# 1. PASSER DE VBA AUX NOUVELLES FONCTIONS DYNAMIQUES

## 1.1. Introduction

Comme nous pouvons l'observer, de manière générale VBA est sur la pente descendante. Microsoft semble l'abandonner peu à peu. Il est donc logique de se tourner vers les nouveaux outils mis à notre disposition. Concernant l'analyse de données, l'introduction des formules dynamiques dans Excel 2021 initie un tournant assez conséquent. Les nouvelles formules mises à notre disposition permettent de complètement revoir nos feuilles de calcul en supprimant le problème de copier/coller/étirer les formules.

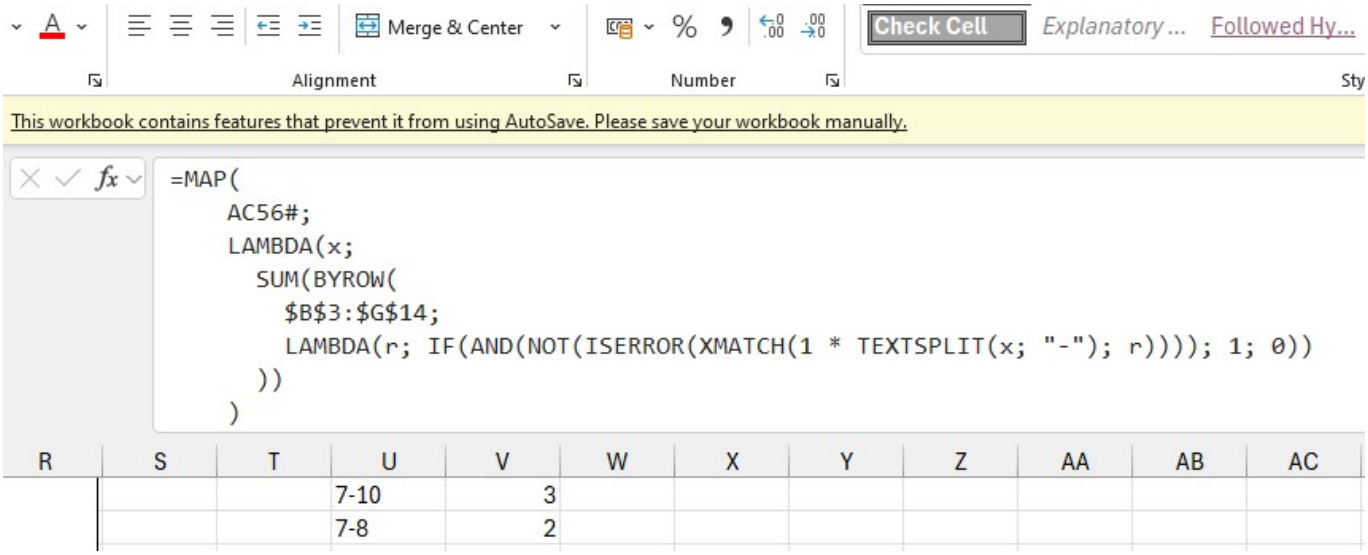
L'élaboration d'une formule dynamique est en réalité très proche de celle d'un code VBA. Ainsi les utilisateurs du VBA devraient s'y retrouver facilement, une fois les nouvelles fonctions "en tête". Je vous propose ci-après un petit dictionnaire destiné aux utilisateurs du VBA un peu perdus (comme je l'ai été, et je pense beaucoup de monde) face à toutes ces nouvelles fonctions mais qui aimeraient les utiliser. L'idée étant de faire un parallèle avec leurs équivalents VBA, et de donner des exemples simples d'application.

Pour finir, comme en VBA il n'y a pas "**une** bonne réponse" mais une infinité de solutions à un problème donné. On peut toujours obtenir le résultat de différentes manières, après plus on réduit les opérations et mieux c'est bien entendu.

