Le modèle tableur.mod 0.4

Pour TeXgraph 1.99

18 juillet 2021

Résumé

Ce modèle permet l'affichage et la gestion de tableaux (le modèle *variations.mod* est requis). Il propose également un certain nombre de macros pour faire du calcul matriciel.

Table des matières

TS2	Maths	Phys	LVI	LVII	Philo	Moy.	Rangs
Coef	3	3	2	1	2		
Alain	12.00	13.00	Abs	12.00	14.00	12.78	2
Edouard	6.00	10.00	16.00		9.00	9.80	6
Eglantine	11.00	8.00	14.00	10.00	11.00	10.64	4
Roger	14.00	15.00	18.00	15.00	16.00	15.45	1
Simone	Abs	9.00	11.00		12.00	10.43	5
Zebulon	10.00	14.00	10.00	14.00	13.00	12.00	3
Moyenne	10.60	11.50	13.80	12.75	12.50	11.85	
Ecart-type	2.65	2.63	2.99	1.92	2.22	1.90	

1 Introduction

Ce modèle commence par charger le modèle *variations.mod* puis le fichier de macros *matrix.mac*. Lors du chargement une variable globale est créée : **CrtMatrix**, elle représente la matrice courante et sa valeur par défaut est $[0,0]$, la plupart des macros du modèle *tableur* travaillent sur la matrice courante. Dans l'interface graphique de TeXgraph, à l'issue du chargement, une fenêtre de dialogue s'ouvre pour demander la taille de cette matrice, celle-ci peut ensuite être modifiée par le bouton *Dimensions* ; les cases du tableau peuvent être remplies ou modifiées à la souris. Lors de l'utilisation dans un document LaTeX, la taille peut-être réglée par la macro *dimTab(lig,col)* du modèle *variations*.

2 Le fichier *matrix.mac*

Une matrice est représentée par une liste dont les deux premiers éléments sont des entiers qui donnent la taille de la matrice : nombre de lignes suivi du nombre de colonnes. Les éléments suivants sont les coefficients de la matrice dans l'ordre lexicographique, c'est à dire : ceux de la ligne 1, suivi par ceux de la ligne 2, ... Par exemple, la matrice $\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}$ est représentée par la liste $[2,3,1,2,3,4,5,6]$. Les coefficients d'une matrice peuvent être des valeurs numériques complexes ou bien des chaînes de caractères. Bien entendu, les macros de calcul matriciel (somme, produit, ...) ne donneront un résultat qu'avec des matrices entièrement numériques. Les sous-sections qui suivent décrivent les macros du fichier *matrix.mac*.

2.1 matId

- Syntaxe: matId(⟨ n ⟩)
- Renvoie la matrice identité de taille *n*.

2.2 matFill

- Syntaxe: matFill(⟨ n ⟩, ⟨ p ⟩, ⟨ élément ⟩)
- Renvoie la matrice de taille ⟨ (n, p) ⟩ dont tous les coefficients sont égaux à ⟨ élément ⟩.

2.3 matResize

- Syntaxe: matResize(⟨ matrice ⟩, ⟨ n ⟩, ⟨ p ⟩, ⟨ élément ⟩)
- Retaille la ⟨ matrice ⟩ à la taille ⟨ (n, p) ⟩, la valeur des éventuels coefficients supplémentaires est ⟨ éléments ⟩ (0 par défaut).

2.4 matReadCsv

- Syntaxe: matReadCsv(⟨ "fichier csv" ⟩, ⟨ "délimiteur" ⟩)
- Cette macro lit le contenu du ⟨ "fichier csv" ⟩, celui-ci est censé représenter une matrice (une ligne par ligne de texte), deux éléments consécutifs d'une même ligne sont séparés par un ⟨ "délimiteur" ⟩ qui, par défaut, est la virgule (","). Chaque ligne est supposée avoir le même nombre d'éléments que la première, la valeur *ND* est affectée aux éléments vides. Les chaînes de caractères sont supposées délimitées par le caractère ". La macro renvoie la matrice qui résulte de la lecture.

Par exemple si le contenu du fichier *exemple.csv* est :

```
,"Mai", "Juin",
1,5,6,11
2,,4,4
3,,,
```

alors la matrice lue est $\begin{pmatrix} \text{ND} & \text{"Mai"} & \text{"Juin"} & \text{ND} \\ 1 & 5 & 6 & 11 \\ 2 & \text{ND} & 4 & 4 \\ 3 & \text{ND} & \text{ND} & \text{ND} \end{pmatrix}$, où **ND** est la constante prédéfinie (qui contient en réalité la chaîne de caractères `"_ND"`) qui représente une valeur non définie.

