spreadtab

v0.5

Manuel de l'utilisateur

Christian Tellechea
unbonpetit@netc.fr

27 février 2019

Résumé

Cette extension permet d'utiliser des fonctionnalités de tableur dans n'importe quel environnement « tableau » avec LaTeX.

La principale fonctionnalité étant de pouvoir écrire des formules dans les cellules d'un tableau qui font références à d'autres cellules, de calculer les formules contenues dans les cellules et d'afficher les résultats numériques de ces formules dans le tableau.

Table des matières

1 Introduction

1.1 Présentation

Cette extension permet de construire des tableaux similaires à des feuilles de calculs. Les cellules du tableau ont des coordonnées (colonne et ligne) qui peuvent être utilisées dans des formules pour calculer des valeurs dans d'autres cellules. On charge le package en écrivant dans le préambule la syntaxe classique

```
\usepackage[<options>]{spreadtab}
```

où les options « fp » ou « xfp » spécifient le moteur de calcul choisi. Si aucune option n'est spécifiée, le moteur sera **fp**. Si les deux options sont spécifiées, le moteur actif au début du document sera **fp**, mais dans ce cas, on peut changer de moteur à tout moment en exécutant \STusefp ou \STusexfp. Cette documentation est compilée avec le moteur **xfp**, sauf exemple où le contraire est spécifié. En règle générale, les deux moteurs de calcul doivent donner des résultats identiques, mais :

- **fp** est un moteur de calcul à virgule fixe (18 chiffres avant et 18 après) et donne donc davantage de chiffres après la virgule que **xfp** qui effectue les calculs en virgule flottante avec 16 chiffres significatifs ; en revanche, **xfp** accepte nativement des tests, gère l'infini et NaN, comprend la multiplication implicite par juxtaposition, etc.
- des différences existent entre les deux moteurs concernant les fonctions disponibles tant du point de vue de la syntaxe que de leur existence même. Ainsi, la fonction `max(...)` n'admet que *deux* nombres en argument avec **fp** tandis que **xfp** accepte autant d'arguments que l'on veut. De la même façon, l'exponentielle s'obtient avec `e^x` ou `pow(x,e)` avec **fp** alors que la syntaxe avec **xfp** est `exp(x)`. Par ailleurs, les fonctions reconnues par les deux moteurs ne sont pas les mêmes, comme on peut le constater en lisant leurs documentations : `ceil` par exemple, n'existe que pour **xfp**. D'une manière générale, on peut considérer **xfp** comme étant mieux doté en fonctionnalités que **fp** ;
- les fonctions `rand` et `randint` ne sont pas disponibles avec le moteur **xfp** lorsqu'on compile avec une version de XeLaTeX trop ancienne.

Ce package nécessite le moteur ε-TeX, le format LaTeX 2_ε ainsi que le package **fp** ou **xfp** à qui sont confiés les calculs. Le package **xstring** est également requis.

J'ai souhaité dès le départ de rendre ce package compatible avec *tous* les environnements de tableaux, sous réserve que les séparateurs entre colonnes soient « & » et les retours à la ligne soient « \\ ». Cette contrainte forte sur la compatibilité m'a conduit à programmer spreadtab pour qu'il agisse d'une façon *totalement indépendante* de l'environnement tableau. Ainsi, la lecture du tableau, le traitement et le calcul des formules se fait *avant* que l'environnement tableau ne prenne la main et ne « voit » le corps du tableau.

Par conséquent, spreadtab procède en 3 étapes :

- en premier lieu, spreadtab lit le corps du tableau et le divise en lignes puis en cellules en reconnaissant dans chacune la présence d'une éventuelle formule ;
- ensuite, il procède au calcul des formules contenues dans les cellules, en ayant pris soin pour chacune de calculer auparavant les cellules dépendantes. L'ordre dans lequel les cellules doivent être calculées est déterminé par spreadtab. Les calculs sont faits par le package **fp** ou **xfp**, sous réserve que les formules à évaluer dans les cellules soient compatibles avec les syntaxes des macros \FPeval du package **fp** ou \fpeval du package **xfp** [1] ;
- enfin, il faut reconstruire le tableau en ayant remplacé chaque formule par la valeur numérique préalablement calculée et passer la main à l'environnement tableau spécifié par l'utilisateur.

Les deux syntaxes possibles (et équivalentes) sont les suivantes, où <nom> représente le nom de n'importe quel environnement de type tableau disponible avec LaTeX ou avec une extension :

```
\begin{spreadtab}{{<nom>}{<parametres>}}
    tableau avec formules et nombres
\end{spreadtab}
```

ou

```
\spreadtab{{<nom>}{<parametres>}}
    tableau avec formules et nombres
\endspreadtab
```

1. Si l'on souhaite utiliser le « ET » logique de **xfp** noté « && », il est *obligatoire* de mettre cet opérateur entre accolades dans un tableau afin que les tokens & ne soient pas compris comme des séparateurs de colonnes. Dans une cellule d'un tableau, pour tester si le contenu de la cellule a1 appartient à un intervalle, on écrira donc par exemple « `a1>1 {&&} a1<10` ».

et après le travail de spreadtab, on obtient un affichage comme si l'on avait écrit :

```
\begin{<nom>}{<parametres>}
    tableau avec nombres
\end{<nom>}
```

Même si disposer de fonctionnalités ressemblant à celles d'un tableur avec LaTeX est appréciable, il ne faut pas perdre de vue que les 3 étapes décrites ci-dessus prennent du temps. L'ensemble conduit donc à des temps de compilation *beaucoup plus importants* qu'avec un tableau classique.

Il faut ajouter que spreadtab *ne peut remplacer un tableur*. En effet, ses possibilités sont très limitées. De plus, surtout pour des tableaux complexes ou de grande taille, le manque d'aide visuelle devient gênant [2], et la syntaxe de spreadtab constitue aussi un obstacle supplémentaire. L'avantage de cette extension est de pouvoir écrire *dans le code LaTeX* des tableaux comportant des calculs, alors que ces tableaux sont généralement exportés [3] d'une feuille de calcul d'un tableur vers le code LaTeX. On évite ainsi les désagréments des programmes d'exportation : mise en forme souvent à retoucher pour obtenir exactement ce que l'on veut, non compatibilité avec tous les environnements de tableaux, obtention de tableaux ne contenant que les valeurs (les formules sont perdues à l'exportation), exportation à recommencer si l'on modifie un seul nombre ou formule dans le tableau.

1.2 Motivation

Quelques mois avant de commencer à m'attaquer à ce package, Derek O'Connor m'avait fait remarquer que rien n'était disponible dans le monde des extensions de LaTeX pour imiter — ne serait-ce qu'un peu — le calcul de formules dans des tableaux, comme cela se fait couramment avec des tableurs. J'ai trouvé le défi intéressant et je me suis lancé dans l'écriture de ce package qui en fait, n'est qu'un exercice de programmation.

La route a été longue avant d'arriver à cette version et je tiens à remercier tout particulièrement Christophe Casseau pour l'intérêt qu'il a porté dès le début à ce travail et les suggestions qu'il m'a faites, ainsi que plus récemment Derek O'Connor pour ses conseils et pour les échanges constructifs que nous avons eus. J'adresse également mes remerciements à Andrew Parsloe pour la relecture et les corrections apportées à la traduction de ce manuel en anglais.

2 Fonctionnalités courantes

Il faut noter tout d'abord qu'à l'intérieur d'un tableau sous environnement spreadtab, les caractères « : », « ; », « ! » et « ? » perdent leur code de catégorie actif qui leur a été attribué si vous utilisez l'option `french` du package `babel`. Par conséquent, l'espace automatique inséré avant ces caractères sera désactivé dans un tableau.

Par défaut, spreadtab considère qu'une ligne se termine avec "\\" qui est habituel dans les tableaux. Ce marqueur de fin de ligne peut être modifié via la commande `\STeol{<macro>}`. On peut par exemple écrire `\STeol{\tabularnewline}`. Il faut bien comprendre que les fins de lignes qui seront insérées dans le tableau final seront toujours "\\", même si le marqueur de fin de ligne lorsque spreadtab *lit* le tableau a été modifié.

2.1 Références absolues

Dans le tableau, les cellules sont repérées par leur références absolues de cette façon :

- la colonne est une lettre de a à z, a représentant la 1re colonne de gauche : on est donc d'emblée limité à 26 colonnes, ce qui devrait suffire pour la grande majorité des cas ; la lettre est insensible à la casse, elle peut donc être indifféremment minuscule ou majuscule ;
- à la suite immédiate de la lettre, un nombre entier strictement positif représente le numéro de la ligne, la ligne numéro 1 étant la ligne du haut.

2. Ceci dit, je certifie qu'avec l'habitude, cette gêne tend à s'estomper (si l'on s'en tient à des tableaux raisonnables, bien sûr).
3. On peut signaler les 2 principaux programmes d'exportation : **cacl2latex** pour « calc » de Open Office, et **excel2latex** pour « excel » de Microsoft Office.

Une référence absolue s'écrit donc par exemple : « `b4` », « `C1` » ou « `d13` » [4]. Visuellement, on peut illustrer ce fonctionnement par ce genre de tableau ressemblant à un tableur, ici volontairement limité à 5 lignes et 5 colonnes :

	A	B	C	D	E
1					
2					
3					
4					
5					

Voici un exemple où l'on calcule la somme de chaque ligne et de chaque colonne puis, la somme totale :

```
\begin{spreadtab}{{tabular}{rr|r}}
22       & 54         & a1+b1 \\
43       & 65         & a2+b2 \\
49       & 37         & a3+b3 \\
\hline
a1+a2+a3 & b1+b2+b3   & a4+b4
\end{spreadtab}
```

22	54	76
43	65	108
49	37	86
114	156	270

Pour les matheux, voici un autre exemple où l'on calcule quelques lignes du triangle de Pascal :

```
\begin{spreadtab}{{tabular}{ccccc}}
1  &       &       &       &    \\
a1 & a1    &       &       &    \\
a2 & a2+b2 & b2    &       &    \\
a3 & a3+b3 & b3+c3 & c3    &    \\
a2 & a4+b4 & b4+c4 & c4+d4 & d4
\end{spreadtab}
```

1				
1	1			
1	2	1		
1	3	3	1	
1	4	6	4	1

2.2 Références relatives

Pour faire référence à une cellule, il peut être commode de spécifier sa position par rapport à la cellule où se trouve la formule. Ainsi, les coordonnées « relatives » d'une cellule sont 2 nombres relatifs écrits selon cette syntaxe : `[x,y]`, où `x` est le décalage horizontal par rapport à la cellule contenant la formule et `y` est le décalage vertical. Ainsi, `[-2,3]` fait référence à la cellule se trouvant 2 colonnes avant (à gauche) et 3 lignes après (plus bas) la cellule où se trouve la formule.

