

Théorie du portefeuille et gestion du risque : application en VBA

Emmanuel Dérivé-Désgardes
emmanuelderibere@hotmail.com
2020 - 2021

Le sujet de ce cours est orienté principalement sur les techniques quantitatives de gestion de portefeuille incluant le choix du portefeuille, la mesure de la performance, l'évaluation des actifs actions et la mesure du risque. Chaque chapitre propose une application en VBA.

Pièce jointe :

Portefeuille.xlsm.

Plan:

PREMIERE PARTIE : Optimisation statique & risque

- 1- Analyse du couple rendement/risque : le cas d'un seul actif risqué*
- 2- Analyse du couple rendement/risque : le cas de n actifs risqués*
- 3- Optimisation du couple rendement/risque et choix du portefeuille efficient au sens de Markowitz*
- 4- L'annualisation*
- 5- Gestion indicielle et benchmarkée*
- 6- Méthodes de sélection des actions*
- 7- Mesure et attribution de la performance*
- 8- Autres indicateurs de mesure du risque : le Value at Risk (VaR)*

DEUXIEME PARTIE : Optimisation dynamique

- 1- Stop Loss*
- 2- OBPI*
- 3- CPPI*

ANNEXE

Références :

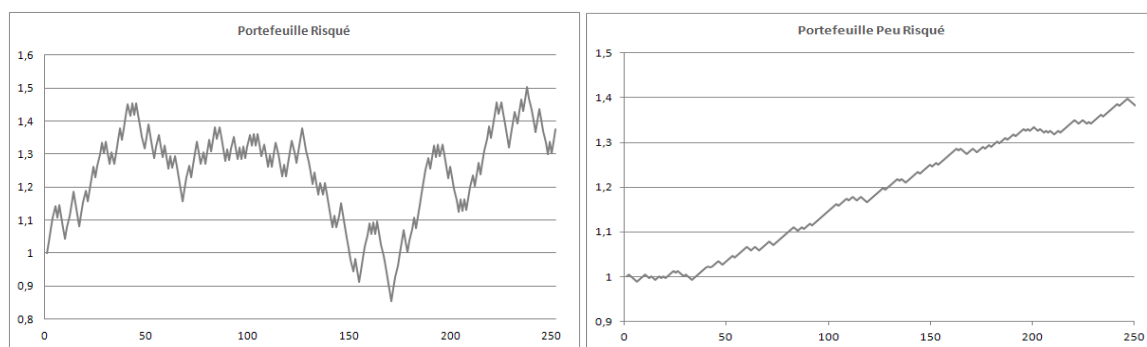
DALLOZ - Finance de Marché - P. Poncet & R. Portait
PEARSON - Options, futures et autres actifs dérivés 8e édition - J. Hull
ECONOMICA - Gestion de Portefeuille, Analyse quantitative et gestion structurée - Bertrand, Prigent
WILEY - Advanced Modeling in Finance using Excel and VBA - P. Jackson, M. Stauton
ENI - VBA Excel 2013, M. Amelot, C. Duigou, H. Laugie

PREMIERE PARTIE : Optimisation statique & risque

1 – Analyse du couple rendement/risque, le cas d'un seul actif risqué:

- 1952 Markowitz : ouverture de l'approche quantitative aux problématiques de gestion de portefeuille.

Une nouvelle notion apparaît : celle du risque mesuré par la variance. Par conséquent, au sens de Markowitz, le portefeuille optimum n'est pas uniquement celui qui fournit le rendement le plus important. La théorie moderne du portefeuille implique de prendre en considération, en plus du rendement, le risque mesuré par la variance. Cette approche est aussi désignée par l'analyse du couple moyenne/variance ou couple rendement/risque. La traduction du problème imposé par ces deux indicateurs, l'un de création de richesse et l'autre de prise de risque (c'est-à-dire de destruction potentielle de richesse) est simple : on doit choisir les actifs financiers et leur pondération afin d'obtenir le minimum de risque et le maximum de rendement. Une représentation graphique de deux portefeuilles, l'un risqué, l'autre peu risqué, aide à comprendre facilement ce problème :



Intuitivement il est préférable d'investir ses avoirs sur le portefeuille peu risqué plutôt que sur le portefeuille risqué. Pour une création de richesse à peu près équivalente sur 250 jours, l'amplitude des variations du graphique de droite est très peu importante comparée au graphique de gauche, ce qui implique que la probabilité de perte et surtout le montant potentiel de cette perte est moindre.

- Gestion standard de portefeuille : cas à un seul actif risqué.

Les méthodes classiques de gestion de portefeuille liées à la théorie du portefeuille impliquent donc une analyse moyenne de la variance. Cette analyse repose sur une hypothèse forte : celle que les investisseurs soient rationnels. Autrement dit, qu'ils préfèrent investir dans le portefeuille peu risqué que dans le portefeuille risqué. Par conséquent, et c'est tout l'intérêt de cette approche, on peut définir un actif uniquement par le rendement moyen qu'il procure en contrepartie de la prise de risque qu'il génère. Cette assertion à la base de la théorie moderne du portefeuille est en fait bien antérieure à 1952, puisqu'elle est déjà étudiée dans la thèse de Louis Bachelier au début du XXème siècle. En effet Louis Bachelier propose de modéliser le comportement d'un cours boursier à l'aide de l'EDS suivante :

$$\frac{dS_t}{S_t} = \mu dt + \sigma dz_t$$

On retrouve bien dans ce processus de diffusion la moyenne, μ et la racine carré de la variance, σ .

- Couple moyenne/variance : quelle moyenne ?

Prenons l'exemple des valeurs d'une action enregistrées de manière quotidienne sur une période de 1 an soit 252 jours ouvrés. Ces valeurs sont exprimées en base 100, nous les retrouvons dans le tableau a-/.

a-/		b-/	
jours	Indice	jours	Indice
1	100	1	-
2	98	2	-2%
3	95	3	-3%
4	99	4	4%
5	104	5	5%
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
252	120	252	X%

Dans le tableau b-/, nous avons calculé la performance quotidienne de l'indice à l'aide de la formule du taux de variation suivante, qui indique tout simplement l'évolution de l'indice en pourcentage :

$$Ri = \frac{V_i}{V_{i-1}} - 1$$

avec : Ri le rendement en date i , V_i la valeur de l'indice en date i , $V(i-1)$ la valeur de l'indice en date $i-1$

Par exemple, en partant du tableau a-/ et en appliquant la formule précédente, le calcul du taux de variation de l'indice entre le jour 1 et le jour 2 s'écrit : $98/100 - 1 = -0.02$ (ou -2%). La performance quotidienne ainsi calculée, on cherche à connaître la performance moyenne de l'actif sur une période de temps déterminée. Il peut arriver que les *sales* de société de gestion de portefeuilles présentent la performance moyenne avec le calcul d'une moyenne arithmétique soit :

$$\mu_{indice} = \frac{1}{n-1} * \sum_{i=2}^n Ri$$

avec : n le nombre total de jours

Ainsi en partant des tableaux a-/ et b-/ et en calculant la performance moyenne arithmétique sur les 5 premiers jours, on obtient : $1/4 * (-0.02 - 0.0306 + 0.0421 + 0.0505) = 0.0105$ (ou 1.05%). Ce résultat s'interprète comme la variation moyenne quotidienne de l'actif sur une durée de 5 jours. Par conséquent, en plaçant 100 € sur notre indice au jour 1, on devrait obtenir le deuxième jour $100 * (1+1.05\%) = 101.05$ €. Le troisième jour on devrait obtenir $101.05 * (1+1.05\%) = 100 * (1+1.05\%)^2 = 102.11$ €. En répétant ce calcul jusqu'au dernier jour on devrait obtenir comme montant 104,26€. La formule utilisée est celle de la capitalisation des intérêts. De manière plus générale elle s'écrit comme ceci :

$$Vn = Vo * (1 + k)^n$$

avec : V_n la valeur de l'actif en $t = n$, V_0 la valeur de l'actif en $t = 0$, k la performance moyenne de l'actif pour un pas de temps donné (ici quotidien)

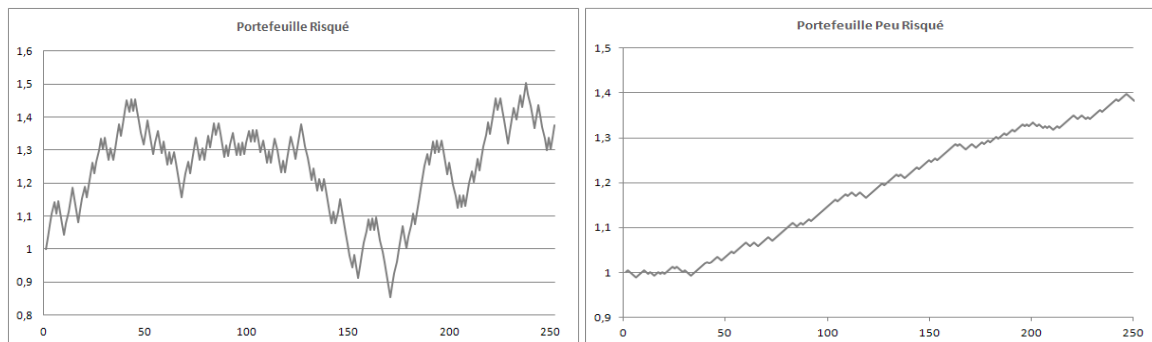
Ce résultat appelle une remarque importante car il est différent de la valeur de l'action (cf. tableau a-/) le cinquième jour qui est, rappelons le, égal à 104. En réalité, la moyenne arithmétique n'est pas la mesure appropriée pour un calcul de performance financière. En effet le calcul à utiliser pour obtenir un résultat correct est celui de la moyenne géométrique. La formule de la moyenne géométrique est la suivante :

$$\mu_{indice} = \left[\prod_{i=2}^n (1 + R_i) \right]^{\frac{1}{n-1}} - 1$$

L'application numérique de cette formule nous permet d'obtenir le résultat suivant : $[(1-0.02)*(1-0.0306)*(1+0.0421)*(1+0.0505)]^{1/4} - 1 = 0.00985$ (ou 0.985%). Nous pouvons vérifier ce calcul de la manière suivante : si l'on place 100 € sur l'indice le jour 1, on obtient le cinquième jour : $100*(1+0.958\%)^4 = 104$, qui correspondent bien à la valeur de l'indice le jour 5.

- Couple moyenne/variance : la variance.

La variance mesure le risque que l'on prend en investissant sur un actif. D'un point de vue statistique, c'est un indicateur de dispersion, c'est-à-dire qu'il indique de quelle manière une variable aléatoire va évoluer autour de sa moyenne, appelée aussi espérance mathématique. Par conséquent une variance peu élevée est un signe que les valeurs sont proches les unes des autres alors qu'une variance élevée est signe qu'elles sont écartées. En reprenant les deux graphiques on peut en déduire que :



$$\mu_{portfeuille risqué} = \mu_{portfeuille peu risqué}$$

mais que :

$$\sigma_{portfeuille risqué}^2 > \sigma_{portfeuille peu risqué}^2$$

La formule mathématique de la variance est la suivante :

$$\sigma^2 = \frac{\sum_{i=1}^n (R_i - \bar{R})^2}{n - 1}$$

avec \bar{R} la performance moyenne de l'actif

La mesure du risque que l'on utilisera par la suite est déduite directement de la variance puisqu'il s'agit de l'écart-type dont la formule est donnée par la racine carré de la variance :

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (R_i - \bar{R})^2}{n - 1}}$$

- Calcul de la moyenne et de la variance d'un actif en VBA : mise en place du projet.

Ce que nous appelons ici projet, consiste en la mise en place par étape (et de manière très structurée) d'une solution simple permettant d'effectuer de manière automatisée tous les calculs que nous verrons dans ce cours. Nous invitons le lecteur à consulter l'annexe (encadré 1), qui représente la solution voulue sous forme de schéma. Celui-ci servira de référence tout au long de ce cours. L'avantage du VBA est que, pour les projets relativement peu consommateurs de mémoire, on peut se servir du tableur Excel pour stocker les données. Pour des projets de plus grande ampleur, on pourra abandonner le tableur Excel en intégrant des fonctions de gestion de fichier txt, csv ou SQL à l'aide d'une librairie complémentaire. Dans notre cas de figure, nous allons partir sur un format simple : celui de la feuille Excel. Pour le calcul du rendement, la fonction développée est la suivante. Cette fonction permet de choisir le pas de temps ainsi que le type de calcul de moyenne voulu c'est à dire arithmétique ou bien géométrique. (Cf. Portefeuille.xlsm, Module mdlLibrary).

```
Function Get_AverageReturnRdt(RdtVector, tenor As Integer, pgs As Integer) As Double
'EDD
'Return the arithmetic or geometric Average return for a specified tenor (in day). if pgs = 1 -> arithmetic ->2 geometric
Dim i As Long
    Select Case pgs
        Case 1
            For i = LBound(RdtVector) + tenor To UBound(RdtVector)
                Get_AverageReturnRdt = Get_AverageReturnRdt + RdtVector(i, 1)
            Next i
            Get_AverageReturnRdt = Get_AverageReturnRdt / (UBound(RdtVector) - tenor)
            Exit Function
        Case 2
            Get_AverageReturnRdt = 1
            For i = LBound(RdtVector) + tenor To UBound(RdtVector)
                Get_AverageReturnRdt = Get_AverageReturnRdt * (1 + RdtVector(i, 1))
            Next i
            Get_AverageReturnRdt = Get_AverageReturnRdt ^ (1 / (UBound(RdtVector) - tenor)) - 1
            Exit Function
    End Select
End Function
```

Pour le calcul de la volatilité, la fonction développée est celle-ci : (Cf. Portefeuille.xlsm, Module mdlLibrary).

```
Function Get_VarianceRdt(RdtVector) As Double
'03/10/2014
'Return the volatility measure by variance
'sigma^2 = (sum (i=1 -> n) (xi - xbarre)^2)/n
Dim i As Long
Dim xi_xbarreP2, Xi, xbarre
xi_xbarreP2 = 0
    'xbarre
    xbarre = Get_AverageReturnRdt(RdtVector, 0, 2)
    'sum (xi-xbarre)^2
    For i = LBound(RdtVector) + 1 To UBound(RdtVector)
        'xi
        Xi = RdtVector(i, 1)
        xi_xbarreP2 = xi_xbarreP2 + (Xi - xbarre) ^ 2
    Next i
    Get_VarianceRdt = (xi_xbarreP2 / (UBound(RdtVector) - 1))
End Function
```

Ces fonctions (comme toutes les fonctions que nous allons développer) peuvent être appelées de deux manières différentes. La première consiste à appeler la fonction via le tableur, et la seconde consiste à écrire une procédure

qui va jouer le rôle « d'appel de la fonction ». Nous proposons la procédure d'appel et d'écriture des résultats suivante (CF « Projet2.xlsm »- Module «mdlPrincipal ») :

```
Sub Calcul_indice()
'-----
'Procédure Principale :
' -> Appel des fonctions de calcul + écriture des résultats
'Date :
'Auteur : EDD
'-----
Dim vVecteurCotation As Variant
Dim dPerfMoyenneGeoIndice As Double, dPerfMoyenneArithIndice As Double, dVariance As Double, dStdev As Double
Dim iPasDeTemps As Integer, colonne As Integer
Dim sIndice As String

With Sheets("AnalysePtf")
    iPasDeTemps = .Range("Pas_de_Temps").Cells(1, 2).Value
    sIndice = .Range("NomIndice").Cells(1, 2).Value
End With
With Sheets("Data")
    colonne = 2
    Do Until .Cells(1, colonne).Value = sIndice
        colonne = colonne + 1
    Loop
    vVecteurCotation = .Range(.Cells(2, colonne), .Cells(2, colonne).End(xlDown)).Value
End With

'Calcul de la Performance Moyenne de l'indice
dPerfMoyenneArithIndice = Get_AverageReturn(vVecteurCotation, iPasDeTemps, 1)
dPerfMoyenneGeoIndice = Get_AverageReturn(vVecteurCotation, iPasDeTemps, 2)

'Calcul de la Variance de l'indice
dVariance = Get_Variance(vVecteurCotation)
dStdev = Get_StDev(dVariance)

'Ecriture du résultat
With Sheets("AnalysePtf")
    .Range("Report_Perf").Cells(1, 2) = dPerfMoyenneArithIndice
    .Range("Report_Perf").Cells(2, 2) = dPerfMoyenneGeoIndice
    .Range("Report_Risk").Cells(1, 2) = dVariance
    .Range("Report_Risk").Cells(2, 2) = dStdev
End With

End Sub
```

2 – Analyse du couple rendement/risque, le cas à n actifs risqués :

- Merton 1972 : cette présentation repose essentiellement sur l'article de Merton (1972).

Considérons dans un premier temps un marché financier composé de n actifs risqués. L'univers d'investissement contient n titres financiers risqués indexés par $i=1, \dots, n$. On peut donc, si on le souhaite, les intégrer dans notre portefeuille. Notons \vec{w} le vecteur des pondérations des actifs du portefeuille P :

$$\vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$$

Notons \vec{R} Le vecteur des rentabilités moyennes des actifs constituant le portefeuille :

$$\vec{R} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{pmatrix}$$

Notons \vec{e} Le vecteur unitaire :

$$\vec{e} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

Notons \vec{V} La matrice des variances covariances des rentabilités des actifs :

$$\vec{V} = \begin{pmatrix} \sigma_{1,1} & \cdots & \sigma_{1,n} \\ \vdots & \ddots & \vdots \\ \sigma_{n,1} & \cdots & \sigma_{n,n} \end{pmatrix}.$$

Par conséquent, la rentabilité espérée du portefeuille P contenant les n actifs risqués est égale à :

$$E_{(Rp)} = \sum_{i=1}^n w_i E[R_i] = \vec{w} \cdot \vec{R}$$

La variance de la rentabilité du portefeuille P est donnée par l'expression

$$\sigma^2(R_p) = \sum_{i=1}^n \sum_{j=1}^n w_i w_j \sigma_{ij} = \vec{w}' \cdot \vec{V} \cdot \vec{w}$$

avec :

$$\sigma_{ij} = \frac{\sum_{i=1}^n (R_i - \bar{R}_i) * (R_j - \bar{R}_{ji})}{n - 1}$$

NB : Concernant ces calculs, les valeurs sont correctes dès lors que les pondérations des portefeuilles sont considérées comme figées, c'est-à-dire qu'il n'y a pas d'achat/vente sur la période de calcul. Dans le cas contraire il est préférable de passer par un modèle prenant en compte l'évolution des pondérations du portefeuille (par exemple de type Brinson / Grap que nous verrons par la suite).