## 1.2. Raisonnement

Avant toute chose, le plus important pour écrire une formule complexe est comme celui d'un Sub en VBA : il faut d'abord définir un mode opératoire. Càd réfléchir aux étapes qui vont nous permettre de passer de nos valeurs d'entrées à notre résultat final. Pour ça, une feuille et un crayon sont très efficaces.

Ensuite un autre point, les formules présentées sont souvent beaucoup plus longues que les formules classiques. Je vous conseille vraiment d'allonger votre barre de formules vers le bas afin de mieux lire la formule.




Dans la suite de ce guide, on réfèrera à un tableau 2D par le mot **matrice**, et à un tableau 1D par le mot **liste**. Les arguments valables pour les matrices le sont bien évidemment pour les listes, mais l'autre sens n'est pas vrai.

Le guide est assez long, pour les plus pressés pensez à utiliser `CTRL+F` pour chercher un mot-clé (Excel ou VBA).

## 1.3. Remerciements

Merci à @JFL pour son aide relative aux traductions des fonctions Anglais/Français, @LooReeD pour les tests de rendu HTML en MP, et aux membres du forum Excel-Pratique de manière générale tant pour leurs questions que pour leurs réponses.

## 1.4. Table des matières


- 1. PASSER DE VBA AUX NOUVELLES FONCTIONS DYNAMIQUES
  - 1.1. Introduction
  - 1.2. Raisonnement
  - 1.3. Remerciements
  - 1.4. Table des matières
  - 1.5. Guide des équivalences
    - 1.5.1. Notations & nommage des variables
    - 1.5.2. Fonctions de traitement
    - 1.5.3. Fonctions de création
    - 1.5.4. Fonctions de découpe/sélection
    - 1.5.5. Fonctions "utilitaires" de VBA
    - 1.5.6. Fonctionnalités Excel


## 1.5. Guide des équivalences

### 1.5.1. Notations & nommage des variables

- `A1#` : Plage dynamique. Si la formule en A1 renvoie une matrice, écrire `=A1#` renvoie l'ensemble de la matrice. L'avantage étant que si cette plage s'étend ou se réduit, on travail toujours avec le nombre de lignes/colonnes exact. Super utile pour réutiliser les résultats d'une autre matrice dynamique !
- `LET` : Création d'un espace de travail (comme un `Sub`/`Fonction`/`Module`/`For` en VBA : les variables ne sont accessibles que dans ce bloc). Dans un `LET` essentiellement on va définir des variables puis leur affecter des valeurs. C'est comme le `Dim` en VBA. Cela va nous permettre de stocker des résultats intermédiaires qui ne seront pas ré-évalués, et donc accélérer significativement le temps de calcul. Syntaxe LET
- `LAMBDA` : Equivalent des `Fonction` VBA. On utilise typiquement un `LAMBDA` dans un `LET` pour définir une action que l'on va réaliser de nombreuses fois. On peut ensuite appeler le LAMBDA dans l'ensemble du `LET`. Le `LAMBDA` est aussi utilisé dans d'autres fonctions comme `MAP` ou `REDUCE`. **Note :** Pour les utilisations plus avancées comme la récursivité (fonction qui s'appelle elle-même, par exemple le calcul de la suite de Fibonacci), il faut définir le LAMBDA dans le gestionnaire de noms Excel. Syntaxe LAMBDA
- `SUPPR.PLAGE` (ou `A.:.A`, appel implicite) : Equivalent de `Range.End(xlUp)` en VBA. On lit souvent que les formules de type "`=A:A`" sont à éviter, et c'est vrai. Mais parfois on est obligé de se baser sur une plage qui n'est pas un tableau ni une matrice dynamique. Dans ce cas, je conseille vraiment de travailler avec cette nouvelle fonction `SUPPR.PLAGE`. Elle va scanner la plage en question, et renvoyer uniquement la plus petite matrice ("rectange") possible contenant toutes les cellules utilisées. Bon si

vous avez des lignes vides dans la plage initiale, elles seront conservées, mais les lignes/colonnes vides AVANT (la 1e valeur) et APRES (la dernière valeur) seront supprimées. Syntaxe SUPPR.PLAGE