Voici à nouveau le triangle de Pascal vu ci-dessus, mais les références sont relatives et l'environnement « `matrix` » du package **amsmath** est utilisé :

```
$
\begin{spreadtab}{{matrix}{}}
1\\
[0,-1] & [-1,-1]\\
[0,-1] & [-1,-1]+[0,-1] & [-1,-1]\\
[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]\\
[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]
\end{spreadtab}
$
```

1				
1	1			
1	2	1		
1	3	3	1	
1	4	6	4	1

On remarque que les références relatives sont plus adaptées ici puisque seulement 2 références différentes sont utilisées : `[0,-1]` qui se réfère à la cellule de dessus et `[-1,-1]` qui se réfère à la cellule au Nord-Ouest de la cellule où se trouve la formule.

On peut utiliser dans une même formule un mélange de références absolues et relatives.

4. Cette notation se retrouve dans les tableurs : la lettre représente la colonne et le nombre qui suit représente la ligne. Cet ordre est le contraire de la convention utilisée avec les matrices en mathématiques.

2.3 Cellules de texte

Si l'on veut mettre du texte dans une cellule, il faut indiquer à spreadtab que la cellule ne doit pas être calculée. Il suffit de placer quelque part dans la cellule le caractère « @ ». Ce faisant, la cellule est ignorée par spreadtab et devient une cellule inerte à qui il n'est pas possible [5] de faire référence nulle part ailleurs dans le tableau.

Voici un exemple :

```
\begin{spreadtab}{{tabular}{|r|ccc|}}
\hline
@ valeurs de $x$ & -5       &        -1 &        4 \\
@ $f(x)=2x$      & 2*[0,-1] &  2*[0,-1] & 2*[0,-1] \\\hline
\end{spreadtab}
```

valeurs de x	-5	-1	4
$f(x) = 2x$	-10	-2	8

Le caractère « @ » est le développement de la séquence de contrôle \STtextcell. On peut donc redéfinir cette séquence de contrôle en ce que l'on veut et par exemple, \renewcommand\STtextcell{'} fera qu'une cellule contenant le caractère "'" sera comprise comme étant une cellule de texte.

De plus, si une cellule est vide ou entièrement constituée d'espaces, alors spreadtab la considérera comme une cellule de texte.

2.4 Cellules mixtes

En réalité, chaque cellule est composée de *deux* champs. D'un côté le *champ numérique* qui contient la formule et de l'autre le *champ textuel* qui sera ignoré par le moteur de calcul et n'entre pas en ligne de compte pour les calculs :

- dans une cellule, si rien n'est précisé, la totalité de la cellule est considérée comme étant le champ numérique, et le champ textuel est vide (c'était le cas pour toutes les cellules du tableau du triangle de Pascal vu précédemment) ;
- si la cellule contient « @ », alors la totalité de la cellule est considérée comme étant le champ textuel. Le champ numérique est vide et inaccessible.
- si la cellule contient « := », alors l'argument entre accolades qui suit est le champ numérique, et tout le reste est le champ textuel. La cellule a cette structure :

  `<champ textuel>:={champ numérique}<suite du champ textuel>`

  On peut changer ce marqueur en la séquence de contrôle « \= » par exemple, en redéfinissant la macro \STnumericfieldmarker de cette façon :

  \renewcommand\STnumericfieldmarker{\=}

  Dans ce cas, le développement de \= n'aurait strictement aucune importance et n'interviendrait pas dans le processus : pour spreadtab, il ne s'agit que d'un marqueur de début de formule qui est cherché et reconnu sans être développé.

Une fois le « champ numérique » calculé, lui seul et le marqueur « := » seront remplacés par la valeur numérique calculée.

Il faut noter que « :={champ numérique} » peut se trouver à l'intérieur d'accolades et ce quelque soit le niveau d'imbrication. Par exemple, dans une cellule, on peut écrire \textbf{:={a1+1}}. Si le champ numérique de la cellule a1 est 5, alors la cellule contiendra au final \textbf{6}.

Pour fixer les idées, voici un exemple très simple :

```
\begin{spreadtab}{{tabular}{|c|c||c|}}\hline
valeur 1 : :={50} & valeur 2 : :={29} & moyenne : \textbf{:={(a1+b1)/2}}\\\hline
\end{spreadtab}
```

5. Il y a une exception à cette règle, voir la page 18.

valeur 1 : 50	valeur 2 : 29	moyenne : **39,5**

À noter également que « `:={}` », qui définit formule vide, a le même effet que « `@` » dans une cellule : celle-ci est comprise comme cellule de texte. Cependant, « `@` » et « `:={}` » *ne sont pas équivalents* car une cellule textuelle contenant ce dernier peut recevoir une formule par copie (voir section suivante), ce qui est impossible avec « `@` ».

2.5 Copier des formules

Pour éviter d'avoir à recopier des formules identiques dans des cellules voisines, spreadtab fournit l'instruction `\STcopy`.

Cette commande se place dans une cellule selon cette syntaxe :

`\STcopy{>`x`,v`y`}{formule}`

où x et y sont des nombres positifs qui représentent des décalages horizontaux et verticaux par rapport à la cellule où se trouve l'instruction. La cellule qui contient la commande et la cellule obtenue par ces décalages définissent une plage rectangulaire de cellules qui recevront la `<formule>` [6]. La commande `\STcopy` *ne doit pas* se trouver dans une cellule où un marqueur de champ numérique « `:=` » est présent.

Voici comment se déroule la copie : elle se fait en partant de la cellule où se trouve l'instruction, la cellule source. Pour chaque cellule cible, toutes les références dans la formule sont incrémentées pour tenir compte du décalage entre la cellule cible et la cellule source. Ainsi par exemple, mettons que la cellule source contienne la formule `a1+b2+c3`. Lorsque cette formule est copiée dans une cellule cible se trouvant 2 colonnes à droite et 5 lignes en dessous, cette formule deviendra : `c6+d7+e8`. La formule peut également contenir des références relatives qui, puisqu'elle sont relatives, ne seront pas modifiées.

En faisant précéder d'un « `!` » une coordonnée d'une référence, cette coordonnée reste inchangée lors de la copie. Reprenons l'exemple précédent avec la formule `a!1+!b2+!c!3`. Si cette formule est copiée dans la cellule se trouvant 2 colonnes à droite et 5 lignes en dessous, alors, elle deviendra : `c1+b7+c3`. Cette fonctionnalité est compatible avec des références relatives : mettons qu'une cellule contienne la formule `[-1,!-1]+[!-1,1]+[!1,!2]` et que cette formule soit copiée vers la cellule se trouvant 2 colonnes à droite et 5 lignes en dessous. Alors, elle deviendra : `[-1,-6]+[-3,1]+[-1,-3]`.

Le caractère « `!` » est le développement de la séquence de contrôle `\STtransposecar`. On peut donc modifier ce caractère en tout autre avec `\renewcommand\STtransposecar{<caractère>}`. Le point d'exclamation, utilisé par défaut, garde son code de catégorie actif qui lui a été attribué par le package `babel` chargé avec l'option `frenchb`.

Dans la syntaxe `\STcopy{>`x`,v`y`}{formule}`, si le nombre x est absent, la copie dans le sens horizontal se fait vers la droite jusqu'au bord droit du tableau. Si y est absent, la copie dans le sens vertical se fait jusqu'en bas du tableau.

Voici quelques exemples :

`{>3,v1}`	copie vers les 3 colonnes à droite et 1 ligne en dessous
`{>3}`	copie vers les 3 cellules de droite
`{v1}`	copie vers la cellule de dessous
`{>}`	copie vers toutes les cellules situées à droite
`{v}`	copie vers toutes les cellules situées en dessous
`{v,>}`	copie vers le bas et vers la droite à partir de la cellule courante

On peut facilement générer la table de multiplication de 1 à 10 :

```
\begin{spreadtab}{{tabular}{|c|*{10}{c}|}}
\hline
@$\times$       & 1                    & \STcopy{>}{b1+1}  & & & & & & & & \\\hline
1               & \STcopy{>,v}{!a2*b!1} &                   & & & & & & & & \\
\STcopy{v}{a2+1} &                     &                   & & & & & & & & \\
                &                      &                   & & & & & & & & \\
                &                      &                   & & & & & & & & \\
                &                      &                   & & & & & & & & \\
                &                      &                   & & & & & & & & \\
```

6. La copie ne peut donc se faire que vers des cellules se trouvant à droite et plus bas que la cellule dans laquelle se trouve la commande.

```
                &                   &                   & & & & & & & & \\
                &                   &                   & & & & & & & & \\
                &                   &                   & & & & & & & & \\
                &                   &                   & & & & & & & & \\\hline
\end{spreadtab}
```

×	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Lors d'une copie de formule, si la cellule cible contient déjà un champ numérique non vide, celui-ci n'est pas écrasé et la copie ne se fait pas.

Si des commandes \STcopy se trouvant dans des plusieurs cellules départ ont une même cellule cible, la formule que cette dernière reçoit est celle correspondant à la dernière commande \STcopy rencontrée lorsque le tableau est lu de haut en bas et de droite à gauche. Dans l'exemple visuel ci-dessous, le \STcopy se trouvant dans la cellule rose B1 a son champ cible partiellement recouvert par celui du \STcopy se trouvant dans la cellule verte C3. En effet, cette dernière est rencontrée plus tard lors de la lecture du tableau :

	A	B	C	D	E	F
1	1	\STcopy {v,>}{!a1+1}				
2	2					
3	3		\STcopy {>2,v1}{!a3*10}			
4	4					
5	5					

Voici cet exemple repris ci-dessous pour mettre en évidence ces comportements. Dans cet exemple, la cellule b5 qui est composée d'un champ numérique et la cellule c5 qui est mixte, leur champ numérique n'étant pas vide, il est conservé.

```
\begin{spreadtab}{{tabular}{|*6{c|}}}\hline
1 &\STcopy{v,>}{!a1+1} &                     & & & \\\hline
2 &                    &                     & & & \\\hline
3 &                    & \STcopy{>2,v1}{!a3*10} & & & \\\hline
4 &                    &                     & & & \\\hline
5 &       -1           &       a:={0}b       & & & \\\hline
\end{spreadtab}
```

1	2	2	2	2	2
2	3	3	3	3	3
3	4	30	30	30	4
4	5	40	40	40	5
5	-1	a0b	6	6	6

Comme on l'a dit à la fin du chapitre précédent, on peut également copier une formule dans une cellule de texte contenant un champ numérique vide (c'est-à-dire une cellule contenant « :={} »). Dans ce cas, la formule est copiée à l'endroit où se trouve « :={} ». Par contre, dans une cellule de texte contenant « @ », la copie ne se fait pas et la cellule reste purement textuelle.