- Code VBA pour la construction du vecteur des rendements moyens et de la matrice des variances covariances. Déduction de la performance globale et de la variance globale. (CF « Projet2.xlsm »- Module «mdlLibrary ») . La fonction de calcul du vecteur des performances moyennes est la suivante :

```
Function Get_VecteurPerf(vPriceMatrice As Variant, pgs As Integer) As Variant
    Dim i As Long
    Dim vVecteurPerformance() As Variant
    ReDim vVecteurPerformance(1 To UBound(vPriceMatrice, 2), 1 To 1)
    Dim PriceVector()
    For i = LBound(vPriceMatrice, 2) To UBound(vPriceMatrice, 2)
        'PriceVector
        ReDim PriceVector(LBound(vPriceMatrice) To UBound(vPriceMatrice), 1 To 1)
        For j = LBound(vPriceMatrice) To UBound(vPriceMatrice)
            PriceVector(j, 1) = vPriceMatrice(j, i)
        Next j
        'xbarre
        vVecteurPerformance(i, 1) = Get_AverageReturn(PriceVector, 1, pgs)
    Next i
    Get_VecteurPerf = vVecteurPerformance
End Function
```

La fonction de calcul de la matrice des variances covariances est la suivante :

```

Function Get_VarCovarMatrix(vPriceMatrice As Variant) As Variant
'EDD
'Return the matrix volatility mesured by standard deviation
'VarCovarMatrix = (sum(i=1->n, sum j=1 ->n) * sigma(i,j)
Dim i As Long, j As Long, k As Long
Dim Xi, xbarre, xi_xbarre
Dim yi, ybarre, yi_ybarre
Dim covij
Dim PriceVectori(), PriceVectorj()
Dim MatriceVarCvar()
ReDim MatriceVarCvar(1 To UBound(vPriceMatrice, 2), 1 To UBound(vPriceMatrice, 2))

For j = LBound(vPriceMatrice, 2) To UBound(vPriceMatrice, 2)
'PriceVector x
ReDim PriceVectorj(LBound(vPriceMatrice) To UBound(vPriceMatrice), 1)
For k = LBound(vPriceMatrice) To UBound(vPriceMatrice)
PriceVectorj(k, 1) = vPriceMatrice(k, j)
Next k
'xbarre
xbarre = Get_AverageReturn(PriceVectorj, 1, 2)
For i = LBound(vPriceMatrice, 2) To UBound(vPriceMatrice, 2)
'Pricevector y
ReDim PriceVectori(LBound(vPriceMatrice) To UBound(vPriceMatrice), 1)
For k = LBound(vPriceMatrice) To UBound(vPriceMatrice)
PriceVectori(k, 1) = vPriceMatrice(k, i)
Next k
'ybarre
ybarre = Get_AverageReturn(PriceVectori, 1, 2)

'covariance formula
covij = 0
For k = LBound(vPriceMatrice) + 1 To UBound(vPriceMatrice)
Xi = vPriceMatrice(k, j) / vPriceMatrice(k - 1, j) - 1
yi = vPriceMatrice(k, i) / vPriceMatrice(k - 1, i) - 1
covij = covij + (Xi - xbarre) * (yi - ybarre)
Next k
civij = covij / (UBound(vPriceMatrice) - 1)
MatriceVarCvar(j, i) = civij

Next i
Next j
Get_VarCovarMatrix = MatriceVarCvar
End Function

```

La fonction de calcul de la performance globale est la suivante :

```

Function Get_PerfGlobale(vecteurpoids As Variant, vecteurReturn As Variant) As Variant
Dim ResultatMmult As Variant
ResultatMmult = Application.MMult(vecteurpoids, vecteurReturn)
Get_PerfGlobale = ResultatMmult(1)
End Function

```

La fonction de calcul de la variance globale est la suivante :

```

Function Get_VarianceGlobale(vecteurpoids As Variant, MatriceVcV As Variant) As Variant
Dim resultat1 As Variant, resultat2 As Variant
resultat1 = Application.MMult(vecteurpoids, MatriceVcV)
resultat2 = Application.MMult(resultat1, Application.Transpose(vecteurpoids))
Get_VarianceGlobale = resultat2(1)
End Function

```

Comme précédemment, nous avons écrit dans le module principal une procédure permettant d'appeler les fonctions de calcul de la performance du portefeuille et de la variance du portefeuille.(CF « Projet2.xlsm »- Module «mdlPrincipal »).


```

Sub Calcul_Portefeuille()
'-----
'Procédure Principale :
' -> Appel des fonctions de calcul + écriture des résultats
'Date :
'Auteur :EDD
'-----

Dim vMatriceCotation As Variant, vVecteurPoids As Variant, vPonderation As Variant, vTPonderation As Variant
Dim vMatriceVarianceCovariance As Variant, vVariancePtf As Variant, vVecteurPerformanceArithm As Variant, vPerformancePtfA As Variant,
Dim bAfficheMatriceVCV As Boolean
Dim i As Integer, j As Integer

With Sheets("Data")
    vMatriceCotation = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
End With
With Sheets("AnalysePtf")
    vPonderation = .Range("Pondérations").Value
    bAfficheMatriceVCV = Range("Affichage_Matrice_VCV").Cells(1, 2).Value
End With

'calcul de la Matrice des variances covariances et de la variance globale du portefeuille
vMatriceVarianceCovariance = Get_VarCovarMatrix(vMatriceCotation)
vVariancePtf = Get_VarianceGlobale(Application.Transpose(vPonderation), vMatriceVarianceCovariance)

'calcul du vecteur des Performances moyennes et calcul de la Performance Globale
vVecteurPerformanceArithm = Get_VecteurPerf(vMatriceCotation, 1)
vPerformancePtfA = Get_PerfGlobale(Application.Transpose(vPonderation), vVecteurPerformanceArithm)

vVecteurPerformanceGeo = Get_VecteurPerf(vMatriceCotation, 2)
vPerformancePtfG = Get_PerfGlobale(Application.Transpose(vPonderation), vVecteurPerformanceGeo)

'Ecriture du résultat
With Sheets("AnalysePtf")
    .Range("Report_PerfPtf").Cells(1, 2) = vPerformancePtfA
    .Range("Report_PerfPtf").Cells(2, 2) = vPerformancePtfG
    .Range("Report_RiskPtf").Cells(1, 2) = vVariancePtf
    .Range("Report_RiskPtf").Cells(2, 2) = vVariancePtf ^ (1 / 2)
    If bAfficheMatriceVCV = True Then
        For j = LBound(vMatriceVarianceCovariance) To UBound(vMatriceVarianceCovariance)
            For i = LBound(vMatriceVarianceCovariance, 2) To UBound(vMatriceVarianceCovariance, 2)
                .Range("MVCV").Cells(i, j).Value = vMatriceVarianceCovariance(i, j)
            Next i
        Next j
        For j = LBound(vVecteurPerformanceGeo) To UBound(vVecteurPerformanceGeo)
            .Range("VecteurPerf").Cells(1, j).Value = vVecteurPerformanceGeo(j, 1)
        Next j
    End If
End With
End Sub

```

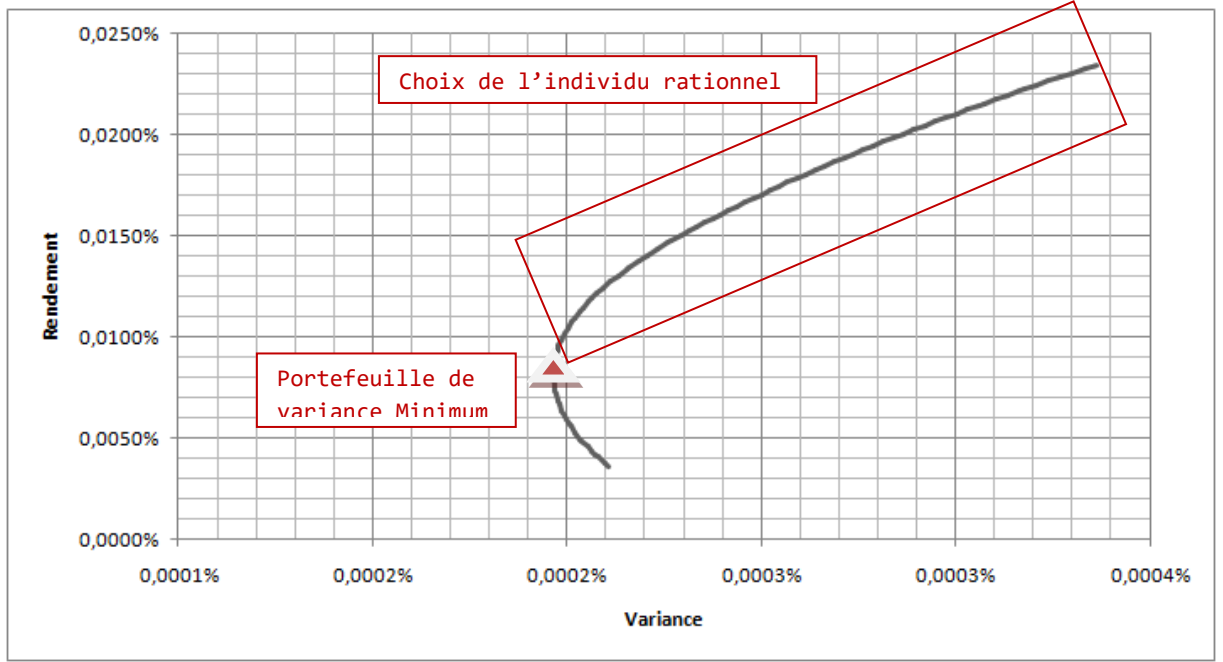
3 – Optimisation du couple rendement/risque et choix du portefeuille efficient au sens de Markowitz:

- Programme d'optimisation sous contrainte de Markowitz.

Idée : On détermine l'ensemble des portefeuilles qui minimisent la variance pour une espérance de rentabilité fixée, avec le programme d'optimisation quadratique suivant :

$$\begin{cases} \min \vec{w}'\vec{V}\vec{w} \\ \text{sc : } \vec{w}'\vec{R} = E[Rp] \\ \vec{w}'\vec{e} = 1 \end{cases}$$

Par conséquent, les contraintes sont que le rentabilité espérée du portefeuille est fixée et que la somme des poids est égale à 100%, ce qui interdit l'effet de levier. Dans un tel cas de figure, le programme d'optimisation autorise les pondérations négatives, ce qui veut dire que les ventes à découvert sont autorisées. Par conséquent, en faisant varier l'espérance de rendement du portefeuille $E[R_p]$, et en déterminant le portefeuille de variance minimale correspondant, on engendre la frontière de variance minimale, c'est-à-dire la frontière efficiente.



Donc le portefeuille de variance minimale peut s'écrire :

$$\vec{w}_{opti} = \begin{pmatrix} w1_{opti} \\ w2_{opti} \\ . \\ w3_{opti} \end{pmatrix}$$

La résolution proposée peut se faire à partir d'une formule fermée, à partir du Lagrangien. Le Lagrangien du problème d'optimisation s'écrit :

$$L(w, \lambda, \delta) = \vec{w}' \cdot \vec{V} \cdot \vec{w} + \lambda(E(R_p) - \vec{w}' \vec{R}) + \delta(1 - \vec{w}' \vec{e})$$

Avec λ, δ les multiplicateurs de Lagrange. Le programme d'optimisation sous contrainte peut s'écrire alors :

$$\min_{\{w, \lambda, \delta\}} L(w, \lambda, \delta) = \vec{w}' \cdot \vec{V} \cdot \vec{w} + \lambda(E(R_p) - \vec{w}' \vec{R}) + \delta(1 - \vec{w}' \vec{e})$$

Après résolution du programme (Cf. démonstration en annexe) il vient :

$$w = \frac{1}{d}(BV^{-1}e - AV^{-1}\bar{R}) + E(R_p) \frac{1}{d}(CV^{-1}\bar{R} - AV^{-1}e)$$

avec :

$$A = e'V^{-1}\bar{R}, B = \bar{R}'V^{-1}\bar{R}, C = e'V^{-1}e, BC - A^2$$

Il est intéressant de constater que tout portefeuille de la frontière efficiente s'écrit comme une combinaison linéaire de deux portefeuilles spéciaux, soit :

$$w = w_1 + E(R_p).w_2$$

L'expression du portefeuille optimal permet de déterminer la relation entre le risque et la rentabilité espérée du portefeuille.

$$\sigma(R_p) = \sqrt{\frac{1}{d}(CE(R_p)^2 - 2AE(R_p) + B)}$$

Cette équation est celle d'une hyperbole, forme de la frontière efficiente et qui illustre la relation fondamentale entre le risque et le rendement au sens de Markowitz. L'une des limites de l'exercice est que, lors de la mise en place de contraintes supplémentaires les solutions explicites n'existent pas forcément pour ce type de problème. Par conséquent il est intéressant d'utiliser d'autres techniques comme par exemple des méthodes de résolution numérique permettant d'approcher précisément la solution.

- Contraintes supplémentaires et résolution à l'aide d'un calcul itératif.

En plus des contraintes spécifiées précédemment, nous allons intégrer en plus une contrainte supplémentaire de non vente à découvert. Le programme d'optimisation à résoudre devient le suivant :

$$\left\{ \begin{array}{l} \text{Min } \vec{w}'\vec{V}\vec{w} \\ \text{sc : } \vec{w}'\vec{R} = E[R_p] \\ \vec{w}'\vec{e} = 1 \\ \forall i \ w_i \geq 0 \end{array} \right.$$

La résolution du problème peut se faire à l'aide du solveur Excel, ou bien à l'aide d'une simulation de Monte-Carlo sur les pondérations.

- Intégration d'un actif sans risque dans le portefeuille :

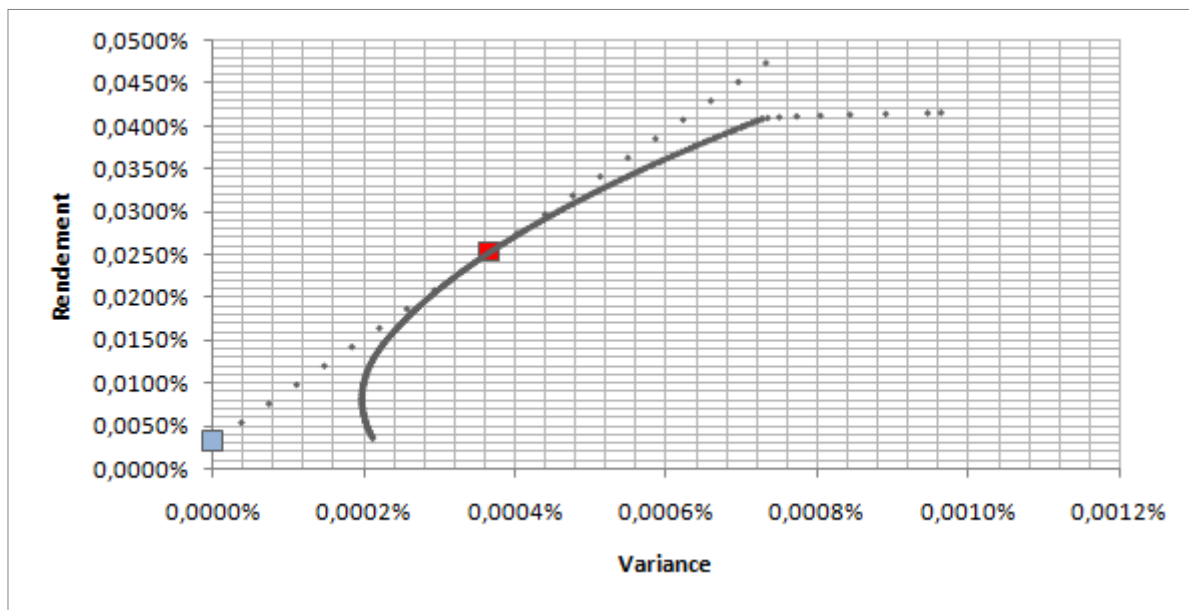
Le vecteur \vec{w} représente toujours les poids des n actifs risqués dont le vecteur des espérances de rentabilité est noté \vec{R} . L'actif sans risque est noté w_0 . Par conséquent, le programme d'optimisation que doit résoudre un investisseur s'écrit désormais :

$$\left\{ \begin{array}{l} \text{Min } \vec{w}'\vec{V}\vec{w} \\ \text{sc : } \vec{w}'\vec{R} + (1 - w'e)R_f = E[R_p] \\ \vec{w}'\vec{e} = 1 \\ \forall i \ w_i \geq 0 \end{array} \right.$$

L'introduction d'un actif sans risque soulève un problème différent en ce sens que l'investisseur aura le choix sur le partage de son investissement entre un portefeuille d'actifs risqué et l'actif sans risque. La résolution du programme peut être envisagée de la manière suivante :

- 1- On détermine la combinaison du portefeuille optimal
- 2- On partage l'investissement entre l'actif sans risque et le portefeuille risqué optimisé.

