

spreadtab

v0.2

Manuel de l'utilisateur

Christian TELLECHEA
unbonpetit@gmail.com

24 janvier 2010

Résumé

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

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	2
1.1	Présentation	2
1.2	Motivation	3
2	Fonctionnalités courantes	3
2.1	Références absolues	3
2.2	Références relatives	4
2.3	Des cellules de texte	4
2.4	Des cellules mixtes	4
2.5	Copier des formules	5
3	Mise en forme du tableau	7
3.1	Retours à la ligne et filets horizontaux	7
3.2	Masquer une ligne ou une colonne	8
3.3	Sauvegarder la valeur d'une cellule	8
3.4	Utiliser <code>\multicolumn</code>	9
3.5	Le package <code>fp</code>	10
3.6	Séparateur décimal	10
4	Macro-fonctions	11
4.1	Macro-fonctions mathématiques	11
4.1.1	Sommer des cellules	11
4.1.2	La factorielle	12
4.1.3	La macro-fonction <code>sumprod</code>	12
4.1.4	Nombres aléatoires	12
4.2	Les macro-fonctions de test	13
4.3	Les macro-fonctions de date	14
4.3.1	Convertir une date en nombre avec <code>frshortdateonum</code>	14
4.3.2	Passer d'un nombre à une date	14
5	Précautions particulières	15
5.1	Redéfinition des commandes de filets horizontaux	15
5.2	Cohabitation de <code>\multicolumn</code> et <code>\SThidecol</code>	16
5.3	Messages émis par <code>spreadtab</code>	17
5.4	Débogage	17
6	Des exemples	19
6.1	Encore un triangle de Pascal	19
6.2	La convergence d'une série	19
6.3	Convergence vers le nombre d'or	20
6.4	Un tableau de facturation	21
6.5	Un carré magique	21
6.6	Une pyramide additive	22

1 Introduction

1.1 Présentation

Cette extension permet de construire des tableaux similaire 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.

Ce package nécessite le moteur ϵ -TEX, le format L^AT_EX 2_ε ainsi que le package **fp** à qui sont confiés les calculs. Le package **xstring** est également requis dans une version *supérieure ou égale* à la version v1.5c [2009/06/05].

J'ai souhaité dès 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** de façon à ce qu'il agisse d'une façon *totale*ment 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** ;
- 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.

La syntaxe est la suivante :

```
1 \begin{spreadtab}{<nom de l'environnement><parametres de l'environnement>}
2     tableau avec formules et nombres
3 \end{spreadtab}
```

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

```
1 \begin{<nom de l'environnement><parametres de l'environnement>}
2     tableau avec nombres
3 \end{<nom de l'environnement>}
```

Même si disposer de fonctionnalités ressemblant à celles d'un tableur avec L^AT_EX est appréciable, il ne faut pas perdre de vue que les 3 étapes décrites ci-dessus prennent du temps et surtout que **fp** est lent dans ses calculs. 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 gênant¹, et la syntaxe de **spreadtab** constitue aussi un obstacle supplémentaire. L'avantage de cette extension est de pouvoir écrire *dans le code* L^AT_EX des tableaux comportant des calculs, alors que ces tableaux sont généralement exportés² d'une feuille de calcul d'un tableur vers le code L^AT_EX. 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. 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).

2. On peut signaler les 2 principaux programmes d'exportation : **cac12latex** pour « calc » de Open Office, et **excel12latex** pour « excel » de Microsoft Office.

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 Huu Dien Khue LE pour la traduction de ce manuel en viet-namien.

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 `frenchb` du package `babel`. Par conséquent, l'espace automatique inséré avant ces caractères sera désactivé dans un tableau.

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 1^{re} 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.