Exemple :

```
\begin{spreadtab}{{tabular}{|*6{c|}}}\hline
1                 & 2   & 3      & 4             & 5 & 6 \\\hline
X\STcopy{>}{a1+1}Y & @XY & X:={}Y & \textbf{:={}} &   &   \\\hline
\end{spreadtab}
```

1	2	3	4	5	6
X2Y	XY	X4Y	**5**	6	7

3 Mise en forme du tableau

3.1 Séparateur décimal

Les packages **fp** et **xfp** renvoient les résultats décimaux avec le point comme séparateur décimal. Il est possible de changer ce séparateur décimal de telle sorte que tout se passe comme si les résultats retournés par **fp** ou **xfp** en tenaient compte. L'instruction \STsetdecimalsep permet de changer le séparateur décimal en n'importe quel caractère en utilisant la syntaxe :

\STsetdecimalsep{<caractère>}

Pour une utilisation en langue française, il faut donc inclure dans le préambule cette ligne :

\STsetdecimalsep{,}

Pour les champs numériques se trouvant en mode math, il faut noter que la virgule n'est pas neutre dans ce mode. En effet, elle est considérée comme une ponctuation ce qui explique qu'elle est suivie d'une espace. Pour neutraliser cet espace, on peut mettre cette virgule entre accolades comme on le voit dans ce code :

```
3,14 n'est pas affiché comme $3,14$.\par
3,14 est affiché comme $3{,}14$
```

3,14 n'est pas affiché comme 3, 14.
3,14 est affiché comme 3,14

Lorsque des cellules sont en mode math, on peut [7] s'inspirer de cette propriété et demander à spreadtab de remplacer le point décimal par une virgule entre accolades avec la commande \STsetdecimalsep{{,}}. Dans ces deux tableaux où chaque cellule est en mode math, on constate que l'espacement après la virgule est neutralisé pour le second :

```
\STsetdecimalsep{,}
\begin{spreadtab}{{tabular}{|*3{>{$}r<{$}}|}}\hline
@x   & @y  & @\text{Moyenne}\\\hline
5    & -4  & (a2+b2)/2\\
-6.1 & -8  & (a3+b3)/2\\
9.85 & 3.7 & (a4+b4)/2\\\hline
\end{spreadtab}\par\smallskip
\STsetdecimalsep{{,}}
\begin{spreadtab}{{tabular}{|*3{>{$}r<{$}}|}}\hline
@x   & @y  & @\text{Moyenne}\\\hline
5    & -4  & (a2+b2)/2\\
-6.1 & -8  & (a3+b3)/2\\
9.85 & 3.7 & (a4+b4)/2\\\hline
\end{spreadtab}
```

x	y	Moyenne
5	−4	0, 5
−6, 1	−8	−7, 05
9, 85	3, 7	6, 775

x	y	Moyenne
5	−4	0,5
−6,1	−8	−7,05
9,85	3,7	6,775

7. Il est cependant préférable d'utiliser le package `numprint` pour formatter les résultats. On peut aussi modifier le code mathématique de la virgule : `\mathcode‘,="013B\relax`. Cette modification, réservée aux utilisateurs avertis, fait entrer la virgule dans la classe 0 des signes ordinaires alors qu'elle est naturellement dans la classe 6 des signes de ponctuation.

3.2 Mise en forme des nombres avec le moteur fp

Comme cela a été précisé avec l'option **fp**, tous les calculs sont faits par le package **fp** et sa macro \FPeval [8]. Ce package fournit d'extraordinaires possibilités de calcul pour TEX et dispose de toutes les fonctions arithmétiques, scientifiques et trigonométriques usuelles. Les calculs sont faits avec une précision de 10^{-18}, et les 18 décimales sont affichées lorsqu'un calcul ne tombe pas juste ! Sans prendre des précautions, on peut se retrouver avec beaucoup de chiffres dans les parties décimales de certains résultats. Toute cette section est compilée avec \STusefp.

Pour se prémunir de ce problème, plusieurs solutions existent :

- on peut utiliser le package **numprint** qui est ce qui se fait de mieux dans l'affichage des nombres ;
- on peut demander à **fp** d'arrondir un résultat avec sa fonction `round(nombre,entier)` qui arrondit `nombre` avec `entier` chiffres après la virgule ;
- on peut également demander à spreadtab d'arrondir *tous* les nombres placés dans le tableau à une certaine précision avec la macro \STautoround dont l'argument est le nombre de chiffres demandés après la virgule. Si l'argument est vide, aucun arrondi n'est fait. Si on utilise la macro étoilée \STautoround*, la partie décimale est remplie si besoin avec des 0 inutiles (uniquement avec **fp**).

Voici un exemple, des nombres de 1 à 7 et leurs inverses sont arrondis à 10^{-6} :

```
\STautoround{6}
\begin{spreadtab}{{tabular}{|l|*7{>{\centering\arraybackslash}m{1.35cm}|}}}
\hline
@$x$     & 1    & 2    & 3    & 4    & 5    & 6    & 7   \\\hline
@$x^{-1}$& 1/b1 & 1/c1 & 1/d1 & 1/e1 & 1/f1 & 1/g1 & 1/h1\\\hline
\end{spreadtab}\medskip

\STautoround*{6}
\begin{spreadtab}{{tabular}{|l|*7{>{\centering\arraybackslash}m{1.35cm}|}}}
\hline
@$x$     & 1    & 2    & 3    & 4    & 5    & 6    & 7   \\\hline
@$x^{-1}$& 1/b1 & 1/c1 & 1/d1 & 1/e1 & 1/f1 & 1/g1 & 1/h1\\\hline
\end{spreadtab}
```

x	1	2	3	4	5	6	7
x^{-1}	1	0,5	0,333333	0,25	0,2	0,166667	0,142857

x	1,000000	2,000000	3,000000	4,000000	5,000000	6,000000	7,000000
x^{-1}	1,000000	0,500000	0,333333	0,250000	0,200000	0,166667	0,142857

Tous les nombres contenus dans le tableau final, qu'ils aient été saisis tels quels ou qu'ils proviennent du résultat d'un calcul sont traités par la macro \STprintnum. Par défaut, cette macro n'a aucun effet et est programmée de la façon suivante :

```
\newcommand\STprintnum[1]{#1}
```

Il est possible d'arrondir tous les nombres via la commande \numprint du package \numprint. Pour ce faire, il faut redéfinir la macro \STprintnum de cette façon :

```
\renewcommand\STprintnum[1]{\numprint{#1}}
\nprounddigits{6}
\begin{spreadtab}{{tabular}{cccccccc}}
@$x$   & 1 &  2 & 3  & 4  & 5  & 6  &7   \\\hline
@$1/x$ &1/b1&1/c1&1/d1&1/e1&1/f1&1/g1&1/h1
\end{spreadtab}
```

8. À ce propos, les notations infixe ou postfixe sont acceptées par \FPeval ce qui signifie que les formules dans spreadtab peuvent être indifféremment sous forme infixe ou postfixe. Par exemple, la formule infixe « a1+b1 » est équivalente aux formules postfixes « a1 b1 add » ou « a1 b1 + ».

x	1,000 000	2,000 000	3,000 000	4,000 000	5,000 000	6,000 000	7,000 000
$1/x$	1,000 000	0,500 000	0,333 333	0,250 000	0,200 000	0,166 667	0,142 857

Voici un autre exemple similaire où l'on teste si le nombre à afficher est négatif avec la commande \FPifneg du package **fp**. Si tel est le cas, le nombre est affiché en rouge. La commande \STautoround a été préférée à \nprounddigits du package numprint puisque cette dernière ajoute des 0 inutiles. On a également remis le point décimal par défaut puisque \numprint est en charge de l'affichage des nombres.

```
\STsetdecimalsep{.}
\renewcommand\STprintnum[1]{\FPifneg{#1}\color{red}\fi\numprint{#1}}
\STautoround{6}
\begin{spreadtab}{{tabular}{cccccccc}}
@$x$   & -1 &  2 & -3 & 4  & -5 & 6  & -7 \\\hline
@$1/x$ &1/b1&1/c1&1/d1&1/e1&1/f1&1/g1&1/h1
\end{spreadtab}
```

x	-1	2	-3	4	-5	6	-7
$1/x$	-1	0,5	$-0{,}333\,333$	0,25	$-0{,}2$	0,166 667	$-0{,}142\,857$

3.3 Retours à la ligne et filets horizontaux

Pour bien délimiter la fin d'une ligne, spreadtab est contraint de reconnaître les retours à la ligne et les filets horizontaux. Ce package permet d'utiliser dans le tableau l'argument optionnel de \\ de cette façon : \\[<dimension>].

Pour les filets horizontaux, on peut utiliser autant de fois que l'on veut :

- \hline ;
- \cline{x-y} où x et y sont les numéro des colonnes de départ et d'arrivée du filet ;
- \hhline{<type>} où <type> est le type de ligne désirée (voir la documentation du package **hhline**).
- n'importe quelle commande du package **booktabs**, à savoir \toprule, \midrule, \bottomrule, \cmidrule, \addlinespace, \morecmidrule et \specialrule. Tous les arguments de ces macros, optionnels ou pas sont gérés ;
- \noalign et son argument peuvent être placés après \\ et pris en compte.

Voici le triangle de Pascal inversé, et massacré pour l'exemple :

```
\begin{spreadtab}{{tabular}{*5c}}
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]\\[1em]
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]       &       \\
[0,1] & [-1,1]+[0,1] & [-1,1]       &              &       \\ \hline\hline
[0,1] & [-1,1]       &              &              &       \\ \cline{2-4}
1     &              &              &              &       \\ \hline
\end{spreadtab}
```

1	4	6	4	1
1	3	3	1	
1	2	1		
1	1			
1				

3.4 Masquer une ligne ou une colonne

Parfois, une colonne ou une ligne entière est destinée à recevoir des calculs intermédiaires qui n'ont pas à être affichés dans le tableau final. Pour cela spreadtab dispose de deux séquences de contrôle \SThiderow et \SThidecol qui, lorsqu'elles sont placées dans une cellule, masquent la ligne ou la colonne dans laquelle se trouve cette cellule.