La solution au premier point permet d'obtenir les pondérations du portefeuille. Puis, l'actif sans risque est alors combiné au portefeuille optimisé. Sur le graphique suivant, nous obtenons la représentation graphique de la « nouvelle » frontière efficiente, intégrant l'actif sans risque ainsi que l'ancienne n'intégrant que le portefeuille risqué.



Remarques :

En présence de l'actif sans risque, la frontière des portefeuilles optimaux est la réunion de deux demies droites dont l'origine est le point de coordonnées (0, R_f). Ici nous n'en avons tracé qu'une seule, nous l'appelons CML (Capital Market Line). La forme de la nouvelle frontière s'explique par le fait que la réunion de l'actif sans risque et du portefeuille risqué est une combinaison linéaire et non pas quadratique. En effet la covariance entre les deux est nulle (la variance de l'actif sans risque étant nulle). La performance du portefeuille de variance minimum est logiquement supérieure à celle de l'actif sans risque sinon, il y a arbitrage.

- Choix d'un unique portefeuille et intégration de la fonction d'utilité :

En partant du principe que les investisseurs sont rationnels, le critère moyenne variance peut servir à choisir un unique portefeuille efficace. Par conséquent il conviendra d'étudier la valeur d'utilité d'un portefeuille sélectionné. Cette valeur d'utilité peut donc s'écrire sous forme d'une fonction de l'espérance et de la variance du portefeuille :

$$V(R_p) = E(R_p) - \frac{a}{2} * \sigma^2(R_p)$$

A se définit comme de degré d'aversion pour le risque de l'investisseur. Ainsi le programme d'optimisation d'un investisseur caractérisé par un certain degré d'aversion pour le risque peut s'écrire :

$$\begin{cases} \text{Max } w'R - \frac{\sigma}{2} * w'Vw \\ \text{sc} \\ w'e = 1 \end{cases}$$

• Résolution du programme d'optimisation sous contrainte à l'aide du Solver Excel (le cas à n actifs risqués et un actif sans risque n'est pas présenté ici. Cf le fichier excel pour le détail du code) :

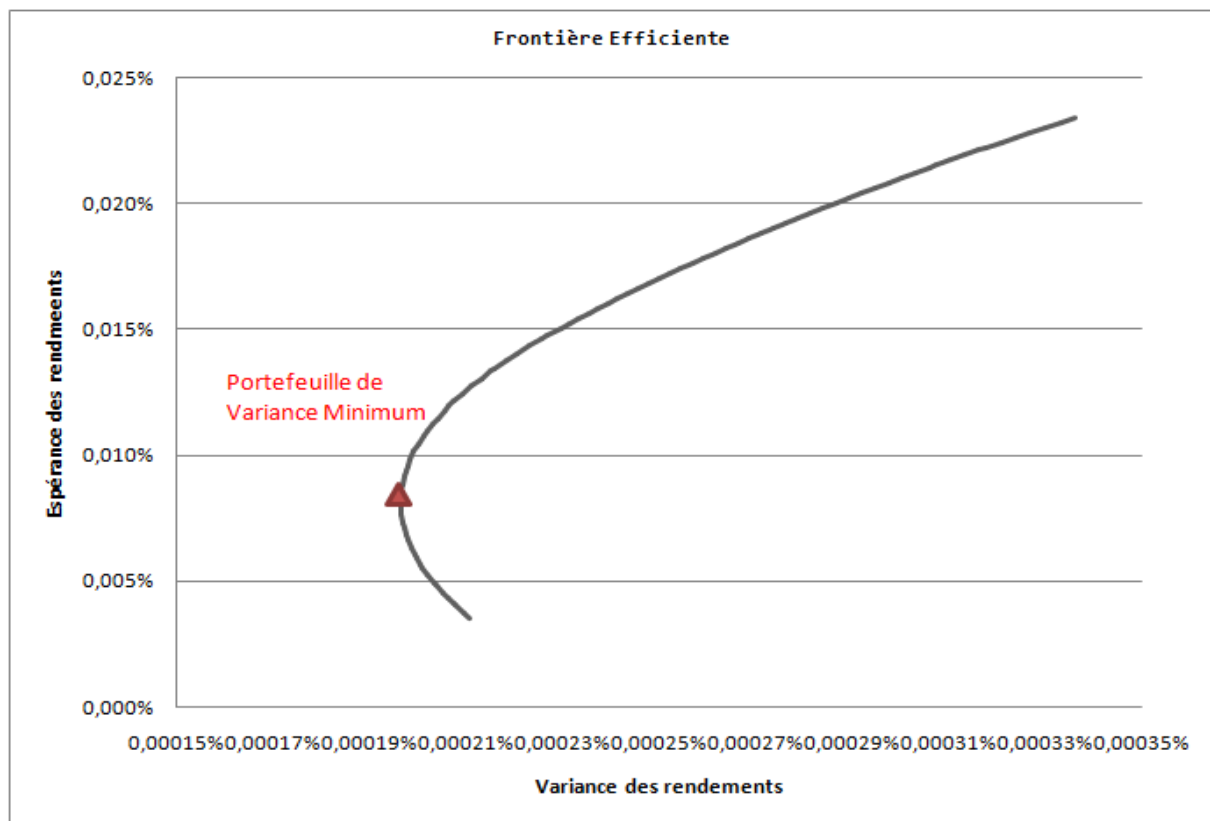
```
' Markowitz Contraint
Sub Opti_Mktz()
    Application.ScreenUpdating = False
    With Sheets("AnalysePtf")
        Matricepoids = .Range("Pondérations").Address(True, True, xlR1C1, True)
        MatricePerf = .Range(.Cells(46, 4), .Cells(46, 4).End(xlToRight)).Address(True, True, xlR1C1, True)
        Matricevol = .Range(.Cells(38, 4), .Cells(38, 4).End(xlToRight).End(xlDown)).Address(True, True, xlR1C1, True)

        .Range("InputOpti").Cells(1, 2).FormulaArray = _
        "=MMULT(" & MatricePerf & "," & Matricepoids & ")"
        .Range("InputOpti").Cells(2, 2).FormulaArray = _
        "=MMULT(" & "MMult(TRANSPOSE(" & Matricepoids & ")"," & Matricevol & ")"," & Matricepoids & ")"
        .Range("InputOpti").Cells(3, 2).FormulaArray = "=Sum(" & Matricepoids & ")"
        'valeur initiale par défaut
        .Range("Choix_Performance_Portefeuille").Value = 0
    SOLVER.Auto_open
    solverreset

    '2 -> Equivaut à "="
    Call Solveradd(.Range("Performance_Portefeuille"), 2, .Range("Choix_Performance_Portefeuille"))
    'contrainte supplémentaire i3 = 100% -> manuel
    .Range("Contrainte_Somme_Allocation") = "&"
    Call Solveradd(.Range("Somme_Allocation_Actuelle"), 2, .Range("Contrainte_Somme_Allocation"))
    .Range("Contrainte_Somme_Allocation").Value = 1
    'cellule cible à définir : Vol Ptf, cellules variables -> Range(Cells(3, 2), Cells(3, 7) , contrainte 2,0 -> >=
    Call SolverOk(.Range("Variance_Portefeuille"), 2, 0, Range("Pondérations"))
    'contrainte supplémentaire pas de resultat <0 -> manuel
    SolverOptions AssumeNonNeg:=True

    perfajout = 0.000001
    Iteratio = 1
    Range("Choix_Performance_Portefeuille").Value = 0.000035492
    Call SolverSolve(True)
    Do While Iteratio <= 200
        Application.ScreenUpdating = False
        solverfinish
        For i = 1 To .Range("Pondérations").Cells.Count
            .Range("Allocation_Cible").Cells(Iteratio, i).Value = .Range("Pondérations").Cells(i, 1).Value
        Next i
        .Range("Performance_Cible").Cells(Iteratio, 1).Value = Range("Performance_Portefeuille").Value
        .Range("Variance_Cible").Cells(Iteratio, 1).Value = Range("Variance_Portefeuille").Value
        Range("Choix_Performance_Portefeuille").Value = Range("Choix_Performance_Portefeuille").Value + perfajout
        Call solverchange(.Range("Performance_Portefeuille"), 2, .Range("Choix_Performance_Portefeuille"))
        Call SolverSolve(True)
        Iteratio = Iteratio + 1
    Loop
End With
End Sub
```

Pour chaque rendement spécifié, le calcul itératif du Solveur Excel permet d'obtenir le portefeuille (c'est-à-dire les pondérations) de variance minimum. La frontière efficiente obtenue est la suivante :



Ainsi, dans notre exemple, le portefeuille de Variance Minimum est le vecteur de pondérations suivant :

CAC 40	0%
STOXX 600	0%
EMTSCR Index	12%
EZCIEZCI Index	0%
HFRXGL Index	88%
Variance Cible	0,0002%
Performance Cible	0,0081%

Le portefeuille de variance minimum peut être facilement obtenu à partir des résultats précédents, en développant une fonction de recherche de la variance Minimum. Cette fonction est écrite ci-dessous (CF « Projet2.xlsm »- Module «mdlLibrary ») :

```

Function Get_PtfSigmaMin(vMatriceAllocationCible As Variant, vVecteurSigmaMu As Variant) As Variant

Dim i As Long, j As Integer, rang As Long
Dim dVarianceMin As Double, dPerf As Double
Dim vResultat() As Variant
    ReDim vResultat(1 To 1, 1 To UBound(vMatriceAllocationCible, 2) + 2)

'on part d'une variance arbitraire
dVarianceMin = 1000
    For i = LBound(vVecteurSigmaMu) To UBound(vVecteurSigmaMu)
        If vVecteurSigmaMu(i, 1) < dVarianceMin Then
            dVarianceMin = vVecteurSigmaMu(i, 1)
            dPerf = vVecteurSigmaMu(i, 2)
            rang = i
        End If
    Next i

'Ecriture du résultat dans la matrice
    For j = LBound(vMatriceAllocationCible, 2) To UBound(vMatriceAllocationCible, 2)
        vResultat(1, j) = vMatriceAllocationCible(rang, j)
    Next j
    vResultat(1, UBound(vMatriceAllocationCible, 2) + 1) = dVarianceMin
    vResultat(1, UBound(vMatriceAllocationCible, 2) + 2) = dPerf

Get_PtfSigmaMin = vResultat
End Function

```

Cette fonction est appelée via la procédure suivante (CF « Projet2.xlsm »- Module «mdlPrincipal ») :

```

Sub PtfSigmaMin()

Dim vMatriceAllocationCible As Variant, vVecteurSigmaMu As Variant
Dim vPtfSigmaMin As Variant

With Sheets("AnalysePtf")
    vMatriceAllocationCible = .Range(Cells(56, 4), Cells(56, 4).End(xlDown).End(xlToRight)).Value
    vVecteurSigmaMu = .Range("Variance_Cible").CurrentRegion.Value
    vPtfSigmaMin = Get_PtfSigmaMin(vMatriceAllocationCible, vVecteurSigmaMu)
'Ecriture du Résultat
    Range("PtfSigmaMin").Value = Application.Transpose(vPtfSigmaMin)
End With

End Sub

```

- Résolution du programme d'optimisation sous contrainte à l'aide d'une simulation de Monte Carlo.

A la différence du calcul itératif du solveur Excel, l'idée est ici de générer de manière aléatoire un grand nombre de pondérations est de chercher ensuite la stratégie qui génère le portefeuille de variance minimum. La fonction développée est la suivante (CF « Projet2.xlsm »- Module «mdlLibrary»). Notons que le générateur de nombre aléatoire utilisé est celui d'Excel. Ses capacités sont relativement limitées, puisqu'au bout d'un certain nombre de tirage (que nous appellerons « strates »), la probabilité de tirer un nombre identique augmente avec le temps.

```

Function Get_SimulationPonderation(NbIndices As Integer, NbTirage As Double) As Variant

Dim VecteurPonderation() As Double, MatriceResultat() As Double
ReDim VecteurPonderation(1, 1 To NbIndices)
ReDim MatriceResultat(1 To NbTirage, 1 To NbIndices)

Dim i As Long, j As Long, alea As Double, aleaTotal As Double

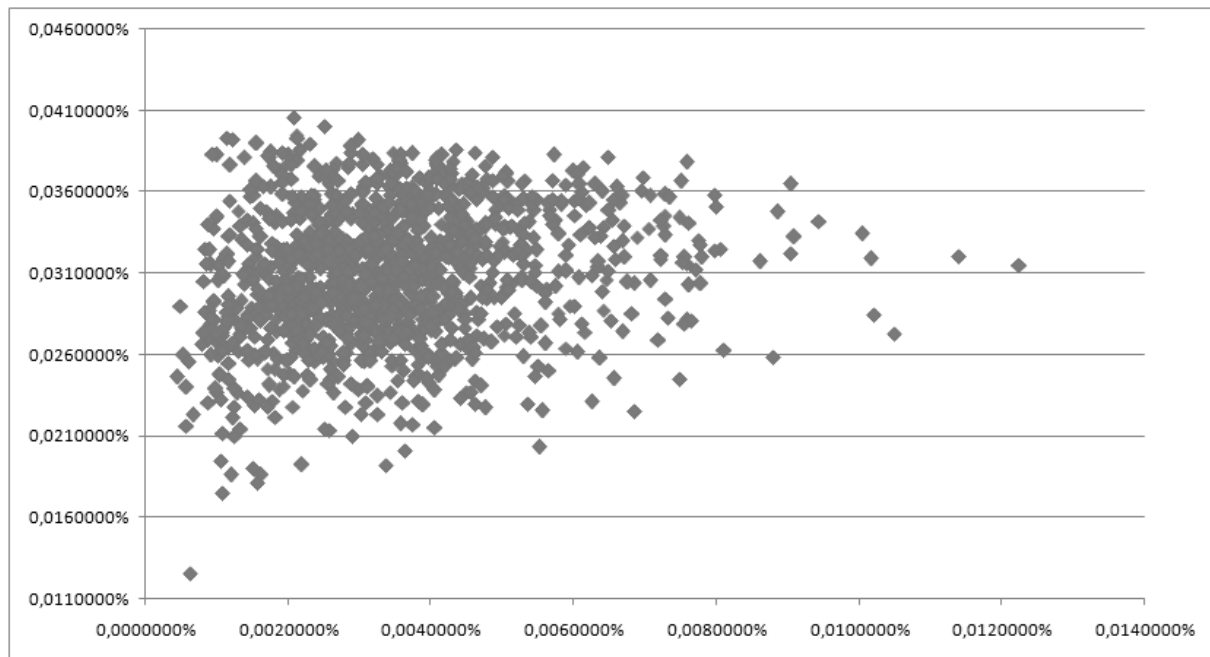
    For i = 1 To NbTirage
        aleaTotal = 0
        For j = 1 To NbIndices
            Randomize
            alea = Rnd()
            VecteurPonderation(1, j) = alea
            aleaTotal = aleaTotal + alea
        Next j

        For j = 1 To NbIndices
            VecteurPonderation(1, j) = VecteurPonderation(1, j) / aleaTotal
            MatriceResultat(i, j) = VecteurPonderation(1, j)
        Next j
    Next i

Get_SimulationPonderation = MatriceResultat
End Function

```

L'ensemble des tirages peut être représenté graphiquement sous forme d'un nuage de points, dont les extrémités dessinent la frontière efficiente. Ici nous avons effectué 5 000 tirages.



4 – L'annualisation:

Il existe un outil de comparaison simple de la performance et du risque. **L'annualisation** que l'on pourrait traduire par la projection de la performance et du risque sur une période de temps donnée. Nous considérons un processus markovien unidimensionnel observé à des instants régulièrement espacés de valeur $X(0)$, $X(1)$, ..., $X(n)$

indépendants, normalement et identiquement distribués (iid, hypothèse forte lorsque l'on cherche à modéliser l'évolution d'un cours boursier). Il est possible d'écrire le processus par ses deux premiers moments, son espérance μ , sa variance σ^2 et $\Delta X(t) \sim N(\mu, \sigma^2)$.

On pose :

$$U(t+1) = \frac{X(t+1) - X(t) - \mu}{\sqrt{\sigma^2}}$$

Donc $U(t+1)$ sont des variables centrées réduites et l'équation qui régit le mouvement X peut s'écrire :

$$X(t+1) - X(t) = \mu + \sigma U(t+1)$$

Ce processus est un processus autorégressif du premier ordre. Par ailleurs en écrivant $X(t) - X(0)$ comme la somme des $X(0)$ et de ses t accroissements entre 0 et t il vient :

$$X(t) - X(0) = \sum_{i=1}^t [X(i) - X(i-1)]$$

Les accroissements étant gaussiens et indépendants (en effet, les accroissements ne dépendent pas de μ et σ qui sont des constantes mais du temps - Cf propriété de la Loi Normale), on peut écrire :

$$E(X(t) - X(0)) = \mu t$$

$$VAR(X(t) - X(0)) = \sigma^2 t$$

$$X(t) - X(0) \sim N(\mu t, \sigma^2 t)$$

Le processus suivi par $X(t)$ peut donc s'écrire :

$$X(t+1) - X(t) = \mu + \sigma \sqrt{t} U$$

Par conséquent, en admettant qu'une valeur cotée suive un tel processus, ses rendements vont croître proportionnellement avec le temps. Son risque va quant à lui croître proportionnellement avec la racine carré du temps.