Une référence absolue s'écrit donc par exemple : « **b4** », « **C1** » ou « **d13** »³. 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 :

1	<code>\begin{spreadtab}{\tabular}{rr r}}</code>			
2	22	& 54	& a1+b1	\\
3	43	& 65	& a2+b2	\\
4	49	& 37	& a3+b3	\\
5	<code>\hline</code>			
6	a1+a2+a3	& b1+b2+b3	& a4+b4	
7	<code>\end{spreadtab}</code>			

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 :

1	<code>\begin{spreadtab}{\tabular}{ccccc}}</code>				
2	1	&	&	&	\\
3	a1	& a1	&	&	\\
4	a2	& a2+b2	& b2	&	\\
5	a3	& a3+b3	& b3+c3	& c3	\\
6	a2	& a4+b4	& b4+c4	& c4+d4	& d4
7	<code>\end{spreadtab}</code>				

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

3. 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.

2.2 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 :

```

1 $
2 \begin{spreadtab}{{matrix}}{{
3 1\\
4 [0,-1] & [-1,-1]\\
5 [0,-1] & [-1,-1]+[0,-1] & [-1,-1]\\
6 [0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]\\
7 [0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]
8 \end{spreadtab}
9 $

```

```

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.

2.3 Des 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 possible⁴ de faire référence nulle part ailleurs dans le tableau.

Voici un exemple :

```

1 \begin{spreadtab}{{tabular}}{{|r|ccc|}}
2 \hline
3 @ valeurs de $x$ & -5 & -1 & 4 \\
4 @ $f(x)=2x$ & 2*[0,-1] & 2*[0,-1] & 2*[0,-1] \\
5 \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{\celltext}` fera qu'une cellule contenant `\celltext` 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 Des 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 `fp` 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) ;

4. Il y a une exception à cette règle, voir la page 14.

- 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 :

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

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,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>`⁵. 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'elles 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

5. 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.

Le caractère « ! » est le développement de la séquence de contrôle `\Stttransposecar`. On peut donc modifier ce caractère en tout autre avec `\renewcommand\Stttransposecar{<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`.

Voici quelques exemples :	<code>{>3,v1}</code>	copie vers les 3 colonnes à droite et 1 ligne en dessous
	<code>{>3}</code>	copie vers les 3 cellules de droite
	<code>{v1}</code>	copie vers la cellule de dessous
	<code>{>}</code>	copie vers toutes les cellules situées à droite
	<code>{v}</code>	copie vers toutes les cellules situées en dessous
	<code>{v,>}</code>	copie vers le bas et vers la droite à partir de la cellule courante

[illegible]

\times	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

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	$\backslash\text{STcopy}\{v,>\}\{!a1+1\}$				
2	2					
3	3		$\backslash\text{STcopy}\{>2,v1\}\{!a3*10\}$			
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.

```

1 \begin{spreadtab}{\tabular{|*6{c|}}}\hline
2 1 & \STcopy{v,>}\{!a1+1\} & & & & \\\hline
3 2 & & & & & \\\hline
4 3 & & \STcopy{>2,v1}\{!a3*10\} & & & \\\hline
5 4 & & & & & \\\hline
6 5 & -1 & & a:=\{0\}b & & \\\hline
7 \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 :

```

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

```

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

3 Mise en forme du tableau

3.1 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` : `\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 :

```

1 \begin{spreadtab}{\tabular}{*5c}}
2 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1] \\ \noalign[1em]
3 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1] & \\
4 [0,1] & [-1,1]+[0,1] & [-1,1] & & \\
5 [0,1] & [-1,1] & & & \\
6 1 & & & & \\
7 \end{spreadtab}

```

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

3.2 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 :

```

1 \begin{spreadtab}{\tabular}{|r|ccc|}}
2 \hline
3 @ valeurs de $x$ & -1 & 0 & \SThidecol & 2 & 3 & \\ \hline
4 @$f(x)=2x-1$ & 2*[0,-1]-1 & 2*[0,-1]-1 & 2*[0,-1]-1 & 2*[0,-1]-1 & \\
5 @$g(x)=x-10$ \SThiderow & [0,-2]-10 & [0,-2]-10 & [0,-2]-10 & [0,-2]-10 & \\
6 @$h(x)=1-x$ & 1-[0,-3] & 1-[0,-3] & 1-[0,-3] & 1-[0,-3] & \\
7 \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