Voici un exemple :

```
\begin{spreadtab}{{tabular}{|r|ccc|}}
\hline
@ valeurs de $x$       & -1         & 0\SThidecol & 2          & 3          \\\hline
@$f(x)=2x-1$           & 2*[0,-1]-1 & 2*[0,-1]-1  & 2*[0,-1]-1 & 2*[0,-1]-1 \\
@$g(x)=x-10$\SThiderow & [0,-2]-10  & [0,-2]-10   & [0,-2]-10  & [0,-2]-10  \\
@$h(x)=1-x$            & 1-[0,-3]   & 1-[0,-3]    & 1-[0,-3]   & 1-[0,-3]   \\\hline
\end{spreadtab}
```

valeurs de x	-1	2	3
$f(x) = 2x - 1$	-3	3	5
$h(x) = 1 - x$	2	-1	-2

On peut observer comment on masque la ligne contenant $g(x)$ et la colonne correspondant à la valeur 0.

Il faut se souvenir que les lignes et les colonnes masquées sont *invisibles* pour l'environnement tableau choisi par l'utilisateur, ce qui explique que dans le préambule du tableau, seules 4 colonnes (`|r|ccc|`) aient été définies et non 5 comme le voit spreadtab.

Pour voir la différence, voici le tableau obtenu en définissant 5 colonnes et en ne masquant aucune ligne ni colonne :

valeurs de x	-1	0	2	3
$f(x) = 2x - 1$	-3	-1	3	5
$g(x) = x - 10$	-11	-10	-8	-7
$h(x) = 1 - x$	2	1	-1	-2

3.5 Sauvegarder la valeur d'une cellule

On peut être amené à avoir besoin de la valeur numérique d'une cellule dans le tableau pour l'afficher en dehors d'une formule ou même à l'extérieur du tableau. On doit alors utiliser la commande :

```
\STsavecell{<sequence de controle>}{<reference absolue>}
```

Avec un `\global\def` [9], cette commande a pour effet de sauvegarder de façon globale dans `<sequence de controle>` le résultat de la formule contenue dans la cellule `<reference absolue>`.

On ne peut utiliser que des références *absolues* ; les références relatives ne sont pas acceptées car cette commande doit se placer dans l'argument optionnel de l'environnement `spreadtab`.

Exemple :

```
\begin{spreadtab}[\STsavecell\result{c1}]{{tabular}{|c|c|c|c|c|}}
\hline
10 & a1+10 & b1+10 & a1+b1+c1 & @cell c1 : \result\\\hline
\end{spreadtab}
\par\medskip
Voici la cellule c1 : \result
```

10	20	30	60	cell c1 : 30

Voici la cellule c1 : 30

Si l'on veut sauvegarder plusieurs cellules, on peut mettre autant de fois que l'on veut la commande `\STsavecell` dans l'argument optionnel.

Exemple :

9. La commande `\def` ne vérifie pas si la macro qu'elle définit existe déjà.

```
\begin{spreadtab}[\STsavecell\hhh{b3}\STsavecell\mmm{c3}\STsavecell\sss{d3}]{{tabular}{|rc|}}\hline
@Vitesse  (km/h) &\SThidecol&\SThidecol&\SThidecol& 35 \\
@distance (km)   &          &          &          & 180\\\hline
@Temps (h min s) & trunc(e2/e1,0) & trunc(60*(e2/e1-b3),0) & trunc(3600*(e2/e1-b3)-60*c3,1) &@\hhh\ h \mmm\ min \
     sss\ s\\\hline
\end{spreadtab}\par\medskip
On met au moins \hhh\ heures
```

Vitesse (km/h)	35
distance (km)	180
Temps (h min s)	5 h 8 min 34,2 s

On met au moins 5 heures

3.6 Afficher la valeur d'une cellule

Pour afficher la valeur du champ numérique d'une cellule dans le champ textuel, on vient de voir qu'on pouvait sauvegarder cette valeur dans une séquence de contrôle et se servir de cette séquence de contrôle pour l'afficher ensuite. Le procédé est un peu fastidieux et détourne de ses intentions la commande \STsavecell qui sert surtout à sauvegarder une valeur pour l'utiliser en dehors du tableau.

Pour afficher simplement la valeur du champ numérique d'une cellule dans un champ textuel, on peut utiliser la syntaxe <<référence>> qui sera remplacé par la valeur numérique de la cellule référence où la référence peut être relativess ou absolue. Si ce qui est entre << et >> n'est pas une référence, alors rien n'est fait et <<texte>> est laissé en l'état. La référence ne doit contenir aucun espace. Si on écrit << a1>> alors, l'espace fait que la référence n'est pas reconnue et rien ne sera fait.

Exemple dans une cellule purement textuelle a3 :

```
\begin{spreadtab}{{tabular}{lr}}
@Prix de vente                & 250  \\
@Prix d'achat                 & 216  \\\hline
@B\'en\'efice (<<b1>>-<<b2>>) & b1-b2
\end{spreadtab}
```

Prix de vente	250
Prix d'achat	216
Bénéfice (250-216)	34

Exemple dans la cellule mixte c1 :

```
\begin{spreadtab}{{tabular}{|c|c||c|}}\hline
23 & 32 & Moyenne $= \frac{<<a1>>+<<b1>>}{2}= :={(a1+b1)/2}$\\\hline
\end{spreadtab}
```

23	32	Moyenne $= \frac{23+32}{2} = 27,5$

Les caractères qui délimitent la référence valent "<<" et ">>" par défaut. Ils peuvent être modifiés avec la commande \STsetdisplaymarks dont les 2 arguments définissent le délimiteur de gauche et le délimiteur de droite. En écrivant \STsetdisplaymarks{|}{|}, on devra écrire |référence| pour provoquer l'affichage du champ numérique de la cellule référence.

3.7 Utiliser \multicolumn

Le package spreadtab est compatible avec la syntaxe \multicolumn{<nombre>}{<type>}{<contenu>} qui fusionne <nombre> cellules en une cellule de type et de contenu spécifiés dans les arguments.

En utilisant \multicolumn, spreadtab permet même de conserver une certaine cohérence au niveau du référencement des cellules. Sur ce tableau où des cellules sont fusionnées, les cases du tableau contiennent les références absolues vues par spreadtab :

a1	b1	c1	d1	e1	f1	g1
a2	b2		d2	e2	f2	g2
a3			d3	e3		g3

Ainsi, quelque soit le nombre de cellules fusionnées, la cellule suivante porte un numéro de colonne qui tient compte du nombre de cellules fusionnées.

Sur la dernière ligne, les cellules `a3`, `b3` et `c3` sont fusionnées, et si la cellule `a3` contient une formule, les cellules `b3` et `c3` *n'existent pas* pour spreadtab : on ne peut nulle part faire référence à ces cellules.

Voici un exemple où chaque nombre de la ligne du haut est le produit des 2 nombres se trouvant au dessous de lui :

```
\newcolumntype{K}[1]{@{}>{\centering\arraybackslash}p{#1cm}@{}}
\begin{spreadtab}{{tabular}{*6{K{0.5}}}}
\cline{2-5}
&\multicolumn{2}{|K{1}|}{:={a2*c2}} & \multicolumn{2}{K{1}|}{:={c2*e2}} &\\\hline
\multicolumn{2}{|K{1}}{:=8}&\multicolumn{2}{|K{1}}{:=7}&\multicolumn{2}{|K{1}|}{:=6}\\\hline
\end{spreadtab}
```

	56	42	
8	7	6	

On remarque que le marqueur de champ numérique « `:=` » est nécessaire dans chaque cellule où se trouve la commande `\multicolumn`. On comprend en effet que si ce marqueur n'existait pas, la totalite de la cellule, c'est-à-dire `\multicolumn{2}{|c|}{<formule>}` serait considérée comme étant une formule.

4 Macro-fonctions

L'extension `fp` fournit un nombre conséquent d'opérations et de fonctions. Malgré tout, dans le cadre d'un package comme spreadtab, celles-ci peuvent être insuffisantes et il est possible au programmeur averti d'écrire des macro-fonctions supplémentaires en utilisant celles fournies par `fp`. Cette section présente les macro-fonctions pour l'instant disponibles [10]. Il y aura plus de précisions sur la méthode pour programmer des macro-fonctions dans la prochaine version de ce manuel.

4.1 Macro-fonctions mathématiques

4.1.1 Sommer des cellules

La fonction « `sum` » permet de faire la somme d'une ou plusieurs plages de cellules.

Elle s'utilise ainsi : `sum(<plage 1>;<plage 2>;...;<plage n>)`, où une plage de cellules est :

- soit une cellule isolée comme « `a1` » ou « `[2,1]` » ;
- soit une zone rectangulaire délimitée par la cellule supérieure gauche et inférieure droite de cette façon : « `<cellule 1>:<cellule 2>` », à condition que « `<cellule 1>` » se trouve *avant* « `<cellule 2>` » lorsqu'on parcourt le tableau de haut en bas, de gauche à droite.
  Voici des exemple de plages de cellules : « `a2:d5` », « `[-1,-1]:[2,3]` », « `b4:[5,1]` ».

Dans les plages de cellules, les cellules vides ou ne contenant que du texte sont considérées comme contenant le nombre 0. Il en est de même pour les cellules masquée car fusionnées par `\multicolumn`.

10. Bien d'autres restent à écrire ! Elles seront disponibles dans des versions futures de spreadtab.

Les références relatives et absolues peuvent être utilisées conjointement, comme il semble bon à l'utilisateur. Voici un exemple où l'on fait la somme des coefficients du triangle de Pascal :

```
\begin{spreadtab}{{tabular}{*5c}}
\multicolumn{5}{c}{somme=:={sum(a2:e6)}}\\
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]  \\
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]       &        \\
[0,1] & [-1,1]+[0,1] & [-1,1]       &              &        \\
[0,1] & [-1,1]       &              &              &        \\
1     &              &              &              &
\end{spreadtab}
```

somme=31

1	4	6	4	1
1	3	3	1	
1	2	1		
1	1			
1				

4.1.2 Macro-fonction fact

La macro-fonction `fact(<nombre>)` permet de calculer la factorielle de son argument. Avec le moteur de calcul **fp** le nombre doit être un entier compris entre 0 et 18 inclus pour éviter des débordements. Le `<nombre>` peut aussi être une référence vers une cellule contenant un nombre entier.