### 1.5.2. Fonctions de traitement

- `UNIQUE` : Equivalent de l'objet `Scripting.Dictionary` en VBA. Cependant une différence importante est que UNIQUE, quand il est utilisé avec plusieurs colonnes va renvoyer les lignes UNIQUES... En regardant toutes les colonnes ! Donc c'est super pratique pour les cas "d'unicité composée". Syntaxe UNIQUE
- `TRIER`/`TRIERPAR` : Equivalent de la fonction `ArrayList.Sort` en VBA. Mais encore une fois, surtout avec `TRIERPAR` on a beaucoup de flexibilité. On peut bien évidemment filtrer une plage par une autre plage (exemple une liste de produits par leurs prix), mais on peut aussi utiliser `TRIERPAR` sur la même plage pour effectuer un tri complexe, par exemple sur un préfixe, un suffixe ou une expression REGEX ! Syntaxe TRIER/Syntaxe TRIERPAR
- `FILTRE` : Equivalent de `Range.AutoFilter` en VBA. Alors on perd l'option de filtrage par couleur certes, mais on ne le répètera jamais assez : dans Excel les couleurs ne sont pas des informations. Pour le reste c'est très complet et en fonction de la "géométrie" des plages données, on peut filtrer verticalement mais aussi horizontalement sans problème ! Utile pour retirer les `#ERREUR` et les lignes vides par exemple. Syntaxe FILTRE
- `MAP` : Equivalent de l'utilisation courante des boucles `For` en VBA, pour chaque donnée d'une matrice. **Mais : MAX de 1 résultat (ou "cellule") par entrée**. Donc si on a besoin par exemple d'extraire plusieurs résultats 2 solutions : ou bien concaténer les valeurs avec un caractère précis, puis le réutiliser dans un `FRACTIONNER.TEXTE`, ou bien utiliser une autre fonction comme `MAKEARRAY` ou `REDUCE` pour boucler sur notre matrice et extraire autant de résultat que nécessaire. Syntaxe MAP **Nota :** Les fonctions `BYCOL` et `BYROW` fonctionnent sur le même principe. L'unique différence est que ces fonctions vont lire la matrice liste par liste (1 ligne/colonne à la fois). Donc cela veut dire que notre variable d'itération dans le `LAMBDA`, au lieu de contenir une valeur unique contiendra une liste de valeurs (uniquement quand on travaille sur une matrice. Sur une liste, `MAP` et `BYROW`/`BYCOL` sont équivalents). Syntaxe BYCOL/Syntaxe BYROW
- `REDUCE` : Le "vrai" équivalent d'une boucle `For Next` en VBA. `REDUCE` est très versatile. On peut reproduire la plupart des autres fonctions Excel en utilisant `REDUCE`. Concrètement cette fonction permet de parcourir les éléments d'une matrice *dans l'ordre*, et d'y effectuer une opération. Sa particularité est qu'elle dispose d'une mémoire des opérations précédentes. Cela permet de réutiliser l'information des calculs précédents dans le calcul courant. C'est-à-dire qu'on va disposer de deux variables avec cette fonction : l'accumulateur et le variant. Le variant va prendre une à une toutes les valeurs de la matrice, et subir une opération. L'accumulateur va prendre le (dernier) résultat de cette opération et le garder en mémoire. En théorie/usage simple, l'accumulateur va prendre une valeur unique : l'objectif de la fonction étant de *réduire* une matrice (à une valeur unique). Cependant l'accumulateur peut prendre comme valeur "unique"... Une matrice ! Donc on peut ainsi reconstruire une matrice à partir d'une autre matrice (en utilisant `ASSEMB.V`/`ASSEMB.H` sur l'accumulateur). Syntaxe REDUCE
- `SCAN` : totaux intermédiaires / petit frère de `REDUCE`. A vrai dire je n'utilise pas trop cette fonction, mais j'ai vu qu'elle pouvait être utile par exemple pour faire des totaux cumulatifs par période. Syntaxe SCAN