3.3 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`⁶, 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 :

```
1 \begin{spreadtab}[\STsavecell\result{c1}]{\tabular}{|c|c|c|c|c|}
2 \hline
3 10 & a1+10 & b1+10 & a1+b1+c1 & @cell c1 : \result\\ \hline
4 \end{spreadtab}
5 \par\medskip
6 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 :

```
1 \begin{spreadtab}[\STsavecell\hhh{b3}\STsavecell\mmm{c3}\STsavecell\sss{d3}]{\tabular}
2 {\rc|}{\hline
3 @Vitesse (km/h) & \SThidecol&\SThidecol&\SThidecol& 35 \\
3 @distance (km) & & & & 180\\ \hline
4 @Temps (h min s) & trunc(e2/e1,0) & trunc(60*(e2/e1-b3),0) & trunc(3600*(e2/e1-b3)
4 -60*c3,1) & @\hhh\ h \mmm\ min \sss\ s\\ \hline
5 \end{spreadtab}\par\medskip
6 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

3.4 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 :

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

```

1 \newcolumntype{K}[1]{@{}>{\centering\arraybackslash}p{#1cm}@{}}
2 \begin{spreadtab}{\tabular}{*6{K{0.5}}}}
3 \cline{2-5}
4 &\multicolumn{2}{|K{1}|}{:=\{a2*c2\}} & \multicolumn{2}{K{1}|}{:=\{c2*e2\}} & \\\hline
5 \multicolumn{2}{|K{1}|}{:=8}&\multicolumn{2}{|K{1}|}{:=7}&\multicolumn{2}{|K{1}|}{:=6}\\hline
6 \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`. En effet, si ce marqueur n'existait pas la totalité de la cellule, c'est-à-dire `\multicolumn{2}{|c|}{<formule>}` serait considérée comme étant une formule.

3.5 Le package fp

Comme cela a été précisé, tous les calculs sont faits par le package `fp` et sa macro `\FPeval`⁷. 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.

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.

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

```

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

3.6 Séparateur décimal

Le package `fp` renvoie ses 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` 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 :

7. 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 +` ».

`\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'un espace. Pour neutraliser cet espace, on peut mettre cette virgule entre accolades comme on le voit dans ce code :

<pre>1 3,14 n'est pas affiché comme \$3,14\$. \par 2 3,14 est affiché comme \$3{,}14\$</pre>	<p>3,14 n'est pas affiché comme 3, 14.</p> <p>3,14 est affiché comme 3,14</p>
--	---

Lorsque des cellules sont en mode math, on peut⁸ 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 :

<pre>1 \STsetdecimalsep{,} 2 \begin{spreadtab}{\tabular}{ *3{>{\$}r<{\$}} }\hline 3 @x & @y & @\text{Moyenne}\hline 4 5 & -4 & (a2+b2)/2\\ 5 -6.1 & -8 & (a3+b3)/2\\ 6 9.85 & 3.7 & (a4+b4)/2\\hline 7 \end{spreadtab}\par\smallskip 8 \STsetdecimalsep{,} 9 \begin{spreadtab}{\tabular}{ *3{>{\$}r<{\$}} }\hline 10 @x & @y & @\text{Moyenne}\hline 11 5 & -4 & (a2+b2)/2\\ 12 -6.1 & -8 & (a3+b3)/2\\ 13 9.85 & 3.7 & (a4+b4)/2\\hline 14 \end{spreadtab}</pre>	<table border="1" style="margin-bottom: 10px; text-align: center;"> <tr><th>x</th><th>y</th><th>Moyenne</th></tr> <tr><td>5</td><td>-4</td><td>0,5</td></tr> <tr><td>-6,1</td><td>-8</td><td>-7,05</td></tr> <tr><td>9,85</td><td>3,7</td><td>6,775</td></tr> </table> <table border="1" style="text-align: center;"> <tr><th>x</th><th>y</th><th>Moyenne</th></tr> <tr><td>5</td><td>-4</td><td>0,5</td></tr> <tr><td>-6,1</td><td>-8</td><td>-7,05</td></tr> <tr><td>9,85</td><td>3,7</td><td>6,775</td></tr> </table>	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
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																							

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 disponibles⁹. 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(<page 1>;<page 2>;...;<page 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`.