Voici les factorielles de 0 à 8 :

```
\begin{spreadtab}{{tabular}{*9c}}
  0    &   1    &   2    &   3    &   4    &   5    &   6   &   7    &8\\\hline
fact(a1)&fact(b1)&fact(c1)&fact(d1)&fact(e1)&fact(f1)&fact(g1)&fact(h1)&fact(i1)
\end{spreadtab}
```

0	1	2	3	4	5	6	7	8
1	1	2	6	24	120	720	5040	40320

4.1.3 Macro-fonction sumprod

La fonction `sumprod` permet de multiplier les éléments correspondants de 2 plages ou plus et ensuite, additionner ces produits.

Cette fonction s'utilise ainsi : `sumprod(<plage 1>;<plage 2>;...;<plage n>)`. Toutes les plages rectangulaires doivent avoir les mêmes dimensions.

Voici un exemple simple où l'on calcule l'âge moyen d'un groupe d'enfants âgés de 10 à 15 ans :

```
\begin{spreadtab}{{tabular}{r*6c}}
@\^Ages & 10 & 11 & 12 & 13 & 14 & 15\\
@Nombre & 5  & 8  & 20 & 55 & 9  & 3\\\hline
@Moyenne&\multicolumn{6}{l}{:={sumprod(b1:g1;b2:g2)/sum(b2:g2)}}
\end{spreadtab}
```

Âges	10	11	12	13	14	15
Nombre	5	8	20	55	9	3
Moyenne	12,64					

De la même façon que pour la macro-fonction `sum`, les cellules de texte, vides ou fusionnées par un `\multicolumn` sont considérées comme contenant le nombre 0.

4.1.4 Nombres aléatoires

`randint` et `rand` renvoient un nombre aléatoire, respectivement entier entre bornes et décimal entre 0 et 1.

Nombres aléatoires avec le moteur fp Avec ce moteur, randint et rand sont des *macro fonctions* de spreadtab qui s'appuient sur la macro \FPrandom de **fp**.

À noter : la « graine » qui initialise la suite aléatoire dépend de la date ainsi que de la la minute à laquelle est faite la compilation. Les séries aléatoires données par cette fonction changeront entre deux compilations faites à des heures dont les minutes diffèrent. Si l'on veut vraiment avoir des nombres aléatoires indépendant du moment de compilation et donc reproductibles, il faut neutraliser la macro interne \ST@seed et définir une graine pour **fp** :

```
\makeatletter
\renewcommand\ST@seed{}% redefinit la macro interne
\makeatother
\FPseed=27% donne une graine (n'importe quel entier) a fp
```

La macro fonction randint([<nombre1>,]<nombre2>) renvoie une nombre *entier* qui dépend des paramètres : <nombre1> est un entier optionnel qui vaut 0 par défaut. L'entier aléatoire renvoyé est compris dans l'intervalle [<nombre1>;<nombre2>].

La macro fonction rand() renvoie un nombre aléatoire décimal compris entre 0 et 1 :

```
\STautoround{6}
\STusefp
\begin{spreadtab}{{tabular}{|l|cccc|}}\hline
@nombres dans [0;1]  &rand()        &rand()        &rand()        &rand()       \\
@nombres dans [-5;5] &randint(-5,5) &randint(-5,5) &randint(-5,5) &randint(-5,5)\\
@nombres dans [0;20] &randint(20)   &randint(20)   &randint(20)   &randint(20)  \\
\hline
\end{spreadtab}
\STusexfp
```

nombres dans [0;1]	0,290979	0,486181	0,241211	0,02513
nombres dans [-5;5]	-2	-3	-3	0
nombres dans [0;20]	4	19	9	9

Nombres aléatoires avec le moteur xfp Avec ce moteur, randint et rand sont des fonctions reconnues nativement par **xfp**. Un avertissement sera émis si XeLaTeX est utilisé dans une version trop ancienne qui n'implémente pas les primitives de génération de nombres pseudo-aléatoires ; dans ce cas, utiliser les fonctions randint et rand ne sera pas possible et conduira à des erreurs de compilation.

Comme le spécifie la documentation de **interface3**, la fonction randint(<a>,<b>) produit un entier aléatoire entre les deux arguments, mais si <a> est omis, randint(<b>) produit un entier entre **1** et <b> ce qui constitue un comportement différent de la macro fonction randint de spreadtab utilisée avec **fp** qui produit un entier entre **0** et <b>.

```
\STautoround{6}
\STusexfp
\begin{spreadtab}{{tabular}{|l|cccc|}}\hline
@nombres dans [0;1]  &rand()        &rand()        &rand()        &rand()       \\
@nombres dans [-5;5] &randint(-5,5) &randint(-5,5) &randint(-5,5) &randint(-5,5)\\
@nombres dans [0;20] &randint(20)   &randint(20)   &randint(20)   &randint(20)  \\
\hline
\end{spreadtab}
```

nombres dans [0;1]	0,193801	0,55877	0,158264	0,584902
nombres dans [-5;5]	3	2	5	1
nombres dans [1;20]	16	11	15	6

4.1.5 PGCD et PPCM

Les macros fonctions "gcd" et "lcm" permettent de calculer le Plus Grand Commun Diviseur (PGCD) et le Plus Petit Commun Multiple (PPCM) des nombres passés en arguments et séparés par des virgules :

```
gcd(nombre1,nombre2,...,nombreN)
lcm(nombre1,nombre2,...,nombreN)
```

Exemple :

```
\begin{spreadtab}{{tabular}{|r|r|r||c|c|}}\hline
\multicolumn3{|c||}{@Nombres}& @PGCD & @PPCM \\\hline
24 & 18 & 12 & \STcopy{v}{gcd(a2,b2,c2)} & \STcopy{v}{lcm(a2,b2,c2)}\\
15 & 10 & 25 & & \\
16 & 12 & 15 & & \\
\hline
\end{spreadtab}
```

Nombres			PGCD	PPCM
24	18	12	6	72
15	10	25	5	150
16	12	15	1	240

4.1.6 Écriture scientifique

La macro fonction "scitodec" permet de convertir un nombre écrit en écriture scientifique en nombre décimal. La syntaxe est `scitodec(<texte>)`, où le `<texte>` est :

- une suite de caractères se présentant sous la syntaxe `<mantisse>EE<exposant>`, où la `<mantisse>` est un nombre décimal et l'`>exposant>` est un entier relatif. Les "E" peuvent être écrit en majuscule ou minuscule.

  Le nombre ainsi représenté est `<mantisse>` $\times$ $10^{\texttt{<exposant>}}$
- une référence au champ *textuel* d'une cellule qui doit contenir des caractères obéissants à la syntaxe vue au point précédent.

Exemple :

```
\begin{spreadtab}{{tabular}{|r|r|}}\hlineÉ
@critures scientifiques & É@critures édcimales \\\hline
@4EE2                   & \STcopy{v}{scitodec([-1,0])}\\
@-3.1EE-3               & \\
@15ee5                  & \\
@-0.025ee7              & \\
@2.125EE0               & \\
@3.1575EE-4             & \\\hline
\end{spreadtab}
```

Écritures scientifiques	Écritures décimales
4EE2	400
-3.1EE-3	-0,0031
15ee5	1500000
-0.025ee7	-250000
2.125EE0	2,125
3.1575EE-4	0,00031575

Le moteur **xfp** est comprend nativement les nombres écrits en notation scientifique sous la forme `<a>e<b>` mais l'utilisation de cette syntaxe est *impossible* à utiliser avec spreadtab car le nombre `4e3`, qui est 4000, serait compris par spreadtab comme 4 suivi du contenu de la cellule `e3`.

4.1.7 Identité

La plus simple des macros fonctions est « id(<nombre>) ». Celle-ci renvoie le nombre qui se trouve entre parenthèses. Mathématiquement, elle ne sert pas à grand chose, mais avec spreadtab, elle permet de passer des expressions mathématiques à des macros qui n'en admettent pas. On peut ainsi s'en servir dans les plages de cellules de sum par exemple.

Dans le code ci dessous, la macro id est utilisée pour déterminer la plage de cellules à additionner avec sum. Dans cet exemple, la valeur du champ numérique de la cellule a2 vaut 8. Et donc, "sum([0,-1]:[id(a2-1),-1])/a2" est équivalent à sum([0,-1]:[7,-1])/8 :

```
\begin{spreadtab}{{tabular}{r*{10}c}}
                          @Entiers & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10\\
Moyenne des :={8} premiers entiers & sum([0,-1]:[id(a2-1),-1])/a2          \\
\end{spreadtab}
```

Entiers	1	2	3	4	5	6	7	8	9	10
Moyenne des 8 premiers entiers	4,5									

4.2 Macro-fonctions de test

Comme **fp** et sa macro \FPeval n'admet pas de test dans son argument, 3 macro-fonctions de test sont disponibles (elles sont moins intéressantes pour **xfp** qui dispose de l'opérateur « ?: », voir plus bas) :

ifeq(nombre1,nombre2,nombre3,nombre4)

ifgt(nombre1,nombre2,nombre3,nombre4

iflt(nombre1,nombre2,nombre3,nombre4)

La comparaison se fait entre nombre1 et nombre2 :

- test d'égalité pour ifeq : nombre1 = nombre2 ?
- test de supériorité stricte pour ifgt : nombre1 > nombre2 ?
- test d'infériorité stricte pour iflt : nombre1 < nombre2 ?

Si le test est vrai, nombre3 est retourné, sinon c'est nombre4.

