j) - x
    If M(j, j) <= 0 Then Exit For
    M(j, j) = Sqr(M(j, j))
    For i = j + 1 To n
        x = 0
        For k = 1 To j - 1
            x = x + M(i, k) * M(j, k)
        Next k
        M(i, j) = (MX(i, j) - x) / M(j, j)
    Next i
Next j
MCholesky = M
End Function
```

Fonction Valeur Propre Vecteur Propre

```
Function MValP(MX As Variant)
    Application.Volatile
    Dim i As Integer, n As Integer
    Dim vap() As Variant, vep() As Variant
    Dim E() As Variant, M() As Variant
    n = MLig(MX)
    ReDim M(n, n)
    ReDim E(n)
    M = MCopy(MX)
    Call Jacobi(M, vap, vep)
    'tri
    For i = 1 To n
        E(i) = Application.WorksheetFunction.Large(vap, i)
    Next i
    MValP = E
End Function

Function MVecP(MX As Variant)
    Application.Volatile
    Dim i As Integer, j As Integer, k As Integer, n As Integer
    Dim rk As Double
    Dim vap() As Variant, vep() As Variant
    Dim E() As Variant, M() As Variant
    n = MLig(MX)
    ReDim M(n, n)
    ReDim E(n, n)
    M = MCopy(MX)
    Call Jacobi(M, vap, vep)
    'tri
    For i = 1 To n
        rk = Application.WorksheetFunction.Large(vap, i)
        k = Application.WorksheetFunction.Match(rk, vap, 0)
        For j = 1 To n
            E(j, i) = vep(j, k)
        Next j
    Next i
    MVecP = E
End Function
'* This section provided by Prof. L. Tatum of Baruch College

Sub Jacobi(a() As Variant, d() As Variant, V() As Variant)
    Dim b() As Double, z() As Double, NROT As Integer
    Dim i As Integer, j As Integer, n As Integer
    Dim SM As Double, THRESH As Double
    Dim G As Double, H As Double, TAU As Double
    Dim t As Double, theta As Double, C As Double, S As Double

    n = MLig(a)
    ReDim d(n), V(n, n), b(n), z(n)
    Dim ip As Long, IQ As Long
    For ip = 1 To n
        V(ip, ip) = 1#
        b(ip) = a(ip, ip)
        d(ip) = b(ip)
    Next ip
    NROT = 0
    For i = 1 To 50
        SM = 0#
        For ip = 1 To n - 1
            For IQ = ip + 1 To n
                SM = SM + Abs(a(ip, IQ))
            Next IQ
        Next ip
        If (SM = 0#) Then GoTo 999
        If (i <= 4) Then
            THRESH = 0.2 * SM / (n ^ 2)
        Else
            THRESH = 0#
        End If
        For ip = 1 To n - 1
            For IQ = ip + 1 To n
                G = 100# * Abs(a(ip, IQ))
                If ((i >= 4) And (Abs(d(ip)) + G = Abs(d(ip)))
And (Abs(d(IQ)) + G = Abs(d(IQ)))) Then
                    a(ip, IQ) = 0#
                ElseIf (Abs(a(ip, IQ)) >= THRESH) Then
                    H = d(IQ) - d(ip)
                    If (Abs(H) + G = Abs(H)) Then
                        t = a(ip, IQ) / H
                    Else
                        theta = 0.5 * H / a(ip, IQ)
                        t = 1# / (Abs(theta) + Sqr(1# + theta
^ 2))
                        If (theta <= 0) Then t = -t
                    End If
                    C = 1# / Sqr(1 + t ^ 2)
                    S = t * C
                    TAU = S / (1# + C)
                    H = t * a(ip, IQ)
                    z(ip) = z(ip) - H
                    z(IQ) = z(IQ) + H
                    d(ip) = d(ip) - H
                    d(IQ) = d(IQ) + H
                    a(ip, IQ) = 0#
                    For j = 1 To ip - 1
                        G = a(j, ip)
                        H = a(j, IQ)
                        a(j, ip) = G - S * (H + G * TAU)
                        a(j, IQ) = H + S * (G - H * TAU)
                    Next j
                    For j = ip + 1 To IQ - 1
                        G = a(ip, j)
                        H = a(j, IQ)
                        a(ip, j) = G - S * (H + G * TAU)
                        a(j, IQ) = H + S * (G - H * TAU)
                    Next j
                    For j = IQ + 1 To n
                        G = a(ip, j)
                        H = a(IQ, j)
                        a(ip, j) = G - S * (H + G * TAU)
                        a(IQ, j) = H + S * (G - H * TAU)
                    Next j
                    For j = 1 To n
                        G = V(j, ip)
                        H = V(j, IQ)
                        V(j, ip) = G - S * (H + G * TAU)
                        V(j, IQ) = H + S * (G - H * TAU)
                    Next j
                    NROT = NROT + 1
                End If
            Next IQ
        Next ip
        For ip = 1 To n
            b(ip) = b(ip) + z(ip)
            d(ip) = b(ip)
            z(ip) = 0#
        Next ip
    Next i
999
End Sub
```

-/-/-