⁸. Il est cependant préférable d'utiliser le package `numprint` pour formater 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.

⁹. 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 bon semble à l'utilisateur. Voici un exemple où l'on fait la somme des coefficients du triangle de Pascal :

<pre> 1 \begin{spreadtab}{{\tabular}{*5c}} 2 \multicolumn{5}{c}{somme:=\sum(a2:e6)}}\\ 3 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1] & \\ 4 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1] & & \\ 5 [0,1] & [-1,1]+[0,1] & [-1,1] & & & \\ 6 [0,1] & [-1,1] & & & & \\ 7 1 & & & & & \\ 8 \end{spreadtab} </pre>	<div style="text-align: right;">somme=31</div> <table style="margin-left: auto;"> <tr><td>1</td><td>4</td><td>6</td><td>4</td><td>1</td></tr> <tr><td>1</td><td>3</td><td>3</td><td>1</td><td></td></tr> <tr><td>1</td><td>2</td><td>1</td><td></td><td></td></tr> <tr><td>1</td><td>1</td><td></td><td></td><td></td></tr> <tr><td>1</td><td></td><td></td><td></td><td></td></tr> </table>	1	4	6	4	1	1	3	3	1		1	2	1			1	1				1				
1	4	6	4	1																						
1	3	3	1																							
1	2	1																								
1	1																									
1																										

4.1.2 La factorielle

La macro-fonction **fact**(<nombre>) permet de calculer la factorielle de son argument, sous réserve que celui-ci soit un entier compris entre 0 et 18 inclus pour éviter des débordements¹⁰. Le <nombre> peut aussi être une référence vers une cellule contenant un nombre entier.

Voici les factorielles de 0 à 8 :

```

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

```

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

4.1.3 La 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 :

```

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

Les macro-fonctions **randint** et **rand** renvoient un nombre aléatoire.

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 faite 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** :

¹⁰. En effet, pour **fp** le plus grand entier est $10^{18} - 1$. La factorielle de 19 dépasse ce nombre.

```

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

```

La macro fonction **randint** renvoie un nombre *entier* qui dépend des paramètres. La syntaxe est la suivante : **randint**([<nombre1>,<nombre2>], où <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 :

```

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

```

nombres dans [0;1]	0,442263	0,10999	0,600691	0,810668
nombres dans [-5;5]	4	-2	4	-4
nombres dans [0;20]	2	12	4	13

4.2 Les macro-fonctions de test

Elles sont au nombre de 3 et ont la syntaxe suivante :

```

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 $f(x) = \begin{cases} 10 & \text{si } x < 1 \\ 0 & \text{si } x = 1 \\ -10 & \text{si } x > 1 \end{cases}$

```

1 \begin{spreadtab}{\tabular}{|*2c|}\hline
2 @x$ & @f(x)$ & \\
3 -0.5 & & \bftt{iflt}([-1,0],1,10,\bftt{ifeq}([-1,0],1,0,-10))\\
4 [0,-1]+0.5 & & \bftt{iflt}([-1,0],1,10,\bftt{ifeq}([-1,0],1,0,-10))\\
5 [0,-1]+0.5 & & \bftt{iflt}([-1,0],1,10,\bftt{ifeq}([-1,0],1,0,-10))\\
6 [0,-1]+0.5 & & \bftt{iflt}([-1,0],1,10,\bftt{ifeq}([-1,0],1,0,-10))\\
7 [0,-1]+0.5 & & \bftt{iflt}([-1,0],1,10,\bftt{ifeq}([-1,0],1,0,-10))\\
8 [0,-1]+0.5 & & \bftt{iflt}([-1,0],1,10,\bftt{ifeq}([-1,0],1,0,-10))\\
9 [0,-1]+0.5 & & \bftt{iflt}([-1,0],1,10,\bftt{ifeq}([-1,0],1,0,-10))\\
10 \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 Les macro-fonctions de date

4.3.1 Convertir une date en nombre avec `frshortdatetotnum`

La macro `frshortdatetotnum` 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 1^{er} mars de l'an 0¹¹. 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 T_EX `\day`, `\month` et `\year` qui contiennent les numéros des jours, mois et année courants.