Attention : les accroissements suivent une loi normale mais la progression des prix est log Normale. Pour illustrer notre propos, il suffit de prendre un exemple simple : une variation de 5€ sur 100 € génère un rendement de 5%. En revanche, une variation d'un montant équivalent sur un capital initial de 10€ génère un rendement de 50%. De plus la valeur des actifs, notamment des actions ne peut pas descendre en dessous de 0, du fait, entre autre, de la responsabilité limitée des actionnaires. Par conséquent, le processus décrit plus haut ne prend pas en compte cet « effet capitalisation », la progression étant arithmétique. Il faut passer à une progression géométrique pour pouvoir modéliser correctement le comportement d'un cours boursier. Il vient alors :

$$\Delta X = X\mu + \sigma X\sqrt{t}U$$

$$\Leftrightarrow \frac{\Delta X}{X} = \mu t + \sigma \sqrt{t} U$$

Ce processus de diffusion bien connu est appelé mouvement brownien géométrique.

- Construction d'une fonction d'annualisation du risque et du rendement en VBA (Cf Projet2.xlsm – mdlLibrary).

```
Function Get_LissageMuSigma(MuSigma As Double, nbper As Integer, chx As Integer) As Double
'renvoi la performance (chx = 1 ) lissée , ou la variance lissée (chx = 2) pour un nbre de périodes données
  If chx = 1 Then
    Get_LissageMuSigma = (1 + MuSigma) ^ nbper - 1
  ElseIf chx = 2 Then
    Get_LissageMuSigma = MuSigma * Sqr(nbper)
  End If
End Function
```

Contrairement aux fonctions développées précédemment, cette fonction n'est pas appelée à l'aide d'une procédure mais est directement intégrée dans le tableur Excel (Cf. feuille AnalysePtf - cellule « C12 et c13 »). L'annualisation a pour principal intérêt de ramener des valeurs sur une même échelle de temps. Nous avons vu que celle-ci fait appel à des hypothèses fortes sur la distribution des rendements (iid).

	A	B	C	D	E
1	Analyse Indice				
2					
3	Input	Calcul Indice			
4					
5	Paramétrage :				
6	Pas de Temps (jour)	1			
7	Indice	CAC 40			
8					
9	Output				
10					
11	Indicateur de Rendement :		Lissage sur (j)	252	
12	Performance Moyenne (Arithmétique) :	0,0404%	=Get_LissageMuSigma(B12;\$D\$11;1)		
13	Performance Moyenne (Géométrique) :	0,0287%	7,507%		
14					
15	Indicateur de risque (à partir de la moyenne géométrique)				
16	Variance :	0,024%			
17	Ecart type :	1,533%	24,3406%		
18					

5 – Gestion indicielle – Gestion benchmarkée :

Pour rappel : un indice est un instrument statistique qui reflète l'évolution d'une place financière, d'une zone géographique, d'un secteur d'activité... Pour aider les investisseurs à suivre les variations de marché.

- Cette partie se déroule en trois étapes : la première est l'analyse de la **tracking error**, ou mesure de l'écart de la différence entre deux portefeuilles. La seconde concerne la **gestion benchmarkée**, c'est-à-dire que le gérant possède un portefeuille de référence, le benchmark, au regard duquel il est évalué. Enfin sera évoqué la

gestion indicielle : l'attitude du gérant vis-à-vis du benchmark est nulle, celui-ci doit répliquer strictement son indice de référence.

La tracking error se mesure comme l'écart type de la différence des rentabilités entre le fonds et le benchmark. Il peut en être fait deux utilisations : la tracking error ex ante qui mesure le risque pris par rapport à l'indice de référence. La tracking error ex post qui mesure le risque réalisé par rapport au benchmark.

La formule de calcul est la suivante :

$$T^2 = \sigma^2(Rp - RB) = \sigma^2B + \sigma^2P - 2\rho\sigma B\sigma P$$

$$T^2 = \sigma^2P + \sigma^2B (1 - 2C)$$

$$\text{avec : } \beta P = \frac{\sigma B \sigma P}{\sigma^2 B}$$

Par construction, on constate que T^2 dépend du risque du portefeuille, du risque du benchmark et du coefficient de corrélation entre les rentabilités du portefeuille et du benchmark. La formule de calcul servant à la construction de la tracking error permet d'aboutir à deux conclusions importantes :

- 1- La tracking error est une fonction décroissante monotone de la corrélation. Autrement dit lorsque ρ augmente l'évolution des rendements du portefeuille se rapprochent de ceux du benchmark.
- 2- La tracking error n'est pas monotone en la volatilité portefeuille et en celle du benchmark.

Si	βP	>1	<1
Alors		$T^2 \nearrow$	$T^2 \searrow$
Si	βP	$\leq \frac{\sigma^2 P}{\sigma^2 B}$	$> \frac{\sigma^2 P}{\sigma^2 B}$
Alors		$T^2 \nearrow$	$T^2 \searrow$

La gestion active benchmarkée a pour but de battre le benchmark choisi, sans cependant trop s'éloigner de l'allocation préconisée par se dernier. Par conséquent le gérant va effectuer des choix stratégiques (surpondérer une classe d'actif au détriment d'une autre), ou tactiques (sélectionner dans une même classe d'actifs un titre plutôt qu'un autre) tout en respectant la tracking error imposée. Alors que l'optimisation de Markowitz que l'on qualifie d'absolue (au sens où c'est le risque absolu qui est pris en compte), l'optimisation relative ne permet pas en général de dominer le Benchmark. En effet, le gérant de portefeuille possède une référence de gestion, qu'il doit suivre. Nous allons développer ici un programme d'optimisation sous contrainte de la tracking error (que nous appellerons optimisation relative). Il s'agit d'un programme de minimisation de la variance de la tracking error sous contrainte de la rentabilité anticipée.

- Notons :

\vec{b} Le vecteur des pondérations des actifs dans le benchmark.

$$\vec{b} = \begin{pmatrix} b1 \\ b2 \\ . \\ bn \end{pmatrix}$$

$\vec{x} = (\vec{w} - \vec{b})$: le vecteur des poids des différences entre les poids du portefeuille et ceux du benchmark.

$\overrightarrow{b'R}$: la rentabilité espérée du benchmark.

$\sigma^2 B = \vec{b} * \vec{V} * \vec{b}$: la variance de la rentabilité du benchmark.

$T = \sqrt{(\vec{w} - \vec{b})' \vec{V} (\vec{w} - \vec{b})}$ = la volatilité de la tracking error

Le programme d'optimisation sous contrainte devient alors :

$$\begin{cases} \text{Min } \overrightarrow{(w - b)'V} = \overrightarrow{(w - b)} \\ \text{sc : } \overrightarrow{(w - b)'R} = G \\ \overrightarrow{w'e} = 1 \\ \forall i \ w_i \geq 0 \end{cases}$$

Le terme G représente l'excès de rentabilité anticipée du fonds par rapport au benchmark.

La **gestion indicielle** s'est beaucoup développée car les investisseurs sont partis du constat que peu de gérants de portefeuille étaient en mesure de battre leur indice de référence à long terme. Par conséquent le marché s'est intéressé aux types de gestion qui suivaient rigoureusement l'indice de référence. De nombreuses études statistiques montrent que sur une période de 5 ans se sont les résultats des gestions indicielles qui sont les plus performantes. De plus elles permettent dans une certaine mesure d'encadrer le risque en ce sens que le seul pris est celui pris par le marché. Le risque lié aux décisions d'investissement (l'alpha) ne rentre plus en ligne de compte. Il existe différentes techniques de réplication des indices. Quelque soit la technique utilisée, l'objectif est de réduire le plus possible la tracking error. Nous allons aborder dans un premier temps les méthodes de construction des indices.

Les indices équipondérés s'obtiennent en calculant simplement la moyenne arithmétique des prix des actifs qui le compose.

$$I_t = \frac{\sum_{i=1}^n P_{i,t}}{nDt}$$

avec : P_i , test le prix de l'actif i à une date donnée t . Dt est le diviseur entier qui permet de calculer l'indice en pourcentage par rapport à une date de référence.

Les indices géométriques permettent de corriger le défaut principal des indices équipondérés, à savoir qu'ils s'obtiennent sur des variations relatives. Ainsi une variation de 1% d'un actif aura un effet identique sur le niveau de l'indice quelque soit la valeur unitaire du titre.

$$I_t = I_{t-1} \sqrt[n]{\prod_{i=1}^n \frac{P_{i,t}}{P_{i,t-1}}}$$

Les indices pondérés par capitalisation. L'idée est de créer un indice qui prenne en compte un nombre suffisant de valeurs afin d'être représentatif du marché dans son ensemble, mais sans être trop grand pour que tous les titres qui le compose puissent être rapidement achetés et vendus.

$$I_t = I_0 \frac{\sum_{i=1}^n P_{i,t} N_{i,t}}{\sum_{i=1}^n P_{i,0} N_{i,0}}$$

Avec $N_{i,t}$ le nombre de titres en circulation en date t .

Ces différents modes de calcul des indices étant présentées nous allons nous intéresser maintenant aux différentes techniques de réplcation des indices.

La réplcation parfaite consiste à créer un portefeuille dont la composition est en tout point identique à celle de l'indice. L'avantage principal est d'obtenir une tracking error minimum mais on peut vite rencontrer des problèmes de liquidité des actifs composant l'indice. Par ailleurs le gérant sera toujours confronté à la gestion de ses coûts de transaction. La réplcation par stratification permet de décomposer l'indice en regroupant les titres ayant des caractéristiques similaires, comme par exemple la zone géographique, le secteur d'activité... L'avantage principal est bien sur la facilité d'implémentation, qui se fait au détriment d'une perte de précision. Par ailleurs il est impossible de mesurer la tracking error ex ante sur ce type de réplcation. La réplcation synthétique consiste à décomposer à réplquer la valeur de l'indice à l'aide d'instruments dérivés. La valeur des instruments dérivés découlent de la valeur des actifs sur lequel ils portent. Théoriquement, le prix à terme F doit être égal à :

$$F = S_0(1 + r)^{T-t} - D$$

avec r le taux sans risque et D le montant des dividendes versés jusqu'à la fin du contrat.

Si le terme initial se monte à 100 et qu'à l'échéance le spot est de 80 la perte est de 20. Mais le placement des liquidités à terme (liées à la marge) au taux sans risque permet de compenser la perte de richesse. Cela génère comme principal avantage une réplcation théoriquement parfaite, mais avec la contrepartie que les indices soient des sous jacents des instruments dérivés. De plus les dérivés ont une durée de vie limitée, il faut donc faire un roll de produit ce qui implique là aussi un coût pour le gérant.

- Développement d'une fonction de calcul de la tracking error (Cf Projet2.xlsm – mdlLibrary) :

```

Function Get_TrackingError(vmatricePoids As Variant, vPriceMatrice As Variant, vVecteurBenchmark As Variant) As Variant
'T^2 = sigma^2(Rp - Rb)
Dim i As Long, j As Integer
Dim vJ1 As Double, vJplus1 As Double
Dim vMatricePerformancePtf()
    ReDim vMatricePerformancePtf(1 To UBound(vPriceMatrice), 1 To 1)
Dim vVecteurRpMoinsRb()
    ReDim vVecteurRpMoinsRb(1 To UBound(vPriceMatrice) - 1, 1 To 1)

    'Rp
    For j = LBound(vPriceMatrice) + 1 To UBound(vPriceMatrice)
        For i = LBound(vPriceMatrice, 2) To UBound(vPriceMatrice, 2)
            vJplus1 = vPriceMatrice(j, i)
            vJ1 = vPriceMatrice(j - 1, i)
            vMatricePerformancePtf(j, 1) = vMatricePerformancePtf(j, 1) + ((vJplus1 / vJ1 - 1) * vmatricePoids(j, i))
        Next i
    Next j
    'Rp-Rb
    vVecteurRpMoinsRb(j - 1, 1) = vMatricePerformancePtf(j, 1) - vVecteurBenchmark(j, 1)
Next j

Get_TrackingError = Get_VarianceRdt(vVecteurRpMoinsRb) ^ (1 / 2)
End Function

```

Cette fonction est appelée par la procédure suivante (CF Projet2.xlsm – Module Principal) :

```

Sub Calcul_TrackingError()

    Dim vmatricePoids As Variant, vMatricePerf As Variant, vVecteurPerfBench As Variant
    Dim vTrackingError As Variant

    With Sheets("Ponds")
        vmatricePoids = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
    End With
    With Sheets("Data")
        vMatricePerf = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
    End With
    With Sheets("Bench")
        vVecteurPerfBench = .Range(.Cells(2, 2), .Cells(3, 2).End(xlDown)).Value
    End With
    vTrackingError = Get_TrackingError(vmatricePoids, vMatricePerf, vVecteurPerfBench)
    With Sheets("AnalysePtf")
        .Range("TrackingError").Cells(1, 2).Value = vTrackingError
    End With

End Sub

```

- Développement d'une fonction de calcul pour la construction d'indices équipondérés et géométriques (Cf Projet2.xlsm – mdlLibrary)

```

Function Get_ConstructionIndex(vMatriceData As Variant, vMatricenbTitresBench As Variant, methode As Integer) As Variant
    Dim i As Long, j As Long, k As Long
    Dim vVecteurIndice()
    ReDim vVecteurIndice(LBound(vMatriceData) To UBound(vMatriceData), 1 To 1)
    Select Case True
        Case methode = 1
            'construction équilibrée -> It = (Somme Pi,t / n)
            For i = LBound(vMatriceData) To UBound(vMatriceData)
                For j = LBound(vMatriceData, 2) To UBound(vMatriceData, 2)
                    vVecteurIndice(i, 1) = vVecteurIndice(i, 1) + vMatriceData(i, j)
                Next j
                vVecteurIndice(i, 1) = vVecteurIndice(i, 1) / UBound(vMatriceData, 2)
            Next i
        Case methode = 2
            'construction géométrique It = (Racine(Produit Pi,t/Pi,t))
            vVecteurIndice(1, 1) = 1
            For i = LBound(vMatriceData) + 1 To UBound(vMatriceData)
                vVecteurIndice(i, 1) = 1
                For j = LBound(vMatriceData, 2) To UBound(vMatriceData, 2)
                    vVecteurIndice(i, 1) = vVecteurIndice(i, 1) * (vMatriceData(i, j) / vMatriceData(i - 1, j))
                Next j
                vVecteurIndice(i, 1) = vVecteurIndice(i - 1, 1) * vVecteurIndice(i, 1) ^ (1 / UBound(vMatriceData, 2))
            Next i
        Case methode = 3
            'Exercice : construction méthode par capitalisation
            '[.... Code VBA.... ]
    End Select

    Get_ConstructionIndex = vVecteurIndice
End Function

```

Cette fonction est appelée par la procédure suivante (CF Projet2.xlsm – Module Principal) :

```

Sub Construction_Index()
    Dim vMatricenbTitresBench As Variant, vMatriceData As Variant, vVeceturIndice As Variant
    Dim j As Long

    With Sheets("Data")
        vMatriceData = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
    End With
    With Sheets("nbTitresBench")
        vMatricenbTitresBench = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
    End With
    vVeceturIndice = Get_ConstructionIndex(vMatriceData, vMatricenbTitresBench, 2)
    With Sheets("AnalysePtf")
        For j = LBound(vVeceturIndice) To UBound(vVeceturIndice)
            .Range("ConstructionIndice").Cells(j, 1).Value = vVeceturIndice(j, 1)
        Next j
    End With
End Sub

```

6 - Méthodes de sélection des actions :

Les portefeuilles de Markowitz nécessitent de connaître deux inputs : la moyenne et la variance. Ici nous allons nous intéresser aux méthodes de détermination de la rentabilité espérée, fondement principal de l'évaluation des actions. L'hypothèse des marchés efficients démontrée par Samuelson (1965-1978) part du principe que dans un tel univers, toute l'information est incorporée dans le prix de l'action au moment présent. Par conséquent les variations des prix ne peuvent pas être anticipées avec justesse. Cette notion d'efficacité informationnelle et très proche de celle des anticipations rationnelles introduites par Muth (1961), pouvant être traduite par le fait que la prévision d'un prix futur aléatoire est égale à l'espérance conditionnelle de ce prix. Par conséquent, la meilleure approximation compte tenu de l'information disponible Ω_t à l'instant considéré est :

$$P_{t+1} = E(P_{t+1}|\Omega_t) + \varepsilon_t \text{ avec } E(\varepsilon_{t+1}) = 0 \text{ } \varepsilon_t \text{ est l'erreur de prévision}$$

D'où en termes de rentabilité :

$$R_{t+1} = E(R_{t+1}|\Omega_t) + \varepsilon_{t+1} \text{ avec } R_{t+1} = \frac{P_{t+1}}{P_t} - 1 \text{ ou } R_{t+1} = \ln\left(\frac{P_{t+1}}{P_t}\right)$$

En supposant que les rentabilités espérées sont constantes et en utilisant l'hypothèse des anticipations rationnelles (i.e, les agents sont capables de tirer parti de toute l'information disponible pour former leurs anticipations de sorte qu'en moyenne ils ne se trompent pas) il vient en termes de rentabilités :

$$R_{t+1} = k + \varepsilon_{t+1}$$

$$\text{où } \ln(P_{t+1}/P_t) \approx k + \varepsilon_{t+1}$$

$$\text{et } k = E(R_{t+1}|\Omega_t) = \text{Cte}$$

$$\ln(P_{t+1}) \approx k + \ln(P_t) + \varepsilon_{t+1}$$

L'équation précédente est l'équation représentative d'une marche aléatoire avec dérive. Il s'agit d'un processus stochastique qui à la propriété martingale. Notons que la propriété martingale d'un processus stochastique est en fait moins restrictive que celle de la marche aléatoire en ce sens qu'il n'y a pas besoin d'exiger que les ε_t soient iid comme le nécessite une marche aléatoire de type mouvement brownien. Seul l'absence d'auto corrélation est postulée et certaines formes de dépendances sont permises comme la dépendance temporelle de la variance. Ces résultats vont nous permettre d'introduire l'un des modèles d'évaluation d'action les plus connus, celui d'évaluation par actualisation des dividendes futurs. En pratique les actions sont évaluées par l'actualisation de leur cash flow futurs anticipés. Le taux d'actualisation est déterminé à Partir du CAPM.