À titre d'exemple, voici quelques valeurs de la fonction $f(x) = \begin{cases} 10 & \text{si } x < 1 \\ 0 & \text{si } x = 1 \\ -10 & \text{si } x > 1 \end{cases}$

```
\begin{spreadtab}{{tabular}{|*2c|}}\hline
@$x$       & @$f(x)$              \\\hline
-0.5       & iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\\
[0,-1]+0.5 & iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\\
[0,-1]+0.5 & iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\\
[0,-1]+0.5 & iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\\
[0,-1]+0.5 & iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\\
[0,-1]+0.5 & iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\\
[0,-1]+0.5 & iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\\\hline
\end{spreadtab}
```

x	$f(x)$
-0,5	10
0	10
0,5	10
1	0
1,5	-10
2	-10
2,5	-10

Le moteur **xfp** et son opérateur ternaire <a>?<b>:<c> rend les tests plus faciles dans les expressions évaluées : si le test <a> est vrai, <b> est retenu sinon, c'est <c>. Ainsi, l'imbrication de tests ci-dessus se programme :

```
\begin{spreadtab}{{tabular}{|*2c|}}\hline
@$x$       & @$f(x)$                \\\hline
-0.5       & [-1,0]<1 ? 10 : [-1,0]=1 ? 0 : -10\\
[0,-1]+0.5 & [-1,0]<1 ? 10 : [-1,0]=1 ? 0 : -10\\
[0,-1]+0.5 & [-1,0]<1 ? 10 : [-1,0]=1 ? 0 : -10\\
[0,-1]+0.5 & [-1,0]<1 ? 10 : [-1,0]=1 ? 0 : -10\\
[0,-1]+0.5 & [-1,0]<1 ? 10 : [-1,0]=1 ? 0 : -10\\
[0,-1]+0.5 & [-1,0]<1 ? 10 : [-1,0]=1 ? 0 : -10\\
[0,-1]+0.5 & [-1,0]<1 ? 10 : [-1,0]=1 ? 0 : -10\\\hline
\end{spreadtab}
```

x	$f(x)$
-0,5	10
0	10
0,5	10
1	0
1,5	-10
2	-10
2,5	-10

4.3 Macro-fonctions de date

4.3.1 Convertir une date en nombre avec `frshortdatetonum`

La macro `frshortdatetonum` permet de convertir une date de la forme 14/7/1789 en un nombre qui est en fait le nombre de jours écoulés depuis le 1er mars de l'an 0 [11]. Il est important de noter que cette macro-fonction requiert un argument *textuel* et non pas un nombre ou le résultat d'un calcul mathématique. Par conséquent, si l'argument de cette macro-fonction fait référence à une cellule, cette cellule *doit* être une cellule textuelle, c'est à dire contenant « `@` » ou « `:={}` »

Dans l'exemple ci-dessous, les deux premières lignes montrent que la cellule de gauche contient une date, et la cellule de droite fait référence à cette date pour calculer le nombre correspondant. La troisième ligne montre dans la cellule de gauche la date 0, mais surtout calcule dans la cellule de droite le nombre correspondant à la date *d'aujourd'hui*, en transformant en nombre les compteurs de TEX `\day`, `\month` et `\year` qui contiennent les numéros des jours, mois et année courants.

```
\begin{spreadtab}{{tabular}{cc}}
@14/7/1789              & frshortdatetonum(a1)\\
1/1/2001 :={}           & frshortdatetonum(a2)\\\hline
frshortdatetonum(1/3/0) & frshortdatetonum(\number\day/\number\month/\number\year)
\end{spreadtab}
```

14/7/1789	653554
1/1/2001	730791
0	737422

Une autre macro-fonction existe, elle transforme une date longue du type « 25 décembre 1789 » en un nombre. L'argument peut aussi être la séquence de contrôle `\today` si l'on a pris le soin de charger l'extension `babel` avec l'option `frenchb`.

11. Cet « an 0 » n'existe d'ailleurs pas, mais cela ne devrait pas être gênant pour les dates contemporaines

```
\begin{spreadtab}{{tabular}{cc}}
frlongdatetonum(\today) & frlongdatetonum(25 décembre 2009)\\
@1 juillet 1970         & frlongdatetonum(a2)
\end{spreadtab}
```

737422	734071
1 juillet 1970	719649

4.3.2 Passer d'un nombre à une date

Plusieurs macro-fonctions permettent de traduire un nombre en une donnée de date. Toutes ces macro-fonctions ont en commun que leur résultat est du *texte*. Par conséquent, *la cellule les contenant deviendra une cellule textuelle* dont le texte sera le résultat de cette fonction. Si un texte cohabitait avec la formule dans cette cellule, le résultat de la formule sera inséré à la place de `:={<formule>}`. Il en résulte que la cellule ne peut plus faire l'objet d'aucun traitement mathématique ensuite.

Voici ces fonctions :

- `numtofrshortdate` transforme un nombre en une date courte du type 14/7/1789 ;
- `numtofrlongdate` transforme un nombre en une date longue du type « 14 juillet 1789 » ;
- `numtofrmonth` extrait d'un nombre représentant une date le nom du mois correspondant ;
- `numtofrday` extrait d'un nombre représentant une date le nom du jour correspondant.

Voici un exemple où l'on se place 1000 jours avant puis 1000 jours après le 1/6/2009. Pour chacune de ces 2 dates, on calcule la date courte, la date longue, le mois et le jour de la semaine.

```
\begin{spreadtab}{{tabular}{cc}}                        \hline
\multicolumn{2}{|c|}{@1/6/2009}                         \\\hline\hline
1000      & numtofrshortdate(frshortdatetonum(a1)+[-1,0])\\
1000      & numtofrlongdate(frshortdatetonum(a1)+[-1,0]) \\
1000      & numtofrmonth(frshortdatetonum(a1)+[-1,0])    \\
1000      & numtofrday(frshortdatetonum(a1)+[-1,0])      \\\hline
-1000     & numtofrshortdate(frshortdatetonum(a1)+[-1,0])\\
-1000     & numtofrlongdate(frshortdatetonum(a1)+[-1,0]) \\
-1000     & numtofrmonth(frshortdatetonum(a1)+[-1,0])    \\
-1000     & numtofrday(frshortdatetonum(a1)+[-1,0])
\end{spreadtab}
```

1/6/2009	
1000	26/2/2012
1000	26 février 2012
1000	février
1000	dimanche
-1000	5/9/2006
-1000	5 septembre 2006
-1000	septembre
-1000	mardi

4.4 Macro fonctions de coordonnées

Plutôt que de faire référence à une cellule par ses coordonnées qui sont difficiles à mémoriser et qui changent si on insère une ligne ou une colonne, il est parfois bien plus pratique de donner un nom à une cellule et d'y faire référence plus loin par ce nom.

La macro fonction "`tag(<nom>)`" a pour effet de nommer la cellule dans laquelle elle se trouve. Il ne s'agit pas vraiment d'une macro fonction comme les autres puisque ce qu'elle retourne rien lorsqu'elle est mise dans une formule : elle disparait sans provoquer aucun effet sur le résultat mathématique. On peut donc écrire "`tag(<nom>)`" *n'importe où* dans le champ numérique d'une cellule. Le `<nom>` peut être n'importe quelle suite

de caractères alphanumériques, mais il est déconseillé de mettre une lettre et un nombre, qui pourraient être compris comme une référence à une cellule, et serait donc modifié lors d'une opération de copie avec \STcopy. Cette macro fonction a une action supplémentaire, elle sauvegarde via un \def la valeur numérique de la cellule dans laquelle elle se trouve de façon à pouvoir y faire appel plus tard *en dehors du tableau* via la commande purement développable \STtag{<nom>}.

Par la suite dans le tableau, au lieu de mettre les coordonnées de la cellule <nom>, on peut écrire "cell(<nom>)" qui est une macro fonction qui renvoit les coordonnées de la cellule précédemment nommée. Par exemple si "tag(<nom>)" a été écrit dans la cellule "B3" si on a écrit plus loin "cell(<nom>)", cette macro fonction renvoit B3.

Voici un exemple où l'on fait la somme de cellules et l'on donne le nom "foo" au premier nombre et le nom "bar" au dernier. On peut observer que tag(bar) se trouve entre "1" et "9" et qu'au final, puisque cette macro fonction disparait, le champ numérique de la cellule contiendra "19" :

```
\begin{spreadtab}{{tabular}{r@{}r}}
   & 15tag(foo)              \\
@+ & 37                      \\
@+ & 13                      \\
@+ & 48                      \\
@+ & 1tag(bar)9              \\\cline{2-2}
   & sum(cell(foo):cell(bar))tag(baz)
\end{spreadtab}

foo=\STtag{foo}, bar=\STtag{bar}, baz=\STtag{baz}
```

```
   15
+  37
+  13
+  48
+  19
  132
```

foo=15, bar=19, baz=132

Pour transmettre des valeurs entre tableaux calculés par spreadtab [12], il est possible de taguer la cellule dans le premier tableau à l'aide de la macro fonction tag(<nom>), puis dans le 2e tableau, de faire appel à la valeur *via* tagguée par valtag(<nom>)

```
\begin{spreadtab}{{tabular}{cc}}
    100 & a1+1tag{abcd}
\end{spreadtab}

\begin{spreadtab}{{tabular}{c}}
    value{abcd}
\end{spreadtab}
```

100 101
101

Lorsque l'environnement spreadtab se trouve inclus dans un autre environnement, les affectations faites par la macro fonction tag restent *locales* à cet environnement et ne pourraient pas être accessibles en dehors de cet environnement via \STtag. Dans l'exemple ci-dessous, le tableau fait par spreadtab est dans un environnement center et l'on doit donc utiliser \STmakegtag{<nom>} pour rendre global la sauvegarde de la valeur numérique contenus dans la cellule marquée par "tag<nom>" :

```
\begin{center}
  \begin{spreadtab}{{tabular}{cccc}}\hline
    6&9&17&21\\\hline
    \multicolumn{4}{c}{moyenne = :={sum(a1:d1)/4tag(moy)}}
  \end{spreadtab}
  \STmakegtag{moy}
\end{center}
La moyenne est de \STtag{moy}.
```

6	9	17	21
moyenne = 13,25			

La moyenne est de 13.25.

L'argument de \STmakegtag peut êre constitué de plusieurs noms séparés par des virgules.

Bien qu'elles soient à première vue moins utiles, spreadtab fournit également les macros fonctions "row(<nom>)" et "col(<nom>)" qui renvoient le numéro de la ligne ou de la colonne où se trouvait tag(<nom>). Voici par exemple une façon de dénombrer des valeurs pour en calculer la moyenne : on nomme "un" la première valeur et "der" la dernière, et le nombre de valeurs est donc row(der)-row(un)+1 :

12. Comme on l'avait demandé sur tex.stackexchange

```
moyenne = \begin{spreadtab}{{tabular}[b]{r}}
7tag(un)\\9\\15\\6\\20\\13\\11\\55tag(der)\\
\hline
sum(cell(un):cell(der))/(row(der)-row(un)+1)
\end{spreadtab}
```

	7
	9
	15
	6
	20
	13
	11
	55
moyenne =	17

5 Précautions particulières

5.1 Redéfinition des commandes de filets horizontaux

On peut être tenté de définir une commande pour produire — par exemple — une double ligne horizontale :

```
\newcommand\dline{\hline\hline}
```

puis essayer de l'utiliser dans un tableau pour produire ce simple exemple qui calcule à la seconde ligne les termes de la suite de Fibonacci :

0	1	2	3	4	5	6
1	1	2	3	5	8	13

Mais en écrivant ce code, il y a un problème :

```
\newcommand\dline{\hline\hline}
\begin{spreadtab}{{tabular}{*7c}}
0 & 1 & 2     & 3     & 4     & 5     & 6     \\\dline
1 & 1 & a2+b2 & b2+c2 & c2+d2 & d2+e2 & e2+f2
\end{spreadtab}
```

En effet la compilation échoue et dans le log, on peut lire que `\FPeval` se plaint :

```
! Improper alphabetic constant.
```

La raison est simple, c'est que le `\dline` de la ligne 4, n'est *pas* reconnu par spreadtab comme un filet horizontal et *il se retrouve donc dans la cellule de la ligne suivante*. Pour spreadtab, cette cellule `b1` contient :

```
\dline 1
```

Comme il n'y a pas de `@` ni de délimiteur de formule `:={...}`, `\FPeval` essaie vaillamment de calculer ce contenu et échoue évidemment !