```

1 \begin{spreadtab}{{tabular}{cc}}
2 @14/7/1789 & frshortdatetotnum(a1)\\
3 1/1/2001 :={} & frshortdatetotnum(a2)\\ \hline
4 frshortdatetotnum(1/3/0) & frshortdatetotnum(\number\day/\number\month/\number\year)
5 \end{spreadtab}

```

14/7/1789	653554
1/1/2001	730791
0	734101

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`.

```

1 \begin{spreadtab}{{tabular}{cc}}
2 frlongdatetotnum(\today) & frlongdatetotnum(25 décembre 2009)\\
3 @1 juillet 1970 & frlongdatetotnum(a2)
4 \end{spreadtab}

```

734101	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, il sera écrasé par le résultat. 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.

```

1 \begin{spreadtab}{{tabular}{cc}} & \hline
2 \multicolumn{2}{@1/6/2009} & \\ \hline
3 1000 & numtofrshortdate(frshortdatetotnum(a1)+[-1,0])\\
4 1000 & numtofrlongdate(frshortdatetotnum(a1)+[-1,0]) \\

```

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

```

5 1000 & numtofrmonth(frshortdatetinum(a1)+[-1,0]) \\
6 1000 & numtofrday(frshortdatetinum(a1)+[-1,0]) \\ \hline
7 -1000 & numtofrshortdate(frshortdatetinum(a1)+[-1,0]) \\
8 -1000 & numtofrlongdate(frshortdatetinum(a1)+[-1,0]) \\
9 -1000 & numtofrmonth(frshortdatetinum(a1)+[-1,0]) \\
10 -1000 & numtofrday(frshortdatetinum(a1)+[-1,0]) \\
11 \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

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 :

```

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

```

1 \newcommand\dline{\hline\hline}
2 \begin{spreadtab}{{\tabular}{*7c}}
3 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \dline
4 :=\{1\} & 1 & a2+b2 & b2+c2 & c2+d2 & d2+e2 & e2+f2
5 \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 *jamaïs* 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 *jamaïs* 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 `a` et `b` dans le cas général.

La règle est la suivante :

- il faut rajouter `b` signes « `&` » après le `\multicolumn` (pour l'exemple, il en faudrait 1).
- les références des colonnes des cellules qui suivent le `\multicolumn` seront décalées de `a` 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 : `a = 2` et `b = 1`. Dans le code, remarquer le « `&` » qui a été ajouté puisque `b = 1`. Par ailleurs, on reste simple avec les formules, on ajoute 1 au nombre du dessus :

```

1 \begin{spreadtab}{\tabular}{|*{7}{c|}}
2 \hline
3 1 & 2 & & \SThidecol3 & \SThidecol4 & 5 & \SThidecol6 & 7 & 8 & 9 & & 10 & \\ \hline
4 a1+1 & \multicolumn{4}{|}{:= {b1+1}} & & & & & i1+1 & & j1+1 & \\ \hline
5 a2+1 & b2+1 & & & & & & & & g2+1 & & h2+1 & \\ \hline
6 \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 `a = 1` et `b = 0` :

```

1 \begin{spreadtab}{\tabular}{|*{9}{c|}}
2 \hline
3 1 & 2 & & 3 & \SThidecol4 & 5 & 6 & 7 & 8 & & 9 & & 10 & \\ \hline
4 a1+1 & \multicolumn{6}{|}{:= {b1+1}} & & & & i1+1 & & j1+1 & \\ \hline
5 a2+1 & b2+1 & & & & & & & & & h2+1 & & i2+1 & \\ \hline
6 \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 alternatives, la syntaxe est la suivante : `\STmessage{true}` ou `\STmessage{false}`.