2.5 matWriteCsv

- Syntaxe: matWriteCsv(⟨ matrice ⟩, ⟨ "fichier csv" ⟩, ⟨ "délimiteur" ⟩)
- Cette macro écrit la ⟨ matrice ⟩ dans le ⟨ "fichier csv" ⟩ : une ligne de texte par ligne, deux éléments consécutifs d'une même ligne sont séparés par un ⟨ "délimiteur" ⟩ qui, par défaut, est la virgule (","). La valeur *ND* n'est pas écrite car elle correspond à un élément vide.

2.6 matShow

- Syntaxe: matShow(⟨ matrice ⟩)
- Renvoie une chaîne permettant d'afficher la matrice.

2.7 matCoef

- Syntaxe: matCoef(⟨ matrice ⟩, ⟨ i ⟩, ⟨ j ⟩)
- Renvoie le coefficient i, j de la ⟨ matrice ⟩.

2.8 matSetCoef

- Syntaxe: matSetCoef(⟨ matrice ⟩, ⟨ i ⟩, ⟨ j ⟩, ⟨ valeur ⟩)
- Renvoie le résultat obtenu en remplaçant le coefficient i, j de la ⟨ matrice ⟩ par la ⟨ valeur ⟩.

2.9 matSwapCoef

- Syntaxe: matSwapCoef(⟨ matrice ⟩, ⟨ i ⟩, ⟨ j ⟩, ⟨ k ⟩, ⟨ l ⟩)
- Renvoie le résultat obtenu en échangeant les coefficients i, j et k, l dans la ⟨ matrice ⟩.

2.10 matAdd

- Syntaxe: mathAdd(⟨ matrice A ⟩, ⟨ matrice B ⟩)
- Renvoie la somme $A + B$ (ou *Nil* si cette somme n'est pas définie).

2.11 matProd

- Syntaxe: matProd(⟨ matrice A ⟩, ⟨ matrice B ⟩)
- Renvoie le produit $A \times B$ (ou *Nil* si cette somme n'est pas définie).

2.12 matScalarMul

- Syntaxe: matScalarMul(⟨ matrice ⟩, ⟨ scalaire ⟩)
- Renvoie le produit de la ⟨ matrice ⟩ par le ⟨ scalaire ⟩.

2.13 matSetLig

- Syntaxe: matSetLig(⟨ matrice ⟩, ⟨ i ⟩, ⟨ [elem1, elem2, ..., elemN] ⟩, ⟨ colonne de départ ⟩)
- Renvoie le résultat obtenu en remplaçant dans la ⟨ matrice ⟩ la ligne ⟨ i ⟩ par ⟨ [elem1, elem2, ..., elemN] ⟩ à partir de la ⟨ colonne de départ ⟩ (les éléments en trop sont ignorés, la colonne de départ est la 1 par défaut).

Exemple : si $A = [2,3,1,2,3,4,5,6]$, c'est à dire si $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$, alors `matSetLig(A,1,[-1,-2,-3,-4])` renvoie `[2,3,-1,-2,-3,4,5,6]`, c'est à dire $\begin{pmatrix} -1 & -2 & -3 \\ 4 & 5 & 6 \end{pmatrix}$.

2.14 matGetLig

- Syntaxe: matGetLig(⟨ matrice ⟩, ⟨ i ⟩, ⟨ colonne de départ ⟩)
- Renvoie la liste des coefficients de la ligne ⟨ i ⟩ dans la ⟨ matrice ⟩ à partir de la ⟨ colonne de départ ⟩ (colonne 1 par défaut). On prendra garde au fait que le résultat renvoyé n'est pas une matrice.

2.15 matInsertL

- Syntaxe: matInsertL(⟨ matrice ⟩, ⟨ i ⟩, ⟨ [elem1, elem2, ..., elemN] ⟩,⟨ élément ⟩)
- Renvoie le résultat obtenu en insérant la ligne ⟨ [elem1, elem2, ..., elemN] ⟩ à la position numéro ⟨ i ⟩. Les éléments en trop sont ignorés, les éléments manquants auront la valeur ⟨ élément ⟩ (0 par défaut).

2.16 matDeleteL

- Syntaxe: matDeleteL(⟨ matrice ⟩, ⟨ i ⟩)
- Renvoie le résultat obtenu en supprimant la ligne ⟨ i ⟩ de la ⟨ matrice ⟩.

2.17 matSwapL

- Syntaxe: matSwapL(⟨ matrice ⟩, ⟨ i ⟩, ⟨ j ⟩)
- Renvoie le résultat obtenu après l'échange des lignes ⟨ i ⟩ et ⟨ j ⟩ dans la ⟨ matrice ⟩.