Analyse du Dividend Yield Model (DDM) (CF démonstration en annexe).

On part de la martingale suivante :

$$R_{t+1} = E(R_{t+1}|\Omega_t) + \varepsilon_{t+1}$$

Que l'on peut écrire aussi, en termes de prix actualisés intégrant les flux de dividendes réinvestis (si les dividendes ne sont pas réinvestis, les prix ne peuvent pas former une martingale) :

$$P_{t+1} = E(P_{t+1}|\Omega_t) + \varepsilon_t$$

Par conséquent, on peut écrire :

$$P_t = E\left(\frac{P_{t+1} + D_{t+1}}{1+k} \middle| \Omega_t\right)$$

Et pour $t + 2$ il vient :

$$P_t = E_t\left(\frac{E_{t+1}\left(\frac{P_{t+2} + D_{t+2}}{1+k}\right) + D_{t+1}}{1+k}\right) = E_t\left(\frac{D_{t+1}}{1+k} + \frac{P_{t+2} + D_{t+2}}{(1+k)^2}\right)$$

Et pour $t + 3$ il vient :

$$P_t = E_t \left(\frac{E_{t+1} \left(\frac{E_{t+2} \left(\frac{P_{t+3} + D_{t+3}}{1+k} \right) + D_{t+1}}{1+k} \right) + D_{t+1}}{1+k} \right) = E_t \left(\frac{D_{t+1}}{1+k} + \frac{D_{t+2}}{1+k} + \frac{P_{t+3} + D_{t+3}}{(1+k)^3} \right)$$

Par conséquent la formule à laquelle on aboutit en prenant n périodes est la suivante :

$$P_t = E_t \left[\sum_{i=1}^n \frac{D_{t+i}}{(1+k)^i} \right] + E_t \left[\frac{P_{t+n}}{(1+k)^n} \right]$$

On pose aussi l'hypothèse que le cours de l'action croît à un taux inférieur à k, c'est-à-dire :

$$\lim_{n \rightarrow \infty} E_t \left[\frac{P_{t+n}}{(1+k)^n} \right] = 0$$

Si cette hypothèse n'est pas respectée nous sommes en présence d'une bulle spéculative sur l'action P et l'équation aurait une infinité de solution. Donc lorsque n tend vers l'infini il vient :

$$P_t = E_t \left[\sum_{i=1}^{\infty} \frac{D_{t+i}}{(1+k)^i} \right] = \sum_{i=1}^{\infty} \frac{E_t[D_{t+i}]}{(1+k)^i}$$

Cette formule est difficilement implémentable en sens qu'il faut connaître la progression des dividendes à l'infini. Par conséquent, il vaut mieux implémenter un modèle à plusieurs périodes.

Modèle 1 : modèle d'actualisation des dividendes à taux de croissance constant. On part de l'hypothèse que le dividende croît au taux constant g :

$$D_t = D_0(1+g)^t$$

On obtient alors en utilisant l'expression de la somme infinie des termes d'une série géométrique:

$$P_0 = D_0 \sum_{i=1}^{\infty} \frac{(1+g)^i}{(1+k)^i} = \frac{D_0(1+g)}{k-g} = \frac{D_1}{k-g}$$

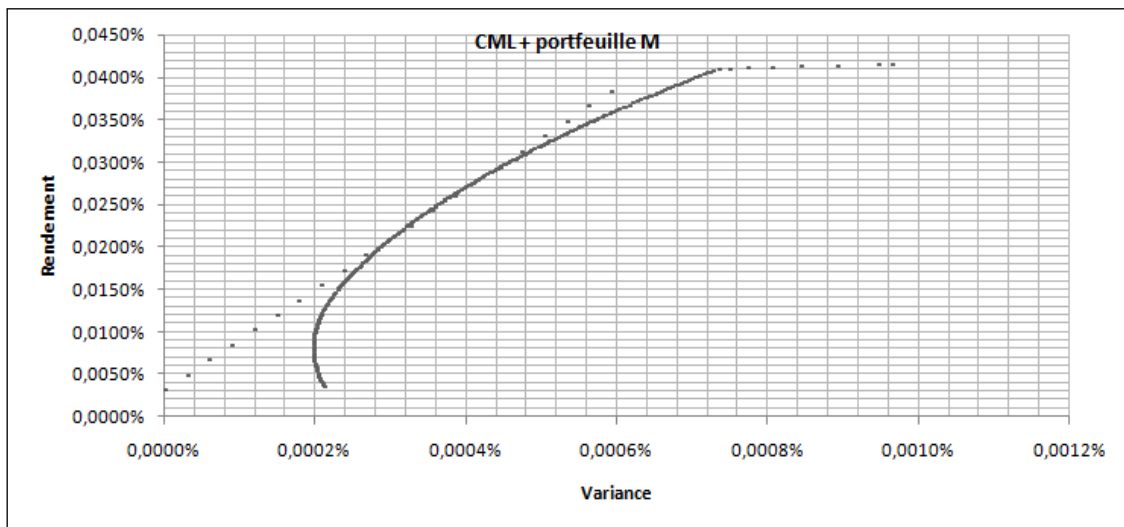
Notons qu'il faut que $k > g$ pour être dans un univers à l'équilibre. Le DDM à taux de croissance constant présuppose que l'entreprise suive une politique de distribution stable dans le temps et que son taux de distribution soit constant ; et l'entreprise dégage une rentabilité constante de ses investissements.

Modèle 2 : modèle d'actualisation à plusieurs phases de croissances, analyse à deux périodes. La première période va dépendre de la capacité des prévisions de l'analyste financier, tandis que la deuxième période fait intervenir le taux de croissance moyen estimé sur le long terme.

$$D_t = \begin{cases} D_0(1+g)^t, & \text{si } t \leq n \\ D_n(1+g)^t = D_0(1+g_0)^n(1+g_1)^{t-n}, & \text{si } t > n \end{cases}$$

$$P_0 = D_1 \left[\frac{1 - \left(\frac{1+g_0}{1+k}\right)^n}{k - g_0} \right] + \left[\frac{\frac{D_0(1+g_0)^n}{(1+g_1)}}{(k - g_1)} \right] \left[\frac{1}{(1+k)^n} \right]$$

Le taux d'actualisation utilisé dans le DDM peut provenir du CAPM, de l'APT ou d'un autre modèle multifactoriel. Le CAPM (Capital Asset Pricing Model) est le modèle d'équilibre des actifs financiers : ce modèle établit que les rentabilités des actifs financiers dépendent des covariances avec le portefeuille de marché. Rappelons que la CML est la droite qui passe par le taux sans risque et le portefeuille de marché.



Remarque : à portefeuille de variance égale, celui constitué de la combinaison de l'actif sans risque et du portefeuille de marché générera un rendement supérieur à celui constitué uniquement du portefeuille de marché. Au point de tangence, la part de l'actif sans risque est nulle dans le portefeuille est nulle. Sur l'ordonnée, elle est égale à 100%. Au delà du point de tangence, la part de l'actif sans risque est négative. Considérons un titre risque i qui n'est pas sur la CML ni sur la FE (frontière efficiente) et considérons un portefeuille P constitué du portefeuille de marché et de l'actif i . Il vient :

$$E[R_i] = R_f + \frac{\text{Cov}(R_i, R_m)}{\sigma^2 R_m} (E[R_M] - R_f)$$

$$E[R_i] = R_f + \beta_i (R_m - R_f)$$

La rentabilité d'un titre se mesure par l'addition du taux sans risque plus la prime de risque. La prime de risque est le prix de marché du risque fois la quantité de risque :

$$\left(\frac{E(R_m - R_f)}{\sigma^2} \right) \beta_i \sigma^2 [R_m]$$

Le CAPM possède une forme générale appelée l'APT (Asset Pricing Theory). On suppose que les investisseurs forment des anticipations homogènes et que la rentabilité de n'importe quel titre risqué R_i dépend linéairement de n facteurs.

$$R_i = a_i + \sum_{k=1}^n b_{ik} F_k + \varepsilon_i, \forall i = 1, \dots, n$$

En soustrayant à cette expression son espérance on obtient :

$$R_i = E[R_i] + \sum_{k=1}^n b_{ik} (F_k - E[F_k]) + \varepsilon_i$$

avec : $E[R_i]$ est l'espérance de rentabilité du titre i , b_{ik} est la sensibilité du titre i au k ème facteur risqué k , $F_k - E[F_k]$ la variation anticipée du k ème facteur d'espérance nulle, ε_i le risque diversifiable

- Développement d'une fonction de calcul du CAPM en VBA.

L'appel de la fonction se fera directement via le tableur. (Cf Projet2.xlsm – mdlLibrary). On part du principe que les variables composant le modèle sont connues.

```
Function Get_CAPM(Ro As Range, Rm As Range, Beta As Range) As Double
    'ER(i) = Ro + Beta * (Rm - Ro)
    Get_CAPM = Ro + Beta * (Rm - Ro)
End Function
```

L'appel de la fonction se fait en renseignant la fonction dans une des cellules du tableur (Cf feuille SelectionActions) :

	A	B	C	D
1	CAPM			
2				
3	Input			
4	Ro	2%		
5	Rm	5%		
6	Beta	1,2		
7				
8	Output			
9	ER(i)	=Get_CAPM(B4;B5;B6)		

- Développement d'une fonction de calcul du DDM (Cf Projet2.xlsm – mdlLibrary).

```
Function Get_DDM(Eo As Range, n As Range, d As Range, Ro As Range, g As Range, Pr As Range) As Double
    Dim i As Long
    Dim FirstMember As Double, SecondMember As Double
    '1st member
    For i = 1 To n
        FirstMember = FirstMember + (d * (1 + g) ^ i) / (1 + Ro + Pr) ^ i
    Next i
    FirstMember = FirstMember * Eo
    '2nd member
    SecondMember = Eo * ((1 + g) ^ n / (1 + Ro + Pr) ^ n) * (1 / (Ro + Pr))
    Get_DDM = FirstMember + SecondMember
End Function
```

L'appel de la fonction se fait en renseignant la fonction dans une des cellules du tableur (Cf. feuille "SelectionActions") :

11	DDM		
12			
13	Input (Données de calibration utilisées Prigent/Bertrand)		
14	Indice actions suivi	S&P	
15	Eo (bénéfice estimé sur indice actions)	39,94	
16	n (nombre d'années)	10	
17	d (taux de distribution des bénéfices)	41%	
18	Ro (sur n années)	6,46%	
19	g (taux de croissance anticipé des bénéfices)	12,71%	
20	Pr (Rm-Ro = prime de risque)	3,05%	
21			
22			
23	Output		
24	Po	=Get_DDM(B15;B16;B17;B18;B19;B20) ←	

L'avantage du vba (et de la programmation en général) est de pouvoir profiter pleinement de la puissance de calcul de sa machine. Par conséquent rien n'empêche d'envisager un ensemble de scénarios déterministes dans lesquels un ou plusieurs facteurs de risque seraient choqués. On pourrait, à l'aide d'une fonction appropriée calculer rapidement un set de valeur d'action. Nous proposons ici deux fonctions permettant de mettre un place ce type de simulation. Dans la première fonction ce sont les facteurs de risque d et Ro qui sont simulés. Dans la deuxième fonction, ce sont les facteurs de risque g et Pr qui sont choqués.(Cf Projet2.xlsm – mdlLibrary).

```

Function Get_SimulationDDM_d_Ro(Eo, n, d, Ro, g, Pr, Param1, Param2) As Variant
    Dim i As Double, j As Double
    Dim ligne As Long, colonne As Long
    Dim vMatriceResultat() As Double
    ReDim vMatriceResultat(1 To 12, 1 To 12)

    ligne = 1
    For i = Param1 - 5 / 100 To Param1 + 5 / 100 Step 1 / 100
        vMatriceResultat(ligne + 1, 1) = i
        colonne = 1
        For j = Param2 - 5 / 100 To Param2 + 5 / 100 Step 1 / 100
            vMatriceResultat(1, colonne + 1) = j
            vMatriceResultat(ligne + 1, colonne + 1) = Get_DDM_V(Eo, n, i, j, g, Pr)
            colonne = colonne + 1
        Next j
        ligne = ligne + 1
    Next i

    Get_SimulationDDM_d_Ro = vMatriceResultat
End Function

```

```

Function Get_SimulationDDM_g_Pr(Eo, n, d, Ro, g, Pr, Param1, Param2) As Variant
    Dim i As Double, j As Double
    Dim ligne As Long, colonne As Long
    Dim vMatriceResultat() As Double
    ReDim vMatriceResultat(1 To 12, 1 To 12)

    ligne = 1
    For i = Param1 - 5 / 100 To Param1 + 6 / 100 Step 1 / 100
        vMatriceResultat(ligne + 1, 1) = i
        colonne = 1
        For j = Param2 - 5 / 100 To Param2 + 6 / 100 Step 1 / 100
            vMatriceResultat(1, colonne + 1) = j
            vMatriceResultat(ligne + 1, colonne + 1) = Get_DDM_V(Eo, n, d, Ro, i, j)
            colonne = colonne + 1
        Next j
        ligne = ligne + 1
    Next i

    Get_SimulationDDM_g_Pr = vMatriceResultat
End Function

```

Notons que comme les deux fonctions précédentes, celle-ci est appelée directement via le tableur. Par ailleurs, elle renvoie comme résultat une matrice de données, ce qui implique qu'après l'avoir renseignée dans une feuille de calcul, il faut appuyer sur **Ctrl + Shift + Entrée** pour avoir le calcul approprié (Cf feuille *SelectionActions*).

[illegible]

7 - Mesure et attribution de performance : l'attribution interne

L'attribution a pour but d'expliquer à posteriori la surperformance d'un fonds d'actifs financiers par rapport à l'objectif de gestion : le benchmark. Une fois le benchmark choisi, le processus de gestion peut-être schématiquement décomposé en deux parties : l'allocation d'actifs qui consiste à décider de la sur ou sous pondération des grandes classes d'actifs par rapport à leur poids dans le benchmark ; la sélection de titres, au niveau de laquelle le gérant sélectionne les titres de la classe d'actifs qui lui permettront de battre l'indice de la classe. Le modèle ici présenté est celui de Brinson/Grap qui décompose l'attribution de performance en trois effets : l'effet allocation, l'effet sélection, et l'effet interaction. Les notation sont les suivantes :

P est le portefeuille

b est le benchmark

U est l'univers des titres $l = 1, \dots, m = \#(U)$

$\{U_1, \dots, U_n\}$ est une partition de l'ensemble U est est l'ensemble des n classes d'actifs.

Z_p est la rentabilité de l'actif l

R_l est la rentabilité moyenne de l'actif l

$\sigma_{k,l}$ est la covariance entre les rentabilités des actifs k et l

$w_{p,l}$ est le poids de l'actif l dans le portefeuille géré

$w_{pi} = \sum_{l \in U_i} w_{p,l}$ est le poids de la classe d'actifs l dans la classe d'actifs i du portefeuille et du benchmark

$\bar{w}_{pi} = \frac{w_{p,l}}{\sum_{l \in U_i} w_{p,l}}$ est poids de l'actif l dans la classe d'actifs i du portefeuille et du benchmark

$Z_{pi} = \sum_{l \in U_i} \bar{w}_{pi} \cdot Z_l$ est la rentabilité de la classe d'actifs i du portefeuille

$R_{pi} = \sum_{l \in U_i} \bar{w}_{pi} \cdot R_l$ est la rentabilité espérée de la classe d'actifs i du portefeuille.

NB : ces notations sont les mêmes dans le cadre de l'analyse du benchmark.

La surperformance globale du portefeuille contre le benchmark s'écrit alors :

$$S = R_p - R_b = \sum_{i=1}^n (w_{pi} R_{pi} - w_{bi} R_{bi}) = \sum_{l=1}^n (w_{p,l} - w_{b,l}) R_l$$

Cette surperformance globale se décompose en ses principales sources de gestion à savoir l'effet allocation et l'effet selection. L'effet allocation peut s'écrire :

$$AA_i = (w_{p,i} - w_{b,i})(R_{bi} - R_{bi})$$

L'effet selection de titres :

$$SE_i = w_{bi}(R_{pi} - R_{bi})$$

$(R_{pi} - R_{bi})$: Lorsqu'il est inférieur le gérant a effectué une pondération des titres différente de celle du benchmark.