### 1.5.3. Fonctions de création

- `ASSEMB.V`/`ASSEMB.H` : Equivalent de `ArrayList.Add` en VBA. Ces deux fonctions vont "stacker" (comprendre "placer cote à cote") les éléments données. Cela peut être des valeurs uniques, des listes ou bien des matrices. Donc pour créer des tableaux assez complexes, ces fonctions peuvent être plus pratiques que le VBA et toutes les boucles nécessaires ! Syntaxe ASSEMB.V/Syntaxe ASSEMB.H
- `MAKEARRAY` : En utilisant `MAKEARRAY` on a à disposition 2 variables : l'indice de ligne (couramment noté "r" ou "l") et l'indice de colonne ("c"). On peut faire le parallèle en VBA avec les doubles boucles `For` (typiquement sur `i` et `j`) pour parcourir & remplir un tableau. Donc ici, on va pouvoir indiquer le calcul *en fonction des indices de ligne/colonne* dans le `LAMBDA` associé. *Rappel, on peut récupérer les éléments aux coordonnées `i`, `j` d'une autre matrice `A1:C5` avec `INDEX(A1:C5; i; j)`*.
- `SEQUENCE` : Création d'une liste de valeurs. Utile pour reproduire l'équivalent d'un `For i = 1 To 10` par exemple. On peut l'utiliser pour créer une liste d'abscisses pour un graphique, générer des ID/indexes, parcourir un texte caractère par caractère (combinaison de `STXT` + `SEQUENCE` + `NBCAR`, mais les nouvelles fonctions REGEX et TEXTE... permettent souvent de se passer de ces boucles), etc. Syntaxe MAKEARRAY

### 1.5.4. Fonctions de découpe/sélection

- `PRENDRE` : Permet de rapidement récupérer les *x* premières (ou dernières avec `-1`, `-2`...) lignes/colonnes d'une matrice. Très utile pour récupérer une ligne d'en-tête par exemple. Syntaxe PRENDRE
- `EXCLURE` : L'inverse de `PRENDRE`, permet de rapidement retirer les x premières (ou dernières avec `-1`...) lignes/colonnes d'une matrice. Très utile pour *supprimer* une ligne d'en-tête par exemple. Syntaxe EXCLURE
- `CHOISIRCOLS`/`CHOISIRLIGNES` : Très versatile, permet de sélectionner une (ou des) ligne(s)/colonne(s) d'une matrice via leur numéro (Attention ! Les numéros en question sont leurs positions relatives dans la matrice, pas dans Excel. Les numéros de ligne/colonne sont "locaux" et ne correspondent pas aux numéros de ligne/colonne de la feuille). On peut sélectionner plusieurs éléments en listant leurs numéros dans le second argument. Par exemple pour prendre les colonnes `1,2 et 3` de la matrice `B2:X5` (CàD la plage `B2:D5`), on écrit `CHOISIRCOLS(B2:X5; ASSEMB.V(1;2;3))` ou encore `CHOISIRCOLS(B2:X5; {1;2;3})` mais attention cette dernière version dépend de vos paramètres de régionaux (langue de Windows). Syntaxe CHOISIRLIGNES