2.18 matAddL

- Syntaxe: matAddL(⟨ matrice ⟩, ⟨ i ⟩, ⟨ j ⟩, ⟨ x ⟩)
- Renvoie le résultat obtenu en ajoutant ⟨ x ⟩ fois la ligne ⟨ j ⟩ à la ligne ⟨ i ⟩ dans la ⟨ matrice ⟩.

2.19 matMultL

- Syntaxe: matMultL(⟨ matrice ⟩, ⟨ i ⟩, ⟨ x ⟩)
- Renvoie le résultat obtenu en multipliant la ligne ⟨ i ⟩ par ⟨ x ⟩ dans la ⟨ matrice ⟩.

2.20 matConcatL

- Syntaxe: matConcatL(⟨ matrice A ⟩, ⟨ matrice B ⟩,⟨ élément ⟩)
- Renvoie le résultat en concaténant les matrices ⟨ A ⟩ et ⟨ B ⟩ (avec B à droite de A). Le nombre de colonnes du résultat est le nombre de colonnes de A ajouté à celui de B. Le nombre de lignes du résultat est le maximum entre le nombre de lignes de A et celui de B, les lignes éventuellement manquantes seront complétées avec la valeur ⟨ élément ⟩ (0 par défaut).

2.21 matSetCol

- Syntaxe: matSetCol(⟨ matrice ⟩, ⟨ j ⟩, ⟨ [elem1, elem2, ..., elemN] ⟩, ⟨ ligne de départ ⟩)
- Renvoie le résultat obtenu en remplaçant dans la ⟨ matrice ⟩ la colonne ⟨ j ⟩ par ⟨ [elem1, elem2, ..., elemN] ⟩ à partir de la ⟨ ligne de départ ⟩ (les éléments en trop sont ignorés, la ligne de départ est la 1 par défaut).

Exemple : si $A = [3,2,1,2,3,4,5,6]$, c'est à dire si $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$, alors `matSetCol(A,2,[-2,-4,-6])` renvoie `[3,2,1,-2,3,-4,5,-` c'est à dire $\begin{pmatrix} 1 & -2 \\ 3 & -4 \\ 5 & -6 \end{pmatrix}$.

2.22 matGetCol

- Syntaxe: matGetCol(⟨ matrice ⟩, ⟨ j ⟩, ⟨ ligne de départ ⟩)
- Renvoie la liste des coefficients de la colonne ⟨ j ⟩ dans la ⟨ matrice ⟩ à partir de la ⟨ ligne de départ ⟩ (ligne 1 par défaut). On prendra garde au fait que le résultat renvoyé n'est pas une matrice.

2.23 matInsertC

- Syntaxe: matInsertC(⟨ matrice ⟩, ⟨ j ⟩, ⟨ [elem1, elem2, ..., elemN] ⟩, ⟨ élément ⟩)
- Renvoie le résultat obtenu en insérant la colonne ⟨ [elem1, elem2, ..., elemN] ⟩ à la position numéro ⟨ i ⟩. Les éléments en trop sont ignorés, les éléments manquants auront la valeur ⟨ élément ⟩ (0 par défaut).

2.24 matDeleteC

- Syntaxe: matDeleteC(⟨ matrice ⟩, ⟨ j ⟩)
- Renvoie le résultat obtenu en supprimant la colonne ⟨ j ⟩ de la ⟨ matrice ⟩.

2.25 matSwapC

- Syntaxe: matSwapC(⟨ matrice ⟩, ⟨ i ⟩, ⟨ j ⟩)
- Renvoie le résultat obtenu après l'échange des colonnes ⟨ i ⟩ et ⟨ j ⟩ dans la ⟨ matrice ⟩.

2.26 matAddC

- Syntaxe: matAddC(⟨ matrice ⟩, ⟨ i ⟩, ⟨ j ⟩, ⟨ x ⟩)
- Renvoie le résultat obtenu en ajoutant ⟨ x ⟩ fois la colonne ⟨ j ⟩ à la colonne ⟨ i ⟩ dans la ⟨ matrice ⟩.

2.27 matMultC

- Syntaxe: matMultC(⟨ matrice ⟩, ⟨ j ⟩, ⟨ x ⟩)
- Renvoie le résultat obtenu en multipliant la colonne ⟨ j ⟩ par ⟨ x ⟩ dans la ⟨ matrice ⟩.

2.28 matConcatC

- Syntaxe: matConcatC(⟨ matrice A ⟩, ⟨ matrice B ⟩, ⟨ élément ⟩)
- Renvoie le résultat en concaténant les matrices ⟨ A ⟩ et ⟨ B ⟩ (avec B sous A). Le nombre de lignes du résultat est le nombre de lignes de A ajouté à celui de B. Le nombre de colonnes du résultat est le maximum entre le nombre de colonnes de A et celui de B, les colonnes éventuellement manquantes sont complétées avec la valeur ⟨ élément ⟩ (0 par défaut).