Pour pouvoir compiler ce code sans erreur, la cellule `a2` *doit* contenir un marqueur de champ numérique :

```
\newcommand\dline{\hline\hline}
\begin{spreadtab}{{tabular}{*7c}}
0     & 1 & 2     & 3     & 4     & 5     & 6     \\\dline
:={1} & 1 & a2+b2 & b2+c2 & c2+d2 & d2+e2 & e2+f2
\end{spreadtab}
```

0	1	2	3	4	5	6
1	1	2	3	5	8	13

5.2 Cohabitation de \multicolumn et \SThidecol

Tout d'abord, dans une utilisation normale, l'utilisation conjointe de \multicolumn et \SThiderow ne doit pas arriver, et la plupart des utilisateurs ne devrait pas rencontrer cette situation ni lire ce chapitre.

Pour les courageux venons-en au cœur du problème : tout d'abord, une colonne masquée ne doit *jamais* contenir une cellule où se trouve la commande \multicolumn ! Mais que se passe t-il si une colonne masquée cache des cellules fusionnées par \multicolumn ?

Déjà, en général, il n'y a pas d'erreur de compilation ni message d'erreur, mais il y a quelques subtilités quant aux références qui sont un peu chamboulées dans la ligne concernée après le \multicolumn…

Prenons un exemple, et mettons que dans le tableau suivant, on fusionne les cellules b2 à h2 et que l'on souhaite cacher les colonnes c, d et f, ici en gris :

a1	b1	c1	d1	e1	f1	g1	h1	i1	j1
a2	b2							i2	j2

Il y a 4 cellules *visibles* fusionnées, on écrira donc \multicolumn{4} car on ne tient *jamais* compte des colonnes masquées dans le décompte du nombre de cellules à fusionner.

Maintenant, on compte 4 lettres à partir de la lettre b en l'incluant dans le décompte. On arrive à la lettre e : cela détermine un intervalle de colonnes « b–e ». Dans cet intervalle, 2 colonnes masquées sont incluses (c et d) et 1 colonne masquée n'est pas comprise (f). Ces 2 nombres sont importants pour comprendre la suite, aussi, notons-les x et y dans le cas général.

La règle est la suivante :

- il faut rajouter y signes « & » après le \multicolumn (pour l'exemple, il en faudrait 1).
- les références des colonnes des cellules qui suivent le \multicolumn seront décalées de x lettres vers le début de l'alphabet. Pour l'exemple donnée, si on veut faire référence à la cellule marquée « i2 », il faudra écrire g2 (au lieu de i2).

Voici un vrai exemple dont la structure est similaire au précédent : $x = 2$ et $y = 1$. Dans le code, remarquer le « & » qui a été ajouté puisque $y = 1$. Par ailleurs, on reste simple avec les formules, on ajoute 1 au nombre du dessus :

```
\begin{spreadtab}{{tabular}{|*{7}{c|}}}
\hline
1   & 2    & \SThidecol3 & \SThidecol4 & 5& \SThidecol6 & 7& 8& 9    & 10  \\\hline
a1+1& \multicolumn4{l|}{:={b1+1}}&                          & i1+1 & j1+1\\\hline
a2+1& b2+1 &             &             &  &             &  &  & g2+1 & h2+1\\\hline
\end{spreadtab}
```

1	2	5	7	8	9	10
2	3				10	11
3	4				11	12

Voici encore un exemple similaire où une seule colonne est masquée (la colonne d), et où $x = 1$ et $y = 0$:

```
\begin{spreadtab}{{tabular}{|*{9}{c|}}}
\hline
1   & 2    & 3 & \SThidecol4 & 5 & 6 & 7 & 8    & 9    & 10  \\\hline
a1+1& \multicolumn6{l|}{:={b1+1}}              & i1+1 & j1+1\\\hline
a2+1& b2+1 &   &             &   &   &   &      & h2+1 & i2+1\\\hline
\end{spreadtab}
```

1	2	3	5	6	7	8	9	10
2	3						10	11
3	4						11	12

5.3 Messages émis par spreadtab

Le package émet des messages d'erreur et arrête la compilation dans ces cas :

- Pour calculer une cellule, on calcule de proche en proche des cellules qui par le jeu des références, reviennent sur la cellule d'origine (références circulaires). Dans ce cas, l'arbre des dépendances est affiché dans le message d'erreur ;
- une formule nécessitant un nombre fait référence à une cellule vide ou ne contenant que du texte ;
- une cellule fait référence à une cellule non définie (hors limite du tableau) ;
- une cellule fait référence à une cellule fusionnée par un \multicolumn ;
- une référence relative entre crochets ne respecte pas la syntaxe.

Le package peut émettre des messages d'information (dans le fichier de log), ce qu'il fait par défaut. La commande \STmessage permet ou pas l'émission de messages d'information. Pour chacune des ces altenatives, la syntaxe est la suivante : \STmessage{true} ou \STmessage{false}.

Pour comprendre la signification des messages, prenons un tableau simple :

```
\begin{spreadtab}{{tabular}{|cccc|c|}}\hline
b1+1 & c1+1 & d1+1 & 10 & a1+b1+c1+d1\\\hline
\end{spreadtab}
```

13	12	11	10	46

Le fonctionnement du tableau est ici très simple à comprendre. Voici les informations délivrées par spreadtab :

```
[spreadtab] New spreadtab {tabular}{|cccc|c|}
* reading tab: ok
* computing formulas:
      cell A1-B1-C1
      cell B1
      cell C1
      cell D1
      cell E1
* building tab: ok
[spreadtab] End of spreadtab
```

L'environnement spécifié par l'utilisateur est repris entre parenthèses (ici {tabular}{|cccc|c|}). Précédées d'une étoile, on retrouve les 3 étapes nécessaires à spreadtab pour mener à bien sa mission : lecture du tableau, calcul des formules et construction du tableau final.

Pour la seconde étape, les cellules sont évaluées de haut en bas, de gauche à droite : spreadtab indique qu'il commence par essayer de calculer la première cellule A1. Pour cela, il indique qu'il doit d'abord évaluer B1 et avant cela encore, évaluer C1. Comme il n'y a plus de cellule après C1, c'est qu'elle peut être évaluée ; en effet, elle ne dépend que de D1 qui est un nombre égal à 10.

Pour chaque ligne suivante, il n'y a qu'une seule cellule ce qui signifie que lorsque spreadtab essaie de les évaluer, elles l'ont déjà été et sont des nombres ou alors, elles ne font références qu'à des cellules déjà calculées.

5.4 Débogage

Pour faciliter l'utilisation de spreadtab, un mode de débogage est disponible. Il est activé lorsque la commande \STdebug est présente dans l'argument optionnel de l'environnement spreadtab. Cette commande change le comportement de spreadtab qui, au lieu d'afficher le tableau final, affiche un (ou plusieurs) tableau de débogage. Cet affichage se fait juste après que spreadtab ait lu l'ensemble de toutes les cellules ; aucun calcul de formule n'a encore eu lieu. Il y a autant de tableaux affichés que de commande \STdebug présentes, sous réserve que leur argument soit différent. Seuls 3 arguments sont possibles, et voici ce qu'il permettent :

- \STdebug{formula} : affichage des champs numériques de toutes les cellules et les fins de ligne ;
- \STdebug{text} : affichage des champs textuels de toutes les cellules ;
- \STdebug{code} : affichage du code interne que spreadtab affecte à chaque cellule. Ce code interne est assigné à chaque cellule lors de la lecture du tableau. Il vaut :
  - −1 s'il s'agit d'une cellule fusionnée par une commande \multicolumn ;

- 0 si la cellule est vide ou purement textuelle ;
- 1 si le champ numérique de la cellule contient une formule qui sera à évaluer plus tard ;
- 2 si la champ numérique de la cellule contient un nombre.

Lorsque le mode débogage est activé, le tableau final *n'est pas affiché*. Il est cependant possible de forcer cet affichage en mettant la commande \STdisplaytab dans l'argument optionnel.

Voici un tableau sur lequel s'appuiera l'exemple suivant :

```
\begin{spreadtab}{{tabular}{|r|r|r|}}\hline
@$x$                &@@$y$                & @$x+y$\\\hline\hline
22                  & 54                  & \STcopy{v3}{a2+b2} \\
43                  & 65                  & \\
49                  & 37                  & \\\hline
$Sx=:={a2+a3+a4}$ & $Sy=:={b2+b3+b4}$  & $Sx+Sy=:={}$\\\hline
\multicolumn2{|r|}{$Sy-Sx=:={b5-a5}$}  & @\multicolumn1c{}\\\cline{1-2}
\end{spreadtab}
```

x	y	$x + y$
22	54	76
43	65	108
49	37	86
$Sx = 114$	$Sy = 156$	$Sx + Sy = 270$
	$Sy - Sx = 42$	

On va demander à ce que spreadtab affiche les 3 tableaux de débogage possibles concernant le tableau ci-dessus. Pour cela, il suffit de modifier la ligne 1 du code ci-dessus comme ceci :

```
\begin{spreadtab}[\STdebug{text}\STdebug{formula}\STdebug{code}]{{tabular}{|rr|r|}}\hline
```

	A	B	C
1	x	y	$x+y$
2	:=	:=	:=
3	:=	:=	:=
4	:=	:=	:=
5	$Sx=:=$	$Sy=:=$	$Sx+Sy=:=$
6	\multicolumn 2{|r|}{$Sy-Sx=:=$}		\multicolumn 1c{}

	A	B	C	\hline
1				\\\hline \hline
2	22	54	a2+b2	\\
3	43	65	a3+b3	\\
4	49	37	a4+b4	\\\hline
5	a2+a3+a4	b2+b3+b4	a5+b5	\\\hline
6	b5-a5			\\\cline {1-2}

	A	B	C
1	0	0	0
2	2	2	1
3	2	2	1
4	2	2	1
5	1	1	1
6	1	-1	0

Ces 3 tableaux de débogage peuvent aider à comprendre un peu mieux le fonctionnement interne de spreadtab. On peut observer dans le tableau 2 que toutes les cellules ayant un champ numérique ont un code interne de 1 ou 2 (voir tableau 3) et ont un marqueur de champ numérique ":=" qui leur est associé (voir tableau 1). Ce marqueur représente l'endroit où sera inséré — par substitution — le résultat du calcul du champ numérique. C'est donc à partir des contenus des champs textuels du tableau 1 et par simple substitution, qu'une fois les champs numériques calculés, les cellules sont reconstituées pour donner celles du tableau final.