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

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

13	12	11	10	46
----	----	----	----	----

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

```
1 [spreadtab] New spreadtab {\tabular}{|cccc|c|}
2 * reading tab: ok
3 * computing formulas:
4   cell A1-B1-C1
5   cell B1
6   cell C1
7   cell D1
8   cell E1
9 * building tab: ok
10 [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 :

```

1 \begin{spreadtab}{{\tabular}{|r|r|r|}}\hline
2 @x$ & @y$ & @x+y$\\ \hline \hline
3 22 & 54 & \STcopy{v3}{a2+b2} \\
4 43 & 65 & \\
5 49 & 37 & \\ \hline
6 $Sx:={a2+a3+a4}$ & $Sy:={b2+b3+b4}$ & $Sx+Sy:={}$\\ \hline
7 \multicolumn2{|r|}{$Sy-Sx:={b5-a5}$} & @\multicolumn1c{}\\ \cline{1-2}
8 \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}{|r|r|}\hline
```

	A	B	C
1	x	y	$x+y$
2	$:=$	$:=$	$:=$
3	$:=$	$:=$	$:=$
4	$:=$	$:=$	$:=$
5	$Sx:={a2+a3+a4}$	$Sy:={b2+b3+b4}$	$Sx+Sy:={}$
6	$Sy-Sx:={b5-a5}$		

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

	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 Des 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 ! Par souci de lisibilité, la commande `\STcopy` n'a pas été utilisée.

6.1 Encore un triangle de Pascal

```

1 \begin{spreadtab}{{\tabular}{*7r}}
2 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] &
   [-1,1] \\
3 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1] & \\
4 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1] & & \\
5 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1] & & & \\
6 [0,1] & [-1,1]+[0,1] & [-1,1] & & & & \\
7 [0,1] & [-1,1] & & & & & \\
8 \color{red}:=1 & & & & & & \\
9 \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 La convergence d'une série

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

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

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

```

1 \STautoround{15}
2 \begin{spreadtab}{{\STsavecell\xvalue{a1}}{\tabular}{cN{2}{15}}}}
3 \multicolumn{2}{c}{Convergence en $x=\color{red}{:=\{0.5\}}$}\\[1.5ex]
4 @n$ & e^{a1} \SThidecol & \hfill{\@ $\displaystyle e^{\backslash
   numprint\xvalue}-\sum_{k=0}^n \frac{\numprint\xvalue^k}{k!}$} \hfill\\[3ex] \hline
5 \color{red}:=0 & a1^{[-1,0]}/fact([-1,0]) & b2-[-1,0] \\
6 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
7 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
8 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
9 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
10 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
11 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
12 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
13 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
14 [0,-1]+1 & a1^{[-1,0]}/fact([-1,0])+[0,-1] & b2-[-1,0] \\
15 \end{spreadtab}

```

Convergence en $x = 0,5$

$$n \quad e^{0,5} - \sum_{k=0}^n \frac{0,5^k}{k!}$$

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 : $F_0 = 1 \quad F_1 = 1 \quad F_{n+2} = F_{n+1} + F_n$

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.

```

1 \STautoround{9}
2 $
3 \begin{spreadtab}{{matrix}}{}
4 @n      & @F_n      & @\dfrac{F_n}{F_{n-1}} & @\varphi-\dfrac{F_n}{F_{n-1}}
5 -1}}\hline
6 \color{red}:1 & \color{red}:1 & & \\\
7 [0,-1]+1 & \color{red}:1 & [-1,0]/[-1,-1] & (1+5^0.5)/2-[-1,0] \\\
8 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
9 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
10 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
11 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
12 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
13 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
14 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
15 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
16 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
17 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
18 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
19 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
20 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
21 [0,-1]+1 & [0,-1]+[0,-2] & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\hline
22 \end{spreadtab}
23 $