2.29 matSetBloc

- Syntaxe: matSetBloc(⟨ matrice ⟩, ⟨ i ⟩, ⟨ j ⟩, ⟨ bloc=[n, p, elem1, elem2, ..., elemN] ⟩)
- Renvoie le résultat obtenu en remplaçant le bloc de taille ⟨ (n, p) ⟩ à partir de la position ⟨ i, j ⟩ par le nouveau ⟨ bloc=[n, p, elem1, elem2, ..., elemN] ⟩ (celui-ci est donc une matrice).

2.30 matExtract

- Syntaxe: matExtract(⟨ matrice A ⟩, ⟨ [indices lignes] ⟩, ⟨ [indices colonnes] ⟩)
- Renvoie la matrice extraite de *A* en ne retenant que les lignes et les colonnes dont les indices sont donnés en argument (sous forme de listes).

2.31 matTranspose

- Syntaxe: matTranspose(⟨ matrice ⟩)
- Renvoie la transposée de la ⟨ matrice ⟩.

2.32 matInv

- Syntaxe: matInv(⟨ matrice ⟩)
- Renvoie l'inverse de la ⟨ matrice ⟩ quand celle-ci existe.

3 Les variables du modèle

3.1 Variables globales

- **CrtMatrix** : elle représente la matrice courante, celle-ci est initialisée à $[0, 0]$ lors du chargement du modèle *tableur.mod*, son contenu est sauvegardé avec le graphique. La taille de la matrice courante est contenue dans deux variables du modèle *variations.mod* qui sont *NbLig* et *NbCol*, **attention à ne pas modifier directement le contenu de ces deux variables !** Pour modifier la taille de la matrice courante, il faut utiliser la macro *dimTab(lig,col)*, les éléments éventuellement en plus sont initialisés à la valeur *ND* (pour non définie).
- **alignDeci** : elle vaut 0 ou 1 (0 par défaut) et indique si les valeurs numériques dans une même colonne doivent être alignées sur la virgule ou non.
- **bordSep** : distance en cm par rapport au bord (0.1 par défaut), celle-ci est utilisée dans l'affichage des labels : pour la valeurs numériques alignées sur la virgule, c'est alors la distance au bord droit, et pour les labels qui sont des textes non centrés au milieu de la case.

3.2 Paramètres pour l'affichage

- **dollar** : ce paramètre vaut 0 ou 1 (1 par défaut), indique si les valeurs numériques doivent être délimitées par un symbole $ de chaque côté.
- **usecomma** : ce paramètre vaut 0 ou 1 (0 par défaut), indique si le séparateur décimal est une virgule ou non.
- **nbdeci** : ce paramètre indique le nombre de décimales à afficher (2 par défaut).
- **labelpos** : avant l'affichage du contenu de chaque case, cette variable est initialisée à *center* (par défaut), puis la macro *tabOptions(lig,col)* est exécutée (avec les numéros de ligne et de colonne en arguments), dans cette macro la valeur de *labelpos* peut être modifiée localement, les valeurs possibles sont *center*, *top*, *bottom*, *left* et *right*. Lorsque l'affichage n'est pas centré, la distance au bord est donnée par la variable *bordSep*.

Modification globale des paramètres : elle se fait lors de l'utilisation de la macro d'affichage *tabShowMatrix* (7).

Modification locale des options d'affichage : l'utilisateur peut créer la macro à deux arguments : `tabOptions(lig, col)` pour modifier les options d'affichage concernant telle ou telle case. Cette macro est automatiquement créée lorsqu'on est dans l'interface graphique de TeXgraph et l'utilisateur peut alors la modifier, mais cette macro peut aussi être créée avec la commande : `tabDefOptions("contenu souhaité pour la macro")`. Par exemple :

```
tabDefOptions("[lig:=%1,col:=%2,if col=NbCol And lig>1 then Color:=red fi]")
```

aura pour effet de mettre la dernière colonne en rouge à partir de la ligne 2.

4 Les macros du modèle

4.1 dimTab

- Syntaxe: dimTab(⟨ n ⟩, ⟨ p ⟩)
- Cette macro permet de (re)dimensionner la matrice courante, à l'issue de l'exécution, tous les coefficients ont la valeur indéfinie *ND*.

4.2 tabInitMatrix

- Syntaxe: tabInitMatrix(⟨ élément ⟩)
- Cette macro permet de réinitialiser tous les coefficients de la matrice courante avec la valeur de ⟨ élément ⟩. L'argument est optionnel et a par défaut la valeur indéfinie *ND*.

4.3 tabSetMatrix

- Syntaxe: tabSetMatrix(⟨ matrice ⟩)
- Cette macro affecte la ⟨ matrice ⟩ à la matrice courante.