### 1.5.5. Fonctions "utilitaires" de VBA

- `REGEX.REMPLACER`: L'équivalent de Replace en VBA mais qui supporte en plus les expressions régulières (pour prendre en main ces "formules magiques", je vous conseille d'abord de lire quelques tutoriels et ensuite d'utiliser l'IA : en général elle ne se débrouille pas trop mal pour "traduire" une demande humaine en expression régulière. Exemple de prompt : `donne moi l'expression régulière qui extrait les 3 premiers numéros se situant après le tiret "-" du texte "abc123-456789`. Pensez à toujours tester & vérifier de manière approfondie vos expressions régulières). Attention par défaut les expressions régulières sont "sensibles à la casse" ("a" != "A"), alors qu'Excel ne l'est pas. Syntaxe REGEX.REMPLACER
- `REGEX.TEST`: En parlant de tester vos expressions, cette fonction renvoie `VRAI` si au moins une correspondance est trouvée, et `FAUX` sinon. Syntaxe REGEX.TEST
- `REGEX.EXTRAIRE`: Très très utile pour extraire une information d'un texte. Une remarque importante : le 3e argument (optionnel) de la fonction `REGEX.EXTRAIRE`, si égal à `1`, vous permet d'extraire **tous** les groupes qui correspondent au pattern. Donc par exemple `=REGEX.EXTRAIRE("abc1231asfsf4564df`

`456";"\d+";1)` va vous renvoyer les 3 groupes de nombres `[1231, 4564, 456]` dans une liste horizontale. Syntaxe REGEX.EXTRAIRE

- `TEXTE.AVANT`/`TEXTE.APRES`/`FRACTIONNER.TEXTE` : encore plus simple que le VBA avec les découpages de type `InStr`/`InStrRev`. Bon on ne va pas entrer dans les détails, les fonctions sont éponymes et la documentation est très claire. Syntaxe FRACTIONNER.TEXTE
- `1*`: conversion de Booléen ou Texte en Nombre. Pratique dans le cas où vous faites des opérations du type `NB.SI` avec des conditions complexes. A la place, il suffit de faire la `SOMME` des conditions`*1`.
- `INDEX` : Permet de récupérer l'élément à ligne i (et potentiellement colonne j) d'une matrice. Pour info si la matrice donnée comme 1e argument est 1D (une liste), l'histoire de horizontal/vertical n'importe pas. Ainsi `INDEX(ASSEMB.H(1;2;3); 2) = INDEX(ASSEMB.V(1;2;3); 2)`. De plus, `INDEX(ASSEMB.V(1;2;3);ASSEMB.V(2;3))` renvoie les éléments 2 et 3 stackés (verticalement). Enfin, utiliser 0 comme numéro de ligne permet de renvoyer une ligne/colonne entière (comme le fait la fonction `CHOISIRLIGNES` en fait). Syntaxe INDEX
- `TABLEAU.EN.TEXTE` : Conversion "rapide" d'un tableau en texte (un peu comme `JOINDRE.TEXTE` mais avec moins d'options).

### 1.5.6. Fonctionnalités Excel

- `GROUPER.PAR`: faire un TCD rapide de regroupement de valeurs par 1 critère commun. Utile pour trier des valeurs par groupe par exemple. Pas besoin de boucler sur une liste, trouver les en-têtes uniques, puis regrouper les valeurs. Excel le fait automatiquement. De plus l'avant-dernier argument (optionnel) permet de filtrer les plages si on veut retirer certaines lignes (comme des en-têtes intermédiaires par exemple). Attention par contre, `GROUPER.PAR` ne fonctionne qu'avec des COLONNES. Pensez à utiliser `TRANSPOSE` ou `TOCOL` si vous avez un tableau horizontal. Syntaxe GROUPER.PAR
- `PIVOTER.PAR`: TCD (regroupement par critères multiples). Pareil que `GROUPER.PAR` mais avec un regroupement vertical et horizontal. Syntaxe PIVOTER.PAR
- `TOROW`/`TOCOL`: pour transformer une matrice (2D) en une liste (1D). Ou bien transposer une liste.
- `ORGA.LIGNES`/`ORGA.COLS` : pour "wrapper" comprenez "réorganiser en"/"aligner sur" une matrice/liste. Pour faire simple, vous allez demander à Excel de lire la matrice (ligne par ligne, du haut vers le bas) et de faire un retour à la ligne/à la colonne au bout de `n` éléments. C'est pratique si par exemple vous avez une liste du type `[produit1, valeur1, produit2, valeur2,...]` et que vous souhaitez aligner les produits et leurs valeurs. Syntaxe ORGA.COLS

---

saboh12617 le 29/04/2025