```

n	F_n	$\frac{F_n}{F_{n-1}}$	$\varphi - \frac{F_n}{F_{n-1}}$
1	1		
2	1	1	0,618033989
3	2	2	-0,381966011
4	3	1,5	0,118033989
5	5	1,666666667	-0,048632678
6	8	1,6	0,018033989
7	13	1,625	-0,006966011
8	21	1,615384615	0,002649374
9	34	1,619047619	-0,00101363
10	55	1,617647059	0,00038693
11	89	1,618181818	-0,000147829
12	144	1,617977528	0,000056461
13	233	1,618055556	-0,000021567
14	377	1,618025751	0,000008238
15	610	1,618037135	-0,000003146
16	987	1,618032787	0,000001202
17	1597	1,618034448	-0,000000459

6.4 Un 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 :

```

1 \nprouddigits2
2 \let\PC\%
3 \begin{spreadtab}{\tabularx{0.8\linewidth}{|>\rule[-1.2ex]{0pt}{4ex}}X>{\color{red}
  }}N42>{\color{red}}cN42>{\color{red}}cN42|}}
4 \hline
5 @D\'esignation & @\multicolumn{1}{c}{Prix U} & @\multicolumn{1}{c}{Qté} & @\
  \multicolumn{1}{c}{Prix} & @\multicolumn{1}{c}{R\'eduction} & @\textbf{À payer}
  \\ \hline
6 @Item 1 & 5.99 & 20 & [-2,0]*[-1,0] & $-:=\{20\}\PC$ & [-2,0]*(1-[-1,0]/100)\\
7 @Item 2 & 12 & 7 & [-2,0]*[-1,0] & $-:=\{10\}\PC$ & [-2,0]*(1-[-1,0]/100)\\
8 @Item 3 & 4.50 & 40 & [-2,0]*[-1,0] & $-:=\{35\}\PC$ & [-2,0]*(1-[-1,0]/100)\\
9 @Item 4 & 650 & 2 & [-2,0]*[-1,0] & $-:=\{15\}\PC$ & [-2,0]*(1-[-1,0]/100)\\ \hline
10 @\multicolumn{6}{c}{\ccline{4-6}% ligne vide et on remonte un peu !}
11 @\multicolumn{1}{c}{\rule[-1.2ex]{0pt}{4ex}} & @\multicolumn{2}{c}{\textbf{Total}} & @
  \sum(d2:[0,-2]) & @\multicolumn{1}{c}{\textbf{\sum(f2:[0,-2])}} & @
  \fontseries{b}\selectfont
12 \ccline{4-6}
13 \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%	1 393,44

6.5 Un carré magique

```

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

```

2	-13	8
5	-1	-7
-10	11	-4

6.6 Une pyramide additive

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

```

1 \newlength\cellsize
2 \setlength\cellsize{1.5cm}
3 \newcolumntype{K}{\rule{0pt}{2.5ex}\centering\arraybackslash$}p{\cellsize}<${}
4 \begin{spreadtab}{\tabular}{*{8}{@{}p{.5\cellsize}@{}}}}
5 \cline{4-5}
6 &&&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&&&\cline{3-6}
7 &&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&&\cline{2-7}
8 &\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&\hline
9 \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
10 \end{spreadtab}

```

		-5		
	-1		-4	
-2		1		-5
-5	3	-2	-3	

★
★ ★

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

Compte tenu de la jeunesse de ce package, soyez indulgent quant aux dysfonctionnements rencontrés : avec les remarques des utilisateurs, des versions plus abouties verront le jour. En effet, cette version n'est qu'un début et l'ensemble, quoique plutôt stable, reste encore imparfait. Il reste notamment beaucoup de macro-fonctions à écrire.

Je vous remercie donc 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 tableaux.

Christian TELLECHEA