4.4 tabReadCsv

- Syntaxe: tabReadCsv(⟨ "fichier csv" ⟩, ⟨ "délimiteur" ⟩)
- Cette macro lit le contenu du ⟨ "fichier csv" ⟩, celui-ci est censé représenter une matrice (une ligne par ligne de texte), deux éléments consécutifs d'une même ligne sont séparés par un ⟨ "délimiteur" ⟩ qui, par défaut, est la virgule (","). Chaque ligne est supposée avoir le même nombre d'éléments que la première, la valeur *ND* est affectée aux éléments vides. Les chaînes de caractères sont supposées délimitées par le caractère ". La macro affecte la matrice qui résulte de la lecture à la matrice courante.

4.5 tabWriteCsv

- Syntaxe: tabWriteCsv(⟨ "fichier csv" ⟩, ⟨ "délimiteur" ⟩)
- Cette macro écrit la matrice courante dans le ⟨ "fichier csv" ⟩ : une ligne de texte par ligne, deux éléments consécutifs d'une même ligne sont séparés par un ⟨ "délimiteur" ⟩ qui, par défaut, est la virgule (","). La valeur *ND* n'est pas écrite car elle correspond à un élément vide.

4.6 tabDefOptions

- Syntaxe: tabDefOptions("[instructions pour affichage cellule (lig, col)]")
- Permet de créer la macro *tabOptions(lig,col)* comme cela a été expliqué dans la section précédente (6).

4.7 tabShowMatrix

- Syntaxe: tabShowMatrix(⟨ [options] ⟩, ⟨ [pré-actions] ⟩, ⟨ [post-actions] ⟩)
- Utilisée dans un élément graphique, cette macro provoque l'affichage de la matrice courante (sans la grille) avec les ⟨ [options] ⟩ spécifiées, celles-ci peuvent être :
  - TeXcalc := ⟨ 0/1 ⟩ : avec la valeur 1 la macro *tabTeXLengthCalc* est appelée (voir la macro suivante), la valeur par défaut de cette option est 0.
  - usecomma := ⟨ 0/1 ⟩ : indique si le séparateur décimal est une virgule ou non (0 par défaut),
  - dollar := ⟨ 0/1 ⟩ : pour ajouter automatiquement le symbole $ autour des valeurs numériques (1 par défaut),
  - nbdeci := ⟨ nombre ⟩ : indique le nombre de décimales à afficher (2 par défaut),
  - aligDeci := ⟨ 0/1 ⟩ : permet d'aligner ou non les valeurs numériques sur le séparateur décimal dans une même colonne (0 par défaut),

– labelpos := ⟨ center/top/bottom/left/right ⟩ : précise la position par défaut des labels dans chaque cellule (*center* par défaut).

À ces options peuvent s'ajouter des changements au niveau des attributs avec *Color*, *LabelSize*, ...

Le deuxième argument ⟨ [pré-actions] ⟩ est une liste d'instructions à exécuter avant l'affichage, comme par exemple peindre des cellules avec la macro *Cadre()* du modèle *variations*.

Le troisième argument ⟨ [post-actions] ⟩ est une liste d'instructions à exécuter après l'affichage, comme par exemple dessiner la grille avec la macro *grille(options)* du modèle *variations*.

4.8 tabTeXLengthCalc

- Syntaxe: tabTeXLengthCalc()
- Cette macro calcule la longueur et la hauteur de la boite TEX pour chaque cellule de *CrtMatrix*, puis ajuste la hauteur des lignes et la largeur des colonnes en conséquence. Le calcul est fait en compilant un document LATEX avec la police *lmodern* dans une taille de 12 pt.

4.9 tabSetCoef

- Syntaxe: tabSetCoef(⟨ i ⟩, ⟨ j ⟩, ⟨ valeur ⟩)
- Modifie le coefficient ⟨ i, j ⟩ de la matrice courante en le remplaçant par la ⟨ valeur ⟩.

4.10 tabGetCoef

- Syntaxe: tabGetCoef(⟨ i ⟩, ⟨ j ⟩)
- Renvoie la valeur du coefficient ⟨ i, j ⟩ de la matrice courante.

4.11 tabSetLig

- Syntaxe: tabSetLig(⟨ i ⟩, ⟨ [elem1, elem2, ..., elemN] ⟩, ⟨ colonne de départ ⟩)
- Modifie la matrice courante en remplaçant la ligne ⟨ i ⟩ par la ligne ⟨ [elem1, elem2, ..., elemN] ⟩ à partir de la ⟨ colonne de départ ⟩ (les éléments en trop sont ignorés, la colonne de départ est la 1 par défaut).