Le terme d'interaction :

$$I_i = (w_{p,i} - w_{b,i})(R_{pi} - R_{bi})$$

La superperformance du portefeuille par rapport au bench s'écrit

$$S = R_p - R_b = \sum_{i=1}^n (w_{pi}R_{pi} - w_{bi}R_{bi}) = \sum_{i=1}^n (AA_i + SE_i + I_i)$$

- Construction du modèle de Brinson/Grap en VBA. Construction des trois fonctions « Allocation » / « Selection » / « interaction » (CF Projet2.xlsm – Module Library).

```
Function get_EffetAllocation(vVecteurPerfPtf, vVecteurPerfBench, vVecteurPondPtf, vVecteurPondBench) As Double
    Dim i As Long
    Dim Resultat As Double, dPerfBenchMoy As Double
    For i = LBound(vVecteurPerfBench, 1) To UBound(vVecteurPerfBench, 1)
        dPerfBenchMoy = dPerfBenchMoy + vVecteurPondBench(1, i) * vVecteurPerfBench(i, 1)
    Next i
    For i = LBound(vVecteurPerfPtf, 1) To UBound(vVecteurPerfPtf, 1)
        Resultat = Resultat + (vVecteurPondPtf(1, i) - vVecteurPondBench(1, i)) * (vVecteurPerfBench(i, 1) - dPerfBenchMoy)
    Next i
    get_EffetAllocation = Resultat
End Function
```

```
Function get_EffetSelection(vVecteurPerfPtf, vVecteurPerfBench, vVecteurPondPtf, vVecteurPondBench) As Double
    Dim i As Long
    Dim Resultat As Double
    For i = LBound(vVecteurPerfBench, 1) To UBound(vVecteurPerfBench, 1)
        Resultat = Resultat + vVecteurPondBench(1, i) * (vVecteurPerfPtf(i, 1) - vVecteurPerfBench(i, 1))
    Next i
    get_EffetSelection = Resultat
End Function
```

```
Function get_EffetInteraction(vVecteurPerfPtf, vVecteurPerfBench, vVecteurPondPtf, vVecteurPondBench)
    Dim i As Long
    Dim Resultat As Double
    For i = LBound(vVecteurPerfPtf, 1) To UBound(vVecteurPerfPtf, 1)
        Resultat = Resultat + (vVecteurPondPtf(1, i) - vVecteurPondBench(1, i)) * (vVecteurPerfPtf(i, 1) - vVecteurPerfBench(i, 1))
    Next i
    get_EffetInteraction = Resultat
End Function
```

La procédure suivante permet d'appeler les fonctions de calcul d'attribution de performance :

```

Sub AttributionPerformance()
'Modèle de Brinson/Grap
Dim vMatricePondBench As Variant, vMatriceDataPtf As Variant, vMatriceDataBench As Variant, vMatricePondPtf As Variant, vMatricePerfBench As Variant
Dim vVecteurDate As Variant, vVecteurPondBench As Variant, vVecteurPondPtf As Variant, vVecteurPerfPtf As Variant, vVecteurPerfBench As Variant
Dim dSurperf As Double, dEffetAllocation As Double, dEffetSelection As Double, dEffetInteraction As Double

With Sheets("PondBench")
vMatricePondBench = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
vVecteurDate = .Range(.Cells(2, 1), .Cells(2, 1).End(xlDown)).Value
End With
With Sheets("Data")
vMatriceDataPtf = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
End With
With Sheets("Ponds")
vMatricePondPtf = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
End With
With Sheets("DataBench")
vMatriceDataBench = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown).End(xlToRight)).Value
End With

'Calcul des rentabilités espérées du portefeuille et du Benchmark
vVecteurPerfPtf = Get_VecteurPerf(vMatriceDataPtf, 2)
vVecteurPerfBench = Get_VecteurPerf(vMatriceDataBench, 2)
'Recherche Pondérations Bench & portefeuille, par défaut dernière date connue
vVecteurPondBench = Get_LastPond(vMatricePondBench)
vVecteurPondPtf = Get_LastPond(vMatricePondPtf)
'*Calcul de la superperformance globale Ptf vs Bench
dSurperf = Get_SurPerf(vVecteurPerfPtf, vVecteurPerfBench, vVecteurPondPtf, vVecteurPondBench)
'*Calcul de l'effet allocation
dEffetAllocation = get_EffetAllocation(vVecteurPerfPtf, vVecteurPerfBench, vVecteurPondPtf, vVecteurPondBench)
'*Calcul de l'effet selection
dEffetSelection = get_EffetSelection(vVecteurPerfPtf, vVecteurPerfBench, vVecteurPondPtf, vVecteurPondBench)
'*Calcul du terme d'interaction
dEffetInteraction = get_EffetInteraction(vVecteurPerfPtf, vVecteurPerfBench, vVecteurPondPtf, vVecteurPondBench)

a = dEffetInteraction + dEffetSelection + dEffetAllocation

With Sheets("AttribPerf")
.Range("ResultatsBrinson").Cells(1, 2).Value = dSurperf
.Range("ResultatsBrinson").Cells(2, 2).Value = dEffetAllocation
.Range("ResultatsBrinson").Cells(3, 2).Value = dEffetSelection
.Range("ResultatsBrinson").Cells(4, 2).Value = dEffetInteraction
End With

End Sub

```

8- Autres indicateurs de mesure du risque : le Value at Risk (VaR).

Un portefeuille de marché contenant différents titres est affecté de nombreuses sources de risque : (taux, spread, volatilité...). Dans un cas où le nombre de sous-jacent existant en portefeuille est élevé, le nombre de facteur de risque existant l'est aussi ce qui rend l'analyse du risque associé à chacun d'entre eux plus complexe. La Value at Risk a l'avantage (et l'inconvénient) de quantifier le risque existant dans un portefeuille à l'aide d'un seul indicateur directement interprétable. Il en ressort que cet indicateur est plus utilisé comme un outil de calcul de contrainte réglementaire que comme un outil de gestion. Par définition, la VaR à un horizon de temps k et au seuil de probabilité p est le plus petit nombre $VaR(k,p)$ tel que :

$$Proba(L_k \leq VaR(k,p)) = p$$

La VaR est donc le p quantile de la perte L_k . La formule peut aussi être interprétée comme le calcul du montant de la perte sur une période de durée k à venir qui sera inférieur ou égal à la VaR avec une probabilité k . La VaR peut donc être interprétée comme une perte maximum sur un horizon de temps donnée avec un seuil de confiance donné. En cas de distribution gaussienne, l'expression analytique de la VaR est la suivante :

$$VaR(k,p) = \alpha_p * \sigma_k + \mu_k$$

Lorsque k est petit, μ_k est négligé.

Exemple : Soit un actif d'une valeur de 1 K€. Sur une période de 10 jours la rentabilité de cet actif est distribuée normalement. La moyenne des rendements est de 0.4%, et l'écart-type est de 5%. Avec un interval de confiance de 95% la perte maximale probable s'obtient en faisant le calcul suivant : $VaR(k,p) = VaR(10j, 95\%) = -1.645 * 5\% + 0.4\% = -7.825\%$. Il existe deux grandes familles de méthodes d'estimation de la VaR : les méthodes full valuation et les méthodes partial valuation. En full valuation, les facteurs de risque composant le modèle de valorisation sont choqués et le prix est systématiquement recalculé. En partial valuation, il conviendra de procéder à une approximation linéaire ou quadratique de la variation de valeur du portefeuille à l'aide de ses facteurs de risque. Pour ces deux familles de méthodes on pourra appliquer soit une simulation historique, soit une simulation paramétrique. Concernant la méthode historique, cette méthode repose sur l'estimation du p-quantile réel de la distribution des rendements de l'actif (ou de ses facteurs de risque) à partir d'un historique de données. Ce résultat peut s'obtenir simplement en utilisant la fonction CENTILE() disponible dans la bibliothèque de fonction d'Excel. Pour un titre ou un portefeuille de titre, il est aussi possible de décomposer la performance à l'aide des différentes valeurs le constituant (pour un portefeuille) ou des différents facteurs de risque le constituant (pour un titre). Par exemple pour le cas d'un portefeuille :

$$R_x(j) = \sum_{i=1}^M x_i R_i(j)$$

avec : j = jour j , x_i la proportion d'actif i et R_i la rentabilité i .

Note : Comme mentionné plus haut, on peut partir du principe que le portefeuille est un actif, et que l'ensemble des titres qui le compose sont les facteurs de risque. Pour chaque classe d'actif (action, obligation, option...) l'identification des facteurs de risque peut se faire à partir à l'aide du modèle de pricing sous-jacent. Par exemple, si le portefeuille est constitué (entre autre) d'obligations à taux variable, les facteurs de risque à intégrer seront : la courbe des taux de référence servant au calcul du coupon, la courbe des taux d'actualisation servant à déterminer les taux d'actualisation pour chaque maturité. Pour chaque valeur historique du facteur de risque une nouvelle valeur de l'actif est déterminée, et pour chaque actif revalorisé, une nouvelle valeur du portefeuille est calculée. Il ne reste plus qu'à calculer les rendements du portefeuille et déterminer la VaR correspondant à l'intervalle de confiance voulu. Une des méthodes paramétriques classique est l'évaluation partielle avec la méthode delta normale (Risk Metrics). Dans cette méthode, les constituants du portefeuille sont décomposés en flux élémentaires, chacun étant sensible à un seul facteur de risque. Une première étape consiste à calculer la DEaR (Daily Earning at Risk) pour une position i :

$$DEaR_i = V_i * S_i * X_i$$

avec : V_i la valeur de marché de la position, S_i la sensibilité de la position face à une variation unitaire du facteur de risque, X_i la grandeur du choc défavorable.

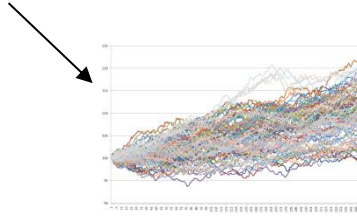
$$S_i = \frac{1}{V_i} * \frac{dV_i}{dY_i}$$

Au niveau du portefeuille il vient :

$$DEaR_p = (D * \Gamma * D')^{\frac{1}{2}}$$

$$VaR = DEaR_p * (NBJ)^{\frac{1}{2}}$$

Une méthode paramétrique en full valuation consiste à spécifier un processus de diffusion stochastique pour un facteur de risque et à générer un grand nombre de trajectoires possibles que l'on réintègrera dans le modèle de valorisation afin de calculer la VaR. Par exemple, pour une obligation à taux fixe, le facteur de risque à simuler est le taux d'actualisation de chaque tombée de coupon. Par conséquent, il faudra choquer à l'aide d'une simulation de Monte-Carlo ces facteurs de risques.

$$V_0 = \sum_{i=1}^n \frac{C_i}{(1 + k_i)^i}$$


Pour chaque trajectoire on en déduit une nouvelle valeur de V. On peut alors en déduire la VaR en recherchant le P quantile. La difficulté de cette méthode consiste à connaître le mouvement régissant l'évolution des taux d'intérêts. On peut par exemple partir de l'équation différentielle suivante :

$$dr = \mu(t, r(t))dt + \sigma(t, y(t))dW$$

Il est possible de générer les trajectoires de prix de deux ou plusieurs actifs corrélés à partir du tirage de deux ou plusieurs nombres aléatoires indépendants, distribués selon une loi normale standard. Prenons l'exemple de trois actifs. Soit η_1, η_2, η_3 les trois nombres obtenus par tirage aléatoire et $\varepsilon_1, \varepsilon_2, \varepsilon_3$ les trois termes aléatoires pour produire la discrétisation appropriée. Les termes ε doivent être construits comme suit :

$$\begin{cases} \varepsilon_1 = \eta_1 \\ \varepsilon_2 = a\eta_1 + b\eta_2 \\ \varepsilon_3 = c\eta_1 + d\eta_2 + e\eta_3 \end{cases}$$

L'expression des termes a, b, c, d, e est obtenue à partir d'un système de cinq équations. Les deux premières stipulent que la variance des termes ε_1 et ε_2 doit être égale à l'unité. Les trois autres permettent de faire en sorte que les termes ε construits respectent les niveaux de corrélation $\rho_{i,j}$ qui sont ceux existant entre les actifs sous-jacents du portefeuille détenu.

$$\begin{cases} \sigma^2(\varepsilon_2) = a^2 + b^2 = 1 \\ \sigma^2(\varepsilon_3) = c^2 + d^2 + e^2 = 1 \\ \rho(\eta_1, a\eta_1 + b\eta_2) = a = \rho_{1,2} \\ \rho(\eta_1, c\eta_1 + d\eta_2 + e\eta_3) = \rho_{1,3} \\ \rho(a\eta_1 + b\eta_2, c\eta_1 + d\eta_2 + e\eta_3) = ac + bd = \rho_{2,3} \end{cases}$$

Soit, sous forme matricielle :

$$\begin{pmatrix} 1 & 0 & 0 \\ a & b & 0 \\ c & d & e \end{pmatrix} \begin{pmatrix} 1 & a & c \\ 0 & b & d \\ 0 & 0 & e \end{pmatrix} = \begin{pmatrix} 1 & \rho_{1,2} & \rho_{1,3} \\ \rho_{2,1} & b & \rho_{2,3} \\ \rho_{3,1} & \rho_{3,2} & 1 \end{pmatrix}$$

En notant A la première partie du terme de gauche de l'équation et Ψ la matrice des corrélations il vient :

$$AA' = \Psi$$

La factorisation de Ψ à partir du produit d'une matrice triangulaire et de sa transposée est connue sous le terme décomposition de Cholesky. L'expression des termes qui interviennent dans A est obtenue à l'aide d'une approche itérative. La détermination des éléments figurant dans la j^{ème} colonne de la matrice A à l'exception de la première suppose que l'on ai préalablement déterminé la valeur des éléments figurant dans les j-1 colonnes qui précèdent. En notant $a_{i,j}$ (respectivement $V_{i,j}$) l'élément de a matrice A (respectivement Ψ) situé i^{ème} ligne, je colonne, on a :

$$a_{i,i} = \sqrt{V_{i,i} - \sum_{k=1}^{i-1} a_{i,k}^2}$$

$$a_{i,j} = \frac{V_{i,j} - \sum_{k=1}^{i-1} a_{j,k} a_{i,k}}{a_{j,j}}$$

Pour $i = j+1, \dots, n$.

La relation suivante permet de simuler les taux de rentabilité de chaque actif.

$$r_i = \left(\mu_i - \frac{\sigma_i^2}{2} \right) \Delta t + \sigma_i \varepsilon_i \sqrt{\Delta t}$$

- Calcul de la VaR à l'aide des méthodes historiques et paramétriques en VBA. Construction des trois fonctions de «VaR Historique»/«VaR Paramétrique Risk Metrics»/«VaR Paramétrique Monte-Carlo» (CF Projet2.xlsm – Module Library).

Méthode Historique :

```
Function Get_VaRHistoActif(vVecteurPrix As Variant, centile As Double)
    Dim i As Long
    Dim vVecteurVariation As Variant, vSortdVecteur As Variant, dRang As Double

    vVecteurVariation = Get_VecteurVariation(vVecteurPrix)
    vSortdVecteur = Get_SortShell(vVecteurVariation)
    dRang = Int((centile * UBound(vSortdVecteur)))

    Get_VaRHistoActif = vVecteurVariation(dRang, 1)
End Function
```

```

Function Get_SortShell(vVecteurVariation)
Dim i As Long, j As Long, h As Long, v As Double
h = LBound(vVecteurVariation)
Do Until h > UBound(vVecteurVariation)
h = 3 * h + 1
Loop
Do Until h = LBound(vVecteurVariation)
h = h / 3
For i = h + 1 To UBound(vVecteurVariation)
v = vVecteurVariation(i, 1): j = i
Do While vVecteurVariation(j - h, 1) > v
vVecteurVariation(j, 1) = vVecteurVariation(j - h, 1): j = j - h
If j <= h Then Exit Do
Loop
vVecteurVariation(j, 1) = v
Next i
Loop
Get_SortShell = vVecteurVariation
End Function

```

Méthode Paramétrique RiskMetrics :

```

'hypothèse : sensibilité = 1
Function Get_DeaRptf(vMatricePrix As Variant, vVecteurPonderation As Variant, dProba As Double) As Double
Dim vMatriceVarCovar As Variant, vVarianceGlobale As Variant, vSigmaGlobal As Variant
Dim dalphai As Double

vMatriceVarCovar = Get_VarCovarMatrix(vMatricePrix)
vVarianceGlobale = Get_VarianceGlobale(vVecteurPonderation, vMatriceVarCovar)
vSigmaGlobal = vVarianceGlobale ^ (1 / 2)
dalphai = Application.NormInv(dProba, 0, 1)

Get_DeaRptf = vSigmaGlobal * dalphai
End Function

Function Get_DeaRi(vVecteurPrix As Variant, dVi As Double, dSi As Double, dProba As Double) As Double
Dim dalphai As Double, dSigmai As Double, vVecteurVariation As Variant
vVecteurVariation = Get_VecteurVariation(vVecteurPrix)
dSigmai = Get_StDev(Get_VarianceRdt(vVecteurVariation))
dalphai = Application.NormInv(dProba, 0, 1)
Get_DeaRi = dVi * dSi * dalphai * dSigmai
End Function

```

Méthode Paramétrique (Monte-Carlo) :

```

Function TrajectoireMonteCarlo(S, t, r, q, sigma, nbper, nbsim) As Variant
'S = Stock, Mu = average, Sigma = Vol, T = Maturity, nbPer = nb de Période, nbSim = nb de trajectoire
'Return Monte Carlo trajectory
ReDim vMatriceResultat(0 To nbper, 0 To nbsim)
stockinit = S
For i = 0 To nbsim
vMatriceResultat(0, i) = S
For j = 1 To nbper
Randomize
rd1 = Rnd()
random = Application.NormSInv(rd1)
vMatriceResultat(j, i) = S * Exp((r - q - sigma ^ 2 / 2) * (t / nbper) + sigma * random * Sqr(t / nbper))
S = vMatriceResultat(j, i)
Next j
S = stockinit
Next i
TrajectoireMonteCarlo = vMatriceResultat
End Function

Function Get_VaRMC(vMatriceTrajectoire, dProba As Double) As Variant
Dim i As Long, j As Long
Dim vVecteurPeriode1() As Double, vVecteurPeriode2() As Double
Dim vVecteurResultat() As Double
Dim vsortVecteur1, vsortVecteur2
Dim dRang1, dRang2

ReDim vVecteurPeriode1(1 To UBound(vMatriceTrajectoire, 2), 1 To 1)
ReDim vVecteurPeriode2(1 To UBound(vMatriceTrajectoire, 2), 1 To 1)
ReDim vVecteurResultat(1 To 2, 1)

For i = LBound(vMatriceTrajectoire, 2) + 1 To UBound(vMatriceTrajectoire, 2)
vVecteurPeriode1(i, 1) = vMatriceTrajectoire(2, i - 1)
vVecteurPeriode2(i, 1) = vMatriceTrajectoire(UBound(vMatriceTrajectoire), i)
Next i

vsortdVecteur1 = Get_SortShell(vVecteurPeriode1)
vsortdVecteur2 = Get_SortShell(vVecteurPeriode2)

dRang1 = Int(dProba * UBound(vsortdVecteur1))
dRang2 = Int(dProba * UBound(vsortdVecteur2))

vVecteurResultat(1, 1) = vsortdVecteur1(dRang1, 1) / 100 - 1
vVecteurResultat(2, 1) = vsortdVecteur2(dRang2, 1) / 100 - 1

Get_VaRMC = vVecteurResultat
End Function