Dans les tableaux ci-dessus, les cellules contenant les coordonnées ne sont grisées que si le package **colortbl** a été chargé.

6 Exemples

Voici quelques tableaux pour finir !

Afin que l'on sache quels nombres sont calculés, seuls les nombres non calculés sont en rouge. Dans ces tableaux, beaucoup d'artifices (des struts, des multicolumn) et des packages (notamment numprint et ses colonnes « N » qui alignent les séparateurs décimaux) ont été utilisés pour obtenir un résultat satisfaisant. Le code est parfois lourd et peu lisible, mais il ne s'agit pas ici d'exemples basiques mais de tableaux peaufinés !

6.1 Encore un triangle de Pascal

```
\begin{spreadtab}{{tabular}{*7r}}
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]\\
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]       & \\
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]       &              & \\
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]       &              &              & \\
[0,1] & [-1,1]+[0,1] & [-1,1]       &              &              &              & \\
[0,1] & [-1,1]       &              &              &              &              & \\
\color{red}:={1}&    &              &              &              &              &
\end{spreadtab}
```

1	6	15	20	15	6	1
1	5	10	10	5	1	
1	4	6	4	1		
1	3	3	1			
1	2	1				
1	1					
1						

6.2 Convergence d'une série

Pour les matheux, il s'agit du développement limité de la fonction exponentielle en 0,5. En effet,

$$\forall x \in \mathbf{R} \qquad e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

et le tableau illustre la rapidité de la convergence au fur et à mesure de l'ordre du développement limité.

```
\STautoround{15}
\renewcommand\STprintnum[1]{\numprint{#1}}
\begin{spreadtab}{{tabular}{cc}}
\multicolumn{2}{c}{Convergence en $x=\color{red}:={0.5}$}\\[1.5ex]
@$n$           & e^a1\SThidecol                & @ $\displaystyle e^{\numprint{<<a1>>}}-\sum_{k=0}^n\frac{\numprint{<<a1>>}^k}{k!}$\\[3ex]\hline
\color{red}:={0}& a1^[-1,0]/fact([-1,0])       & \STcopy{v}{b!2-b3}\\
\STcopy{v}{a3+1}& \STcopy{v}{a!1^a4/fact(a4)+b3}&                   \\
                &                              &                   \\
                &                              &                   \\
                &                              &                   \\
                &                              &                   \\
                &                              &                   \\
                &                              &                   \\
                &                              &                   \\
                &                              &                   \\\hline
\end{spreadtab}
```

Convergence en $x = 0{,}5$

n	$e^{0,5} - \sum_{k=0}^{n} \frac{0,5^k}{k!}$
0	0,648 721 270 700 128
1	0,148 721 270 700 128
2	0,023 721 270 700 128
3	0,002 887 937 366 795
4	0,000 283 770 700 128
5	0,000 023 354 033 461
6	0,000 001 652 644 572
7	0,000 000 102 545 366
8	0,000 000 005 664 166
9	0,000 000 000 281 877

6.3 Convergence vers le nombre d'or

Voici la définition des nombres de Fibonacci : $F_0 = 1 \qquad F_1 = 1 \qquad F_{n+2} = F_{n+1} + F_n$

On va mettre ici en évidence que le quotient de 2 nombres de Fibonacci consécutifs F_n et F_{n-1} tend vers le nombre d'or $\varphi = \frac{1+\sqrt{5}}{2}$ de telle sorte que la suite $u_n = \varphi - \frac{F_n}{F_{n-1}}$ soit alternativement positive et négative.

```
\STautoround{9}
$\begin{spreadtab}{{array}{ccN39N{3}{9}}}
@n              & @F_n              & @\hfill{\dfrac{F_n}{F_{n-1}}}\hfill\null& @\hfill{\varphi-\dfrac{F_n}{F_{n-1}}}\hfill\null\\[2ex]\hline
\color{red}:=1  & \color{red}:=1    &                   &                       \\
\STcopy{v}{a2+1} & \color{red}:=1    & \STcopy{v}{b3/b2} & (1+5^0.5)/2-[-1,0]    \\
                & \STcopy{v}{b2+b3} &                   & \STcopy{v}{d!3+1-c4}  \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\
                &                   &                   &                       \\\hline
\end{spreadtab}$
```

n	F_n	$\frac{F_n}{F_{n-1}}$	$\varphi - \frac{F_n}{F_{n-1}}$
1	1		
2	1	1	0,618 033 989
3	2	2	−0,381 966 011
4	3	1,5	0,118 033 989
5	5	1,666 666 667	−0,048 632 678
6	8	1,6	0,018 033 989
7	13	1,625	−0,006 966 011
8	21	1,615 384 615	0,002 649 374
9	34	1,619 047 619	−0,001 013 63
10	55	1,617 647 059	0,000 386 93
11	89	1,618 181 818	−0,000 147 829
12	144	1,617 977 528	0,000 056 461
13	233	1,618 055 556	−0,000 021 567
14	377	1,618 025 751	0,000 008 238
15	610	1,618 037 135	−0,000 003 146
16	987	1,618 032 787	0,000 001 202
17	1597	1,618 034 448	−0,000 000 459

6.4 Tableau de facturation

Voici un tableau de facturation, où les séparateurs décimaux sont alignés dans les colonnes grâce au spécificateur de colonne « N » du package `numprint`.

Ce tableau est généré par l'environnement `tabularx` de façon à occuper 80% de la largeur de la ligne. La commande `\multicolumn` a été largement utilisée pour la mise en forme :

```
\nprounddigits2
\let\PC\%
\newcommand\Mystrut{\rule[-1.2ex]{0pt}{4ex}}
\newcommand\RED{\color{red}}
\begin{spreadtab}{{tabularx}{0.8\linewidth}{|>\Mystrut X>\RED N42>\RED c N42>\RED c<\PC N42|}}
\hline
@D\'esignation &@\multicolumn1c{Prix U}& @\multicolumn1c{Qt\'e} & @\multicolumn1c{Prix} & @\multicolumn1c{R\'
    eduction} & @\textbf{\`A payer}\\\hline
@Item 1 & 5.99 & 20 & [-2,0]*[-1,0] & $-:={20}$ & [-2,0]*(1-[-1,0]/100)\\
@Item 2 & 12   & 7  & [-2,0]*[-1,0] & $-:={10}$ & [-2,0]*(1-[-1,0]/100)\\
@Item 3 & 4.50 & 40 & [-2,0]*[-1,0] & $-:={35}$ & [-2,0]*(1-[-1,0]/100)\\
@Item 4 & 650  & 2  & [-2,0]*[-1,0] & $-:={15}$ & [-2,0]*(1-[-1,0]/100)\\\hline
@\multicolumn6c{}\\[-1.5ex]\cline{4-6}% ligne vide et on remonte un peu !
@\multicolumn1c{\Mystrut}&@\multicolumn2{r|}{\textbf{Total}}& sum(d2:[0,-2]) & \multicolumn1c{$:={round
    (([1,0]/[-1,0]-1)*100,0)}\PC$} & {\fontseries{b}\selectfont}:={sum(f2:[0,-2])}\\
\cline{4-6}
\end{spreadtab}
```

Désignation	Prix U	Qté	Prix	Réduction	**À payer**
Item 1	5,99	20	119,80	−20%	95,84
Item 2	12,00	7	84,00	−10%	75,60
Item 3	4,50	40	180,00	−35%	117,00
Item 4	650,00	2	1 300,00	−15%	1 105,00
		Total	1 683,80	−17%	**1393,44**

6.5 Carré magique

```
\begin{spreadtab}{{tabular}{|*3{>{\hfill\rule[-0.4cm]{0pt}{1cm}$}m{0.7cm}<{$\hfill\null}|}}}
\hline
\color{red}:=2 & 5*b2-4*a1         & 3*a1-2*b2 \\\hline
2*a1-b2        & \color{red}:={-1} & 3*b2-2*a1 \\\hline
4*b2-3*a1      & 4*a1-3*b2         & 2*b2-a1   \\\hline
\end{spreadtab}
```

2	−13	8
5	−1	−7
−10	11	−4

6.6 Pyramide additive

Chaque nombre est la somme des deux nombres se trouvant au dessous de lui.

```
\newlength\cellsize
\setlength\cellsize{1.5cm}
\newcolumntype{K}{@{}>{\rule{0pt}{2.5ex}\centering\arraybackslash$}p{\cellsize}<$@{}}
\begin{spreadtab}{{tabular}{*{8}{@{}p{.5\cellsize}@{}}}}
\cline{4-5}
&&&\multicolumn{2}{|K|}{:={[-1,1]+[1,1]}}&&&\\\cline{3-6}
&&\multicolumn{2}{|K}{:={[-1,1]+[1,1]}}&\multicolumn{2}{|K|}{:={[-1,1]+[1,1]}}&&\\\cline{2-7}
&\multicolumn{2}{|K}{:={[-1,1]+[1,1]}}&\multicolumn{2}{|K}{:={[-1,1]+[1,1]}}&\multicolumn{2}{|K
    |}{:={[-1,1]+[1,1]}}&\\\hline
\multicolumn{2}{|K}{\color{red}:={-5}}&\multicolumn{2}{|K}{\color{red}:={3}}&\multicolumn{2}{|K}{\color{red
    }:={-2}}&\multicolumn{2}{|K|}{\color{red}:={-3}}\\\hline
\end{spreadtab}
```

			−5				
		−1		−4			
	−2		1		−5		
−5		3		−2		−3	

⋆

⋆ ⋆

C'est tout, j'espère que cette extension vous sera utile !

Je vous remercie d'avance de me signaler par **email** tout bug, toute macro-fonction à implémenter que vous pensez utile ou toute proposition d'amélioration *réaliste* : il ne faut pas oublier que cette extension doit rester modeste, que spreadtab n'est pas excel ou calc et qu'il est impossible d'implémenter toutes les fonctionnalités avancées de ces tableurs.

Christian Tellechea