4.12 tabGetLig

- Syntaxe: tabGetLig(⟨ i ⟩ ,⟨ colonne de départ ⟩,⟨ colonne d'arrivée ⟩)
- Renvoie la liste des coefficients de la matrice courante situés sur la ligne ⟨ i ⟩ à partir de la ⟨ colonne de départ ⟩ jusqu'à la ⟨ colonne d'arrivée ⟩.

4.13 tabInsertL

- Syntaxe: tabInsertL(⟨ i ⟩ ,⟨ [elem1, elem2, ..., elemN]] ⟩)
- Insère une nouvelle ligne ⟨ [elem1, elem2, ..., elemN] ⟩ à la position numéro ⟨ i ⟩.

4.14 tabDeleteL

- Syntaxe: tabDeleteL(⟨ i ⟩)
- Supprime la ligne ⟨ i ⟩.

4.15 tabSwapL

- Syntaxe: tabSwapL(⟨ i ⟩ ,⟨ j ⟩)
- Échange les lignes ⟨ i ⟩ et ⟨ j ⟩.

4.16 tabSetCol

- Syntaxe: tabSetCol(⟨ j ⟩, ⟨ [elem1, elem2, ..., elemN] ⟩, ⟨ ligne de départ ⟩)
- Modifie la matrice courante en remplaçant la colonne ⟨ j ⟩ par la colonne ⟨ [elem1, elem2, ..., elemN] ⟩ à partir de la ⟨ ligne de départ ⟩ (les éléments en trop sont ignorés, la ligne de départ est la 1 par défaut).

4.17 tabGetCol

- Syntaxe: tabGetCol(⟨ j ⟩, ⟨ ligne de départ ⟩,⟨ ligne d'arrivée ⟩)
- Renvoie la liste des coefficients de la matrice courante situés sur la colonne ⟨ j ⟩ à partir de la ⟨ ligne de départ ⟩ jusqu'à la ⟨ colonne d'arrivée ⟩.

4.18 tabInsertC

- Syntaxe: tabInsertC(⟨ j ⟩ ,⟨ [elem1, elem2, ..., elemN]] ⟩)
- Insère une nouvelle colonne ⟨ [elem1, elem2, ..., elemN] ⟩ à la position numéro ⟨ j ⟩.

4.19 tabDeleteC

- Syntaxe: tabDeleteC(⟨ j ⟩)
- Supprime la colonne ⟨ j ⟩.

4.20 tabSwapC

- Syntaxe: tabSwapC(⟨ i ⟩ ,⟨ j ⟩)
- Échange les colonnes ⟨ i ⟩ et ⟨ j ⟩.

4.21 tabSetBloc

- Syntaxe: tabSetBloc(⟨ i ⟩,⟨ j ⟩, ⟨ bloc=[n, p, elem1, elem2, ..., elemN] ⟩)
- Modifie la matrice courante en remplaçant le bloc de taille ⟨ (n, p) ⟩ à partir de la position ⟨ i, j ⟩ par le nouveau ⟨ bloc=[n, p, elem1, elem2, ..., elemN] ⟩ (celui-ci est donc une matrice).

4.22 tabGetBloc

- Syntaxe: tabGetBloc(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩)
- Renvoie la matrice dont les coefficients sont ceux des lignes ⟨ i ⟩ à ⟨ k ⟩ et des colonnes ⟨ j ⟩ à ⟨ l ⟩ de la matrice courante.

4.23 tabSum, tabSumL, tabSumC

- Syntaxe: tabSum(⟨ liste ⟩ [, ⟨ liste coef ⟩]) ; tabSumL(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩ [, ⟨ liste coef ⟩]) ; tabSumC(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩ [, ⟨ liste coef ⟩])
- La première macro renvoie la somme des nombres de la ⟨ liste ⟩, lorsqu'une ⟨ liste de coefficients ⟩ est donnée, chaque nombre est multiplié par le coefficient correspondant. De la même façon, les deux autres macros calculent les sommes du bloc allant de la position ⟨ i, j ⟩ à la position ⟨ k, l ⟩ suivant les lignes (*tabSumL*) ou suivant les colonnes (*tabSumC*), et renvoient la liste des résultats. **Le nombre de coefficients éventuels doit correspondre au nombre de termes dont on calcule la somme**. Les éléments de la matrice qui ne sont pas des nombres sont ignorés ainsi que la constante *jump*. Un résultat qui n'est pas défini prend la valeur de la constante *ND*.