```

Méthode Paramétrique (Monte-Carlo) -> Décomposition de Cholesky

```

Function Get_Cholesky(MatriceCorrelation As Variant) As Variant
Dim i As Long, j As Long
Dim A() As Double
ReDim A(1 To UBound(MatriceCorrelation), 1 To UBound(MatriceCorrelation))
For j = 1 To UBound(MatriceCorrelation)
For i = 1 To UBound(MatriceCorrelation)
If i < j Then
A(i, j) = 0
ElseIf i = j Then
A(i, i) = Get_Diagonnale(MatriceCorrelation, A, i)
Else:
A(i, j) = Get_Hors_Diagonnale(MatriceCorrelation, A, i, j)
End If
Next i
Next j
Get_Cholesky = A
End Function
Function Get_Diagonnale(MatriceCorrelation, A, i)
Get_Diagonnale = Sqr(MatriceCorrelation(i, i) - Get_Somme_1(A, i - 1))
End Function
Function Get_Somme_1(A, k_up)
Dim k As Long
Dim total As Double
If k_up < 1 Then
Somme_1 = 0
Else:
total = 0
For k = 1 To k_up
total = total + A(k_up + 1, k) ^ 2
Next k
Somme_1 = total
End If
End Function
Function Get_Hors_Diagonnale(MatriceCorrelation, A, i, j)
Get_Hors_Diagonnale = (MatriceCorrelation(j, i) - Get_Somme_2(A, i - 1, j)) / A(j, j)
End Function
Function Get_Somme_2(A, k_up, j)
Dim k As Long
Dim total As Double
If k_up < 1 Then
Somme_2 = 0
Else:
total = 0
For k = 1 To k_up
total = total + A(j, k) * A(k_up + 1, k)
Next k
Somme_2 = total
End If
End Function

```

DEUXIEME PARTIE : Optimisation dynamique

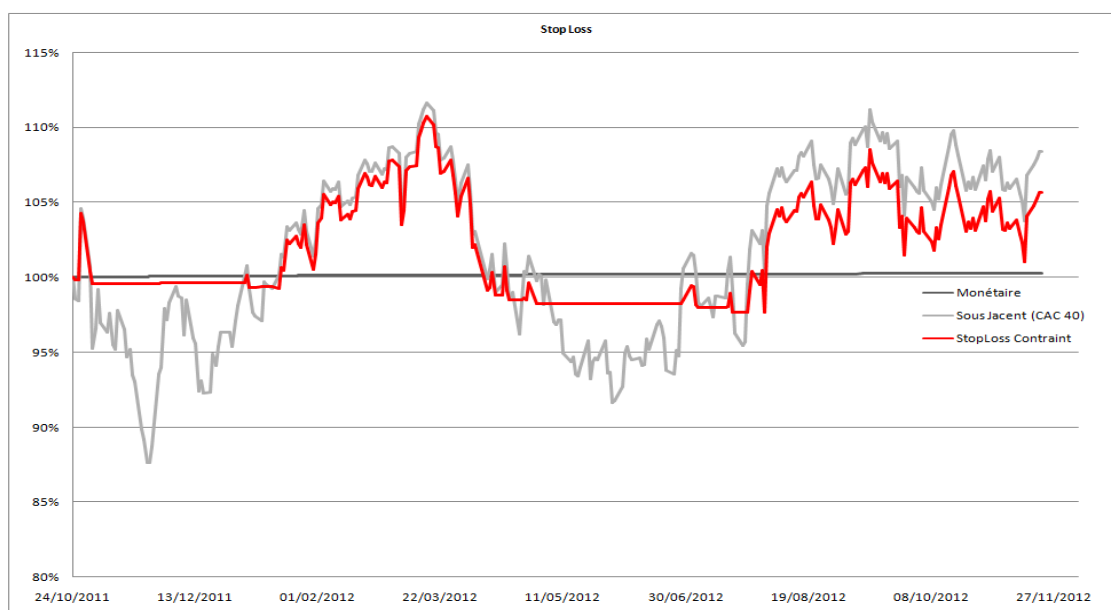
Rappel : Markowitz formalise les notions de diversification et d'efficience. Dans les exemples étudiés précédemment, l'investisseur prend une position initiale puis ne réalise aucune transaction jusqu'à une date terminale (stratégie « buy and hold »). Dans un tel contexte de gestion, il n'y a pas de révision de portefeuille entre deux dates, on parle alors d'optimisation statique. Nous allons maintenant aborder les stratégies dites dynamiques c'est-à-dire intégrant des variations de l'allocation initiale entre deux dates. La gestion indicielle, abordé précédemment, permet d'effectuer des arbitrages afin de répliquer la performance du portefeuille à celle du benchmark. Il s'agit d'une gestion passive en ce sens qu'elle recherche avant tout à répliquer (dans une certaine mesure) le comportement de l'indice de référence. Elle ne permet pas de couvrir la valeur du portefeuille contre un choc de marché. Dans une approche de gestion active on cherchera avant tout à couvrir le portefeuille contre des chocs de marché important. Pour y parvenir on distinguera tout d'abord l'allocation stratégique (quelles grandes classes d'actif intégrer dans le portefeuille) de l'allocation tactique (quelle quantité de titre, et même quel titre, intégrer dans les classes d'actifs définis préalablement). Il existe trois approches liées à l'allocation stratégique : les règles de bon sens, l'assurance de portefeuille, les modèles d'optimisation dynamique. Nous développerons principalement les points 2 et 3. En assurance de portefeuille, trois qualités sont requises : avoir une valeur minimum assurée, la probabilité de crever le planché doit être proche de 0, en cas de hausse du sous jacent, la rentabilité doit être prévisible. Nous allons maintenant étudier les principales stratégies d'assurance de portefeuille.

1- Stop Loss :

Analyse de la valeur dans le temps d'un portefeuille fictif (exprimé dans notre exemple en base 100) $M(t)$ et constitué uniquement d'actifs monétaires, c'est-à-dire supposés être sans risque.

Les fonds sont alloués dans un portefeuille risqué (par exemple un indice action) pour une valeur $S(0)$ en $t=0$.

L'intégralité de S est liquidée pour acheter M dès que $S(t) < M(t)$ et on liquide l'intégralité de M dès que $S(t) > M(t)$ pour acheter S .



De nombreuses critiques peuvent être faites concernant cette méthode notamment le problème des coûts de transaction qui, s'ils sont trop nombreux (ce qui se produit lorsque l'indice action a une grande volatilité autour de $M(t)$) vont venir éroder significativement la performance du portefeuille. C'est pour cette raison que l'on cherchera à définir un interval de tolérance dans lequel le portefeuille ne sera pas arbitré.

- Développement d'une stratégie stop-loss en VBA :
 - a. Développer une fonction de calcul intégrant la méthode du stop loss.
 - b. Développer une fonction de calcul intégrant la méthode du stop loss ainsi que les coûts de transaction.
 - c. Pour quel coût de transaction la valeur créée est nulle à la fin de la période.
 - d. Intégrer un interval de confiance sur la période pour lequel la méthode n'est pas destructrice de valeur.

NB: Il s'agit ici d'un travail effectué en backtesting. Par conséquent nous allons utiliser les sets de données disponibles dans la feuille data. Nous prendrons comme actif risqué l'indice CAC40 et comme actif sans risque l'indice Eonia.

- a. Construction de la fonction Stop Loss :

```
Function Get_VecteurStopLoss(vVecteurAction As Variant, vVecteurMonetaire As Variant)
'Construction par BackTesting du vecteur StopLoss
    Dim i As Long
    Dim vVecteurStopLoss()
    ReDim vVecteurStopLoss(LBound(vVecteurAction) To UBound(vVecteurAction), 1 To 1)
    For i = LBound(vVecteurAction) To UBound(vVecteurAction)
        If vVecteurAction(i, 1) >= vVecteurMonetaire(i, 1) Then
            vVecteurStopLoss(i, 1) = vVecteurAction(i, 1)
        ElseIf vVecteurAction(i, 1) < vVecteurMonetaire(i, 1) Then
            vVecteurStopLoss(i, 1) = vVecteurMonetaire(i, 1)
        End If
    Next i
    Get_VecteurStopLoss = vVecteurStopLoss
End Function
```

- b. Fonction Stop Loss intégrant les coûts de transaction

```

Function Get_StopLossContraint(vVecteurAction As Variant, vVecteurMonetaire As Variant, vVecteurStopLoss As Variant, dCoutArbitrage As Double) As Variant
Dim i As Long, j As Long
Dim vVecteurArbitrage() As String
ReDim vVecteurArbitrage(LBound(vVecteurStopLoss) To UBound(vVecteurStopLoss), 1 To 1)
Dim vVecteurCoutTransaction() As Double
ReDim vVecteurCoutTransaction(LBound(vVecteurStopLoss) To UBound(vVecteurStopLoss), 1 To 1)
Dim vVecteurVLrecalculee() As Double
ReDim vVecteurVLrecalculee(LBound(vVecteurStopLoss) To UBound(vVecteurStopLoss), 1 To 1)
'-->construction d'une "VL" (Valeur liquidative) du portefeuille intégrant les couts de transaction
'-Recherche des arbitrages possibles
For i = LBound(vVecteurAction) To UBound(vVecteurAction)
If vVecteurAction(i, 1) >= vVecteurMonetaire(i, 1) Then
vVecteurArbitrage(i, 1) = "Action"
ElseIf vVecteurAction(i, 1) < vVecteurMonetaire(i, 1) Then
vVecteurArbitrage(i, 1) = "Monetaire"
End If
Next i
'-Integration des couts de transaction pour chaque arbitrage
For i = LBound(vVecteurArbitrage) + 1 To UBound(vVecteurArbitrage)
If vVecteurArbitrage(i, 1) <> vVecteurArbitrage(i - 1, 1) Then
vVecteurCoutTransaction(i, 1) = dCoutArbitrage
Else:
vVecteurCoutTransaction(i, 1) = 0
End If
Next i
'-Recalcul de la VL du portefeuille après paiement des coûts de transaction
For i = LBound(vVecteurCoutTransaction) To UBound(vVecteurCoutTransaction)
If vVecteurCoutTransaction(i, 1) <> 0 Then
For j = i To UBound(vVecteurStopLoss)
vVecteurStopLoss(j, 1) = vVecteurStopLoss(j, 1) - vVecteurCoutTransaction(i, 1)
Next j
End If
Next i
Get_StopLossContraint = vVecteurStopLoss
End Function

```

c. Recherche du cout de transaction Maximum :

```

Function Get_CoutTransactionMax(vVecteurAction, vVecteurMonetaire)
Dim i As Double
Dim vVecteurMaxCout As Variant
Dim vStopLoss As Variant

For i = 0.000001 To 1 Step 0.000001
vStopLoss = Get_VecteurStopLoss(vVecteurAction, vVecteurMonetaire)
vVecteurMaxCout = Get_StopLossContraint(vVecteurAction, vVecteurMonetaire, vStopLoss, i)
If vVecteurMaxCout(UBound(vVecteurMaxCout), 1) < vVecteurMonetaire(UBound(vVecteurMonetaire), 1) Then Exit For
Next i

Get_CoutTransactionMax = i - 0.000001
End Function

```

2- OBPI :

L'assurance de portefeuille à base d'options (OBPI) permet d'assurer une valeur minimale au portefeuille à une échéance déterminée tout en permettant de bénéficier de l'éventuelle hausse du sous-jacent. Il existe deux méthodes : l'assurance de portefeuille à l'aide d'un put acheté et l'assurance de portefeuille à l'aide d'un put répliqué. La position peut s'écrire alors :

$$[\text{Sous jacent} + \text{put d'échéance } T] = \text{portefeuille}$$

Dans ce montage, l'appréciation du put permettra de couvrir la baisse éventuelle du sous-jacent. Ainsi, à l'échéance T , la valeur $V(T)$ d'une position constituée d'un sous-jacent et d'un put sur S de prix d'exercice E s'écrit :

$$V(T) = \begin{cases} S(T) & \text{si } S(T) > E \\ S(T) + E - S(T) & \text{si } S(T) \leq E \end{cases}$$

$$\text{D'où } V(T) = \text{Max}[S(T); E]$$

Par conséquent, E constitue la valeur plancher en dessous de laquelle la valeur du portefeuille ne peut théoriquement pas descendre. Notons que pour limiter la prime du put, il conviendra d'acheter une option de type européenne (exercable uniquement à l'échéance) plutôt qu'une option de type américaine (exercable jusqu'à la date d'échéance). De plus, pour les grandes échéances et les strikes très éloignés de la monnaie, les options sont

peu liquides et par conséquent coûtent cher. Dans le cadre de la réplication du put, notons $P(S,t)$ le prix théorique d'un put à l'instant t , donc distant de l'échéance de $(T-t)$, de prix d'exercice E , le prix du sous-jacent étant $S(t)$. L'idée étant maintenant de répliquer ce put avec une position B d'actif monétaire longue et une position de sous-jacent courte. La valeur du put selon le modèle de Black a Scholes s'écrit :

$$P(t) = -S(t)N(-d1) + Ke^{-rt}N(-d2)$$

Avec :

$$d1 = \frac{1}{\sigma\sqrt{t}} \left[\ln\left(\frac{S0}{K}\right) + r\frac{1}{2}\sigma^2T \right]$$

$$d2 = d1 - \sigma\sqrt{t}$$

Le portefeuille risqué contient l'actif risqué plus le put. Il vient :

$$S(t) + P(t) = S(t) - S(t)N(-d1) + Ke^{-rt}N(-d2)$$

$$S(t) + P(t) = S(t)[1 - N(-d1)] + Ke^{-rt}N(-d2)$$

$$S(t) + P(t) = S(t)N(-d1) + Ke^{-rt}N(-d2)$$

L'idée est maintenant de trouver quelle proportion $\alpha(t)$ doit être placée dans l'actif risqué et $(1-\alpha(t))$ dans l'actif sans risque. Il vient :

$$\alpha(t) = \frac{S(t)N(d1)}{S(t)N(d1) + Ke^{-rt}N(-d2)}$$

$$1 - \alpha(t) = \frac{Ke^{-rt}N(-d2)}{S(t)N(d1) + Ke^{-rt}N(-d2)}$$

Afin d'assurer le portefeuille, sur l'horizon de temps fixé, les positions sur l'actif risqué et l'actif sans risque doivent en permanence être ajustées en raison de la variabilité de l'action et également de l'effet du passage du temps. Il faut maintenant déterminer la valeur du strike pour garantir un planché donné.

$$K[S(0) + P(K)] = K$$

Avec $P(K)$ la prime du Put d'exercice K . La formule de B/S n'étant pas inversible, il faut passer par une méthode numérique pour la résoudre.

- Utilisation de l'algorithme de Newton-Raphson
- En effectuant un développement de Taylor d'ordre 1 de la relation précédente autour de $K(0)$ qui est la valeur d'amorçage de l'algorithme il vient :

$$K[S(0) + P(K(0))] + (K - K(0)) \frac{\delta\{K[S(0) + P(K)]\} - K}{\delta K}(K(0)) = 0$$

La dérivée du prix B/S d'un put par rapport au strike étant égale à :

$$e^{-rT} N(-d_2)$$

Il vient :

$$K_1 = \frac{k * K_0 e^{-rT} N(-d_2) - K * [S(0) + P(K_0)]}{K e^{-rT} N(-d_2) - 1}$$

De manière plus générale :

$$K_{n+1} = \frac{k * K_n e^{-rT} N(-d_2) - K * [S(0) + P(K_0)]}{K e^{-rT} N(-d_2) - 1}$$

- Développer en VBA une procédure de calcul intégrant la méthode OBPI.