4.24 tabProd, tabProdL, tabProdC

- Syntaxe: tabProd(⟨ liste ⟩ [, ⟨ liste coef ⟩]) ; tabProdL(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩ [, ⟨ liste coef ⟩]) ; tabProdC(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩ [, ⟨ liste coef ⟩])
- La première macro renvoie le produit des nombres de la ⟨ liste ⟩, lorsqu'une ⟨ liste de coefficients ⟩ est donnée, chaque nombre est multiplié par le coefficient correspondant. De la même façon, les deux autres macros calculent les produits du bloc allant de la position ⟨ i, j ⟩ à la position ⟨ k, l ⟩ suivant les lignes (*tabProdL*) ou suivant les colonnes (*tabProdC*), et renvoient la liste des résultats. **Le nombre de coefficients éventuels doit correspondre au nombre de termes dont on calcule le produit.** Les éléments de la matrice qui ne sont pas des nombres sont ignorés ainsi que la constante *jump*. Un résultat qui n'est pas défini prend la valeur de la constante *ND*.

4.25 tabMoy, tabMoyL, tabMoyC

- Syntaxe: tabMoy(⟨ liste ⟩) ; tabMoyL(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩) ; tabMoyC(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩)
- La première macro renvoie la moyenne des nombres de la ⟨ liste ⟩, les deux autres calculent les moyennes du bloc allant de la position ⟨ i, j ⟩ à la position ⟨ k, l ⟩ suivant les lignes (*tabMoyL*) ou suivant les colonnes (*tabMoyC*), et renvoient la liste des résultats. Les éléments de la matrice qui ne sont pas des nombres sont ignorés ainsi que la constante *jump*. Un résultat qui n'est pas défini prend la valeur de la constante *ND*.

4.26 tabVar, tabVarL, tabVarC

- Syntaxe: tabVar(⟨ liste ⟩) ; tabVarL(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩) ; tabVarC(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩)
- La première macro renvoie la variance des nombres de la ⟨ liste ⟩, les deux autres calculent les variance du bloc allant de la position ⟨ i, j ⟩ à la position ⟨ k, l ⟩ suivant les lignes (*tabVarL*) ou suivant les colonnes (*tabVarC*), et renvoient la liste des résultats. Les éléments de la matrice qui ne sont pas des nombres sont ignorés ainsi que la constante *jump*. Un résultat qui n'est pas défini prend la valeur de la constante *ND*.

4.27 tabEcart, tabEcartL, tabEcartC

- Syntaxe: tabEcart(⟨ liste ⟩) ; tabEcartL(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩) ; tabEcartC(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩)
- La première macro renvoie l'écart-type des nombres de la ⟨ liste ⟩, les deux autres calculent les écarts-types du bloc allant de la position ⟨ i, j ⟩ à la position ⟨ k, l ⟩ suivant les lignes (*tabEcartL*) ou suivant les colonnes (*tabEcartC*), et renvoient la liste des résultats. Les éléments de la matrice qui ne sont pas des nombres sont ignorés ainsi que la constante *jump*. Un résultat qui n'est pas défini prend la valeur de la constante *ND*.

4.28 tabBary, tabBaryL, tabBaryC

- Syntaxe: tabBary(⟨ liste ⟩, ⟨ coef ⟩) ; tabBaryL(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩, ⟨ coef ⟩) ; tabBaryC(⟨ i ⟩,⟨ j ⟩,⟨ k ⟩,⟨ l ⟩, ⟨ coef ⟩)
- La première macro renvoie la moyenne pondérée des nombres de la ⟨ liste ⟩ avec la liste des ⟨ coef ⟩, les deux autres calculent les moyennes pondérées du bloc allant de la position ⟨ i, j ⟩ à la position ⟨ k, l ⟩ suivant les lignes (*tabBaryL*) ou suivant les colonnes (*tabBaryC*), et renvoient la liste des résultats. **Le nombre de coefficients doit correspondre au nombre de termes dont on calcule la moyenne.** Les éléments de la matrice qui ne sont pas des nombres sont ignorés ainsi que la constante *jump*. Un résultat qui n'est pas défini prend la valeur de la constante *ND*.

4.29 tabRound

- Syntaxe: tabRound(⟨ liste valeurs ⟩, ⟨ nombre décimales ⟩)
- Cette macro renvoie la ⟨ liste des valeurs ⟩ arrondies avec le ⟨ nombre de décimales ⟩ souhaité. Les éléments de la liste que ne sont pas numériques sont renvoyés sans modification. Par exemple, `tabRound([sqrt(2),ND,1/6,"toto"],2)` renvoie comme résultat `[1.41,"_ND",0.17,"toto"]`.

4.30 tabRank

- Syntaxe: tabRank(⟨ liste ⟩,⟨ décroissant (0/1) ⟩)
- Classe la ⟨ liste ⟩ dans l'ordre demandé (décroissant par défaut) et renvoie la liste des rangs. Par exemple, la commande `tabRank([2,3,-1,3,"A",6,ND])` renvoie la liste des rangs `[4,2,5,2,7,1,6]`.

4.31 tabSortL

- Syntaxe: tabSortL(⟨ liste d'indices de lignes ⟩, ⟨ liste de clés ⟩, ⟨ décroissant (0/1) ⟩)
- Trie les lignes dont l'indice figure dans la liste, en fonction des ⟨ clés ⟩ (la première clé est celle du premier indice de ligne, la deuxième celle du deuxième indice, ...). **Il doit y avoir autant de clés que d'indices de lignes**. Le tri se fait dans l'ordre croissant par défaut, et les lignes dont l'indice ne figure pas restent à leur position.

4.32 tabSortC

- Syntaxe: tabSortC(⟨ liste d'indices de colonnes ⟩, ⟨ liste de clés ⟩, ⟨ décroissant (0/1) ⟩)
- Trie les colonnes dont l'indice figure dans la liste, en fonction des ⟨ clés ⟩ (la première clé est celle du premier indice de colonne, la deuxième celle du deuxième indice, ...). **Il doit y avoir autant de clés que d'indices de colonnes**. Le tri se fait dans l'ordre croissant par défaut, et les colonnes dont l'indice ne figure pas restent à leur position.

Exemple

```
\begin{texgraph}[name=preface, file]
Include "tableur.mod";
Graph image = [
 //les données
 matieres :=["Maths","Phys", "LVI", "LVII", "Philo"], coef :=[3,3,2,1,2],
 noms :=["Alain", "Edouard", "Eglantine","Roger", "Simone", "Zebulon"],
 effectif :=Nops(noms), NbMat :=Nops(matieres),
 notes :=[NbMat,effectif,      //taille
       12,6,11,14,"Abs",10,  //maths
       13,10,8,15,9,14,      //phys
       "Abs",16,14,18,11,10, //LVI
       12,ND,10,15,ND,14,    //LVII optionnelle
       14,9,11,16,12,13],    //Français
 //entrée des données dans le tableau
 dimTab(effectif+4, NbMat+3),
 tabSetLig(1,["TS2",matieres,"Moy.","Rangs"]), tabSetLig(2,["Coef",coef]), tabSetCol(1,[noms,"Moyenne", "Ecart-type"],3),
 tabSetBloc(3,2, matTranspose(notes)),
 //moyennes par élève et calcul des rangs
 moyennesE :=tabBaryL(3,2,effectif+2,NbMat+1, coef),
 tabSetCol(NbCol-1, moyennesE, 3), tabSetCol( NbCol, tabRank(moyennesE), 3),
 //moyennes par discipline
 moyennes :=tabMoyC(3,2,effectif+2,NbMat+2), ecarts :=tabEcartC(3,2,effectif+2,NbMat+2),
 tabSetLig(NbLig-1, moyennes, 2),  tabSetLig(NbLig, ecarts, 2),
 //options affichage
 tabDefOptions("[if %2=NbCol-1 And %1>2 then Color :=red fi, if %2=NbCol Or %1=2 then alignDeci :=0 fi]"),
 //affichage
 tabShowMatrix( [alignDeci :=1, TeXcalc :=1, LabelSize :=large], //options
          [FillStyle :=full, FillColor :=seagreen, FillOpacity :=0.5, LineStyle :=noline, Cadre(3,2,effectif+2,NbMat+1)], //pre-actions
          [grille(LineStyle :=solid), Width :=8, traitH(2), traitH(NbLig-2), traitV(1)] //post-actions
        )
];
\end{texgraph}
```

TS2	Maths	Phys	LVI	LVII	Philo	Moy.	Rangs
Coef	3	3	2	1	2		
Alain	12.00	13.00	Abs	12.00	14.00	12.78	2
Edouard	6.00	10.00	16.00		9.00	9.80	6
Eglantine	11.00	8.00	14.00	10.00	11.00	10.64	4
Roger	14.00	15.00	18.00	15.00	16.00	15.45	1
Simone	Abs	9.00	11.00		12.00	10.43	5
Zebulon	10.00	14.00	10.00	14.00	13.00	12.00	3
Moyenne	10.60	11.50	13.80	12.75	12.50	11.85	
Ecart-type	2.65	2.63	2.99	1.92	2.22	1.90	

FIGURE 1 – *Exemple de la première page*

5 Changements

5.1 Version 0.4

- Dans l'interface graphique, un clic droit dans une case du tableau permet maintenant d'effacer son contenu (et la matrice est mise automatiquement à jour).

5.2 Version 0.3

- Ajout de la macro *tabRound*.
- Modification des macros *tabSum*, *tabSumL*, *tabSumC*, *tabProd*, *tabProdL*, *tabProdC* : possibilité de multiplier chaque terme par un coefficient avant de faire la somme ou le produit.
- Correction d'un bug dans la macro d'écriture *matWriteCsv*.

5.3 Version 0.2

- Correction d'un bug dans la macro d'affichage *tabShowMatrix*.

5.4 Version 0.1

- Première version.