Dans un premier temps nous allons présenter la procédure puis nous analyserons les résultats du calcul. Ce calcul est développé sur la base de l'historique de valeur de notre base de données. Il s'agit en quelque sorte d'un « back testing ».

```

Sub OBPI_Histo()
Dim S0 As Double, mu As Double, sigma As Double, r As Double, k As Double, Kprs As Double, dt As Double, spot As Double, alpha As Double, dist_ech As Double
Dim PondSJ As Double, PondMonetaire As Double, TauxRtbSJ As Double, Position As Double
Dim vVecteurPerf() As Double
Dim nbPoint As Long, nbSimul As Long, i As Long
Dim vVecteurPrix As Variant, vVecteurActionBase100 As Variant

'intégration des données
With Sheets("Data")
    vVecteurPrix = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown)).Value
    vVecteurActionBase100 = Get_VecteurBase100(vVecteurPrix)
    S0 = vVecteurActionBase100(1, 1) * 100
    mu = Get_LissageMuSigma(Get_AverageReturn(vVecteurPrix, 1, 2), UBound(vVecteurPrix), 1)
    sigma = Get_LissageMuSigma(Get_StDev(Get_Variance(vVecteurPrix)), UBound(vVecteurPrix), 2)
End With
With Sheets("AssurancePtf")
    r = .Range("Put").Cells(5, 2).Value
    k = .Range("Put").Cells(6, 2).Value
    nbPoint = UBound(vVecteurPrix)
    nbSimul = .Range("nbSimul").Cells(1, 2).Value
End With

'recherche d'une valeur précise du strike
Kprs = Get_StrikeByItart(S0, k, r, sigma, 1, k / 100, 0.00000001)

dt = 1 / nbPoint
ReDim vVecteurPerf(1 To nbPoint, 1 To 3)
spot = S0
dist_ech = 1
alpha = Get_PctSJ(spot, Kprs, r, sigma, dist_ech)
PondSJ = alpha * S0
PondMonetaire = (1 - alpha) * S0
vVecteurPerf(1, 3) = 1
vVecteurPerf(1, 1) = alpha
vVecteurPerf(1, 2) = (1 - alpha)
|
    For j = 1 To nbPoint - 1
        TauxRtbSJ = vVecteurActionBase100(j + 1, 1) - vVecteurActionBase100(j, 1)
        spot = vVecteurActionBase100(j + 1, 1)
        Position = PondSJ * Exp(TauxRtbSJ) + PondMonetaire * Exp(r * dt)
        dist_ech = dist_ech - dt
        alpha = Get_PctSJ(spot * 100, Kprs, r, sigma, dist_ech)
        PondSJ = alpha * Position
        PondMonetaire = (1 - alpha) * Position
        vVecteurPerf(j + 1, 3) = Position / 100
        vVecteurPerf(j + 1, 1) = alpha
        vVecteurPerf(j + 1, 2) = (1 - alpha)
    Next j
' écriture du résultat

With Sheets("AssurancePtf")
    .Range(.Cells(9, "h"), .Cells(UBound(vVecteurPerf) + 8, "j")).Value = vVecteurPerf
End With

End Sub

```

Les premières lignes du code concernent le chargement des données dans des variables ainsi que le paramétrage initial de l'algorithme. Dans notre exemple, nous devons déterminer le vecteur des prix de l'indice que nous allons passer en base 100. La fonction de passage en base 100 est la suivante :

```

Function Get_VecteurBase100(VecteurPrix As Variant) As Variant
Dim i As Long
Dim VecteurBase100() As Variant
ReDim VecteurBase100(LBound(VecteurPrix) To UBound(VecteurPrix), LBound(VecteurPrix, 2) To UBound(VecteurPrix, 2))

For i = LBound(VecteurPrix) To UBound(VecteurPrix)
    VecteurBase100(i, 1) = VecteurPrix(i, 1) / VecteurPrix(1, 1)
Next i
Get_VecteurBase100 = VecteurBase100
End Function

```

Nous intégrons aussi le valeur initiale de l'action, la moyenne des rendements ainsi que la volatilité historique. Les données de la plage de cellule « Put » de la feuille « AssurancePtf » nous permettent de charger dans deux variables, le taux sans risque ainsi que le strike de l'option. Enfin nous choisissons le nombre de points qui

```

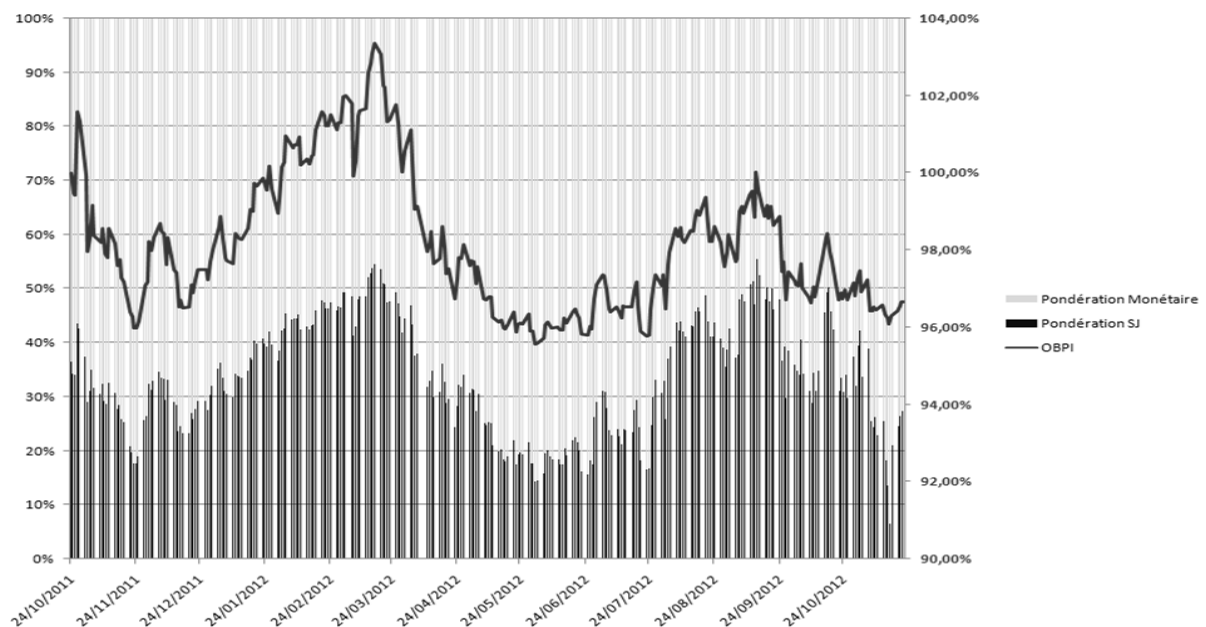
Function Get_PctSJ(spot As Double, Kprs As Double, r As Double, sigma As Double, dist_ech As Double)
Dim d1 As Double, d2 As Double
d1 = (Application.Ln(spot / Kprs) + (r + sigma ^ 2 / 2) * dist_ech) / (sigma * Sqr(dist_ech))
d2 = d1 - sigma * Sqr(dist_ech)

Get_PctSJ = spot * Application.NormSDist(d1) / (spot * Application.NormSDist(d1) + Kprs * Exp(-r * dist_ech) * Application.NormSDist(-d2))
End Function

```

correspond au nombre d'observations du vecteur prix. La proportion à détenir initialement en actif risqué (alpha) est calculée à l'aide de la fonction `Get_PctSJ` :

On peut ainsi déduire le montant à investir en actif risqué et en actif sans risque que l'on stockera dans les variables `PondSJ` et `PondMonetaire`. La procédure itérative qui suit va permettre d'ajuster en conséquence les positions que devra prendre l'investisseur pour assurer le montant qu'il souhaite garantir à l'échéance, donné par la variable `plancher`. A chaque itération, le taux de rentabilité historique du vecteur action est sélectionné dans le vecteur action, ce qui permettra de calculer le nouveau prix de l'actif. Il est alors possible de déterminer la nouvelle valeur de la position de l'investisseur : la position en actif risqué s'apprécie à hauteur du taux de rentabilité tandis que la position en actif sans risque est capitalisée au taux sans risque sur l'intervalle de temps dt . Dans un même temps, la distance à l'échéance est réduite de dt .



Une deuxième procédure a été développée. Celle-ci ne se base plus sur un historique de valeurs comme la précédente, mais sur une simulation des rentabilités des actions dans le futur. Le code de la procédure est le suivant :

```
Sub OBPI_Simul()
Dim S0 As Double, mu As Double, sigma As Double, r As Double, k As Double, Kprs As Double, dt As Double, spot As Double, alpha As Double, dist_ech As Double
Dim PondSJ As Double, PondMonetaire As Double, TauxRtbSJ As Double, Position As Double
Dim vVecteurPerf() As Double
Dim nbPoint As Long, nbSimul As Long, i As Long
Dim vVecteurPrix As Variant

'intégration des données
With Sheets("Data")
    vVecteurPrix = .Range(.Cells(2, 2), .Cells(2, 2).End(xlDown)).Value
End With
With Sheets("AssurancePtf")
    S0 = .Range("Put").Cells(2, 2).Value
    mu = .Range("Put").Cells(3, 2).Value
    sigma = .Range("Put").Cells(4, 2).Value
    r = .Range("Put").Cells(5, 2).Value
    k = .Range("Put").Cells(6, 2).Value
    nbPoint = .Range("nbPoints").Cells(1, 2).Value
    nbSimul = .Range("nbSimul").Cells(1, 2).Value
End With

'recherche d'une valeur précise du strike
Kprs = Get_StrikeByItart(S0, k, r, sigma, 1, k / 100, 0.00000001)

dt = 1 / nbPoint
ReDim vVecteurPerf(1 To nbSimul, 1 To 1)

For i = 1 To nbSimul
    spot = S0
    dist_ech = 1
    alpha = Get_PctSJ(spot, Kprs, r, sigma, dist_ech)
    PondSJ = alpha * S0
    PondMonetaire = (1 - alpha) * S0
    For j = 1 To nbPoint
        TauxRtbSJ = Get_TauxRtbSJ(mu, sigma, dt)
        spot = spot * Exp(TauxRtbSJ)
        Position = PondSJ * Exp(TauxRtbSJ) + PondMonetaire * Exp(r * dt)
        dist_ech = dist_ech - dt
        alpha = Get_PctSJ(spot, Kprs, r, sigma, dist_ech)
        PondSJ = alpha * Position
        PondMonetaire = (1 - alpha) * Position
    Next j
    vVecteurPerf(i, 1) = Position
Next i
End Sub
```

3- CPPI :

La méthode du coussin (CPPI). Cette méthode repose sur une gestion réalisée en parallèle de deux portefeuilles :

- Le portefeuille géré : $S(t) + M(t)$
- Le portefeuille fictif, composé d'un planché P et d'un coussin C(t).

L'idée et d'essayer d'obtenir en permanence une proportion constante d'exposition au risque. Cet objectif est atteint en maintenant une position en actif risqué proportionnelle au coussin.

Le niveau d'exposition est donné par le rapport :

$$m * = \frac{S(t)}{C(t)}$$

Ce rapport est appelé le multiple.

$$\lambda^* = \frac{C(t)}{S(t)}$$

Ce rapport est appelé ratio de gestion.

Notons :

$$C(t) = V(t) - P$$

Exemple numérique :

Partie 1 :

Notons $V(0) = 1000$, $P = 900$ donc $C(0) = 100$. Supposons que $m^* = 4$ et $\lambda^* = 0.25$.

Par conséquent : $S(0) = m * C(0) = 400$

Donc la part de l'actif monétaire en $t = 0$ est égale à 600.

Partie 2 :

On intègre maintenant un paramètre de tolérance égal à 20%, ce qui implique que les seuils d'intervention $m(t)$ sont égaux à 3.2 et 4.8. ($4 \pm 20\%$) et par conséquent $\lambda(t)$ est compris entre 0.2083 et 0.3125.

Supposons que le cours de S s'élève de 10%, S passe donc à 440. La valeur du portefeuille est alors de 1040. Le coussin est de 140, le multiple devient alors $440 / 140 = 3.14 < 3.2$ et le ratio $0.3182 > 0.3125$.

L'investisseur rétablit m à 4 et λ à 0.25 en achetant des titres risqués pour 120 et en vendant des actifs monétaires pour le même montant. En effet on aura bien $440 + 120 = 560 = 4 * 140$.

Le portefeuille comprendra donc 560 investis en actif risqués et 480 investis en actif sans risque.

Ainsi l'équation générale à résoudre pour obtenir les achats/ventes du titre risqué est la suivante :

$$S(t+) - S(t-) = m(V(t) - P) - S(t-) = 4 * (1040 - 900) - 440 = 560 - 440 = 120$$

- Développer une procédure de calcul intégrant la méthode OBPI.

La fonction développée est la suivante :

```

Function Get_CPPI(dPlancher As Double, dmEtoile As Double, dTolerance As Double, vVecteurAction As Variant, vVecteurMonetaire As Variant) As Variant
    Dim dCt As Double, dSt As Double, dMt As Double, Vi As Double, dArbitrage As Double
    Dim i As Long
    Dim vVecteurResultat() As Variant
    ReDim vVecteurResultat(LBound(vVecteurAction) To UBound(vVecteurAction), 1 To 4)

    For i = LBound(vVecteurAction) To UBound(vVecteurAction)
        'initialisation des valeur sur la première valeur du vecteur action
        If i = LBound(vVecteurAction) Then
            'on part de d'une valeur arbitraire du portefeuille = 1
            dVi = 1
            dCt = dVi - dPlancher
            dSt = dmEtoile * dCt
            dMt = dVi - dSt
            vVecteurResultat(i, 1) = dSt
            vVecteurResultat(i, 2) = dMt
            vVecteurResultat(i, 3) = dSt * vVecteurAction(i, 1) + dMt * vVecteurMonetaire(i, 1)
        Else:
            dSt = vVecteurResultat(i - 1, 1) * vVecteurAction(i, 1)
            dMt = vVecteurResultat(i - 1, 2) * vVecteurMonetaire(i, 1)
            dVi = dSt + dMt
            dCt = dVi - dPlancher
            dMt = dVi - dSt
            If dSt / dCt < dmEtoile Or dSt / dCt > dmEtoile Then
                dArbitrage = dMt * (dVi - dPlancher) - dSt
                dSt = vVecteurResultat(i - 1, 1) + dArbitrage
                dMt = vVecteurResultat(i - 1, 2) - dArbitrage
                vVecteurResultat(i, 1) = dSt
                vVecteurResultat(i, 2) = dMt
                vVecteurResultat(i, 3) = dSt * vVecteurAction(i, 1) + dMt * vVecteurMonetaire(i, 1)
            Else:
                dSt = vVecteurResultat(i - 1, 1)
                dMt = vVecteurResultat(i - 1, 2)
                vVecteurResultat(i, 1) = dSt
                vVecteurResultat(i, 2) = dMt
                vVecteurResultat(i, 3) = dSt * vVecteurAction(i, 1) + dMt * vVecteurMonetaire(i, 1)
            End If
        End If
    Next i

    Get_CPPI = vVecteurResultat
End Function

```

- La procédure d'appel de la fonction et d'écriture du résultat est la suivante :

```

Sub CPPI()
    Dim vVecteurMonetaireBase100 As Variant, vVecteurActionBase100 As Variant
    Dim V0 As Double, dPlancher As Double, dmEtoile As Double, dTolerance As Double
    Dim vVecteurPerf As Variant

    'intégration des données
    With Sheets("Data")
        vVecteurMonetaireBase100 = Get_VecteurBase100(.Range(.Cells(2, 8), .Cells(2, 8).End(xlDown)).Value)
        vVecteurActionBase100 = Get_VecteurBase100(.Range(.Cells(2, 2), .Cells(2, 2).End(xlDown)).Value)
    End With

    With Sheets("AssurancePtf")
        dPlancher = .Range("InputCPPI").Cells(2, 2).Value
        dmEtoile = .Range("InputCPPI").Cells(3, 2).Value
        dTolerance = .Range("InputCPPI").Cells(4, 2).Value
    End With

    vVecteurPerf = Get_CPPI(dPlancher, dmEtoile, dTolerance, vVecteurActionBase100, vVecteurMonetaireBase100)

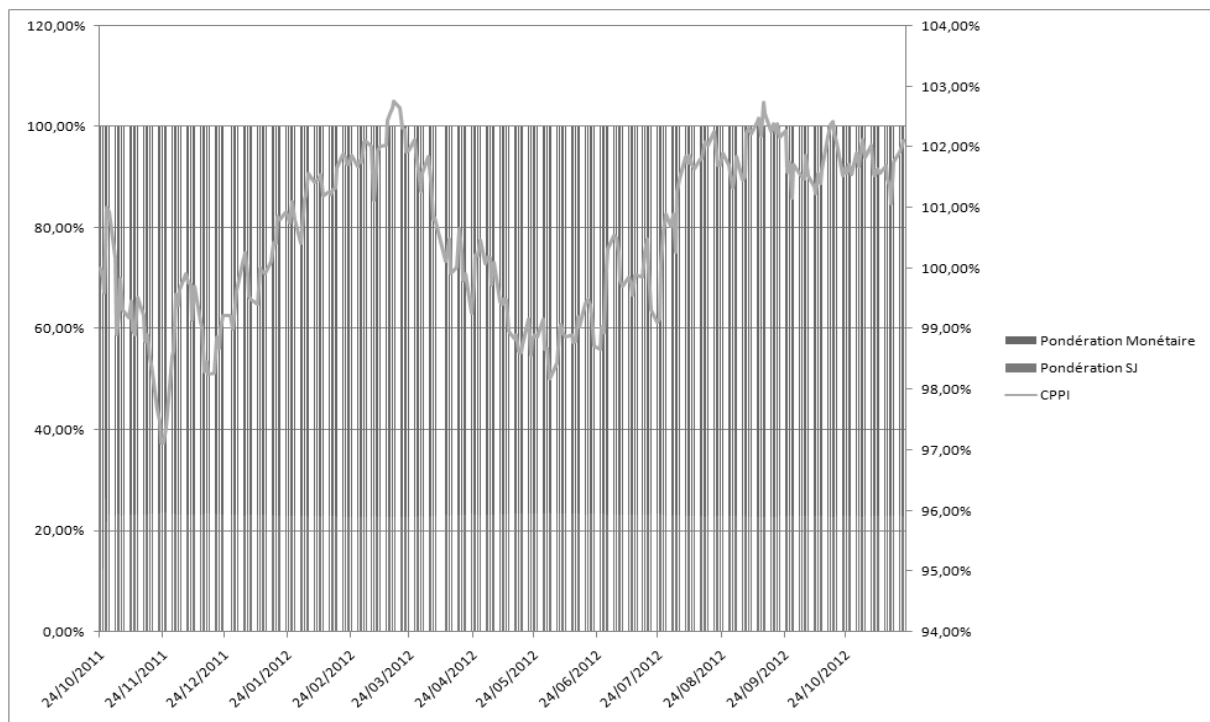
    ' écriture du résultat

    With Sheets("AssurancePtf")
        .Range(.Cells(9, "n"), .Cells(UBound(vVecteurPerf) + 8, "q")).Value = vVecteurPerf
    End With

End Sub

```

La représentation graphique du résultat du back-testing est la suivante :



Annexes :

