spreadtab

v0.6

Manuel de l'utilisateur

Christian Tellechea
unbonpetit@netc.fr

27 février 2025

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 référence 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 Présentation

Ce manuel décrit l'extension spreadtab. Elle permet de construire des tableaux similaires à des feuilles de calcul. 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

```
\usepackage{spreadtab}
```

Ce package nécessite le format LaTeX 2_ε postérieur au 2022/06/01 ainsi que les packages `xstring` et `simplekv`.

Ce package est censé être compatible avec *tous* les environnements de tableaux, sous réserve que les séparateurs entre colonnes soient « & » et les retours à la ligne soient « \\ ». spreadtab agit de 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. Si la macro **`\STxp`** est présente une ou plusieurs fois, son argument est développé au maximum. Le corps du tableau est ensuite divisé en lignes puis en cellules ; *New v0.6*
- le calcul des formules contenues dans les cellules est ensuite effectué, en ayant pris soin pour chacune de calculer auparavant les cellules dépendantes ;
- le corps du tableau est enfin reconstruit en ayant remplacé chaque formule par la valeur numérique préalablement calculée et le tout est donné à l'environnement tableau spécifié par l'utilisateur.

La syntaxe est :

```
\begin{spreadtab}[⟨options⟩]{{⟨tabname⟩}{⟨colspec⟩}}
    tableau avec formules et nombres
\end{spreadtab}
```

où ⟨*tabname*⟩ représente le nom de n'importe quel environnement de type tableau disponible avec LaTeX ou avec un package. Il est possible de remplacer `\begin{spreadtab}...\end{spreadtab}` par `\spreadtab...\endspreadtab`.

Les ⟨*options*⟩ doivent se présenter sous la forme `clé` = ⟨*valeur*⟩ pour ne s'appliquer qu'au tableau en cours. Pour spécifier des ⟨*options*⟩ valables pour tous les tableaux à venir, on utilise *New v0.6*

```
\STset{⟨clé⟩=⟨valeur⟩}
```

La liste des `clés` et des ⟨*valeurs*⟩ associées est disponible à la page 17.

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.

2 Nouveau dans la version 0.6

Cette version est susceptible de casser l'utilisation selon les versions précédentes en raison de la suppression de certaines macros et d'un changement dans l'argument optionnel. Il est donc important de prêter attention aux points qui suivent, surtout si des erreurs de compilation se produisent : *New v0.6*

1. `l3fp` est désormais le seul moteur de calcul possible et donc, il est obligatoire d'utiliser une version de LaTeX postérieure à l'intégration de `xfp` dans le noyau le 2022/06/01 ;
2. le support du vieux moteur de calcul `fp` est définitivement abandonné ;
3. lors de l'appel du package par `\usepackage{spreadtab}`, aucun argument optionnel n'est désormais possible puisqu'il n'y a qu'un moteur de calcul ;
4. l'environnement agit désormais dans un groupe semi simple ;
5. Le package `simplekv` est chargé pour utiliser le système de `clé` = ⟨*valeur*⟩. Tous les réglages de spreadtab se font désormais avec la macro `\STset{clé` = ⟨*valeur*⟩`}` ;
6. l'argument optionnel qui suit `\begin{spreadtab}[`⟨*options*⟩`]` contient désormais **exclusivement** les clés/valeurs que l'on souhaite appliquer au tableau en cours et donc :
   - la macro `\STsavecell` est supprimée et remplacée par la clé `save list` (voir page 8)

— les macros `\STdebug` et `\STdisplaytab` sont supprimées et remplacés par la clé `debug` (voir page 16)

Ce point va provoquer des erreurs de compilation si l'argument optionnel contient les macros ci-dessus comme c'était le cas dans les versions précédentes.

7. si `\STxp` est présent dans le corps du tableau une ou plusieurs fois, le développement maximal de son argument est effectué pendant la lecture du tableau ;
8. suppression de la macro `\STmakegtag`, les assignations par la macro-fonction `tag` sont toujours globales à cause du point 4 ;
9. la macro-fonction `tag` permet, si la clé `tag to aux =` ⟨*true*⟩, d'écrire les assignations dans le fichier auxiliaire ;
10. la clé `aux save list` permet de sauvegarder le contenu de cellules choisies par l'utilisateur dans des macros et écrire les assignations dans le fichier auxiliaire ;
11. appartenant aux anciennes versions, les macros suivantes `\STautoround`, `\STsetdecimalsep`, `\STeol`, `\STsetdisplaymarks`, `\STmessage`, `\STnumericfieldmarker`, `\STtranposechar` et `\STtextcell`, bien que fonctionnant encore (mais plus documentées), seront supprimées à la prochaine version ;
12. les macro-fonctions `ifeq`, `ifgt` et `iflt` seront supprimées à la prochaine version.

3 Fonctionnalités courantes

3.1 Calcul

Une cellule peut désormais contenir tout calcul compris par `l3fp` : opérations courantes, trigonométrie et inverses, exponentielles et logarithmes, comparaisons, connecteurs logiques [1], factorielle, nombres aléatoires, etc. Voir la documentation de `interface3`.

Par conséquent, les macro-fonctions `rand`, `rnd` et `fact` sont toujours utilisables, mais ne sont *plus* des macro-fonctions de spreadtab : elles sont directement comprises par `l3fp`.

3.2 Fins de ligne

Par défaut, spreadtab considère qu'une ligne se termine avec `\\`, ce qui est habituel dans les tableaux. L'éventuel argument optionnel `\\[`⟨*dimension*⟩`]` qui le suit est pris en compte. Ce marqueur de fin de ligne peut être modifié par la clé `tabline sep`. On peut par exemple écrire

```
tabline sep = \tabularnewline
```

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.

Juste après le retour à la ligne, spreadtab reconnaît la plupart des filets horizontaux ainsi que leur éventuel argument : `\hline`, `\cline`, `\hhline`, `\noalign`, `\toprule`, `\midrule`, `\bottomrule`, `\cmidrule`, `\addlinespace`, `\morecmidrules` et `\specialrule`.

Pour d'autres cas, voir page 14 comment procéder.

3.3 Références absolues

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

— la colonne est une lettre, majuscule ou minuscule, de `a` à `z` où `a` représente la 1re colonne de gauche : on est donc d'emblée limité à 26 colonnes, ce qui devrait suffire pour la grande majorité des cas ;

— 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.

1. Si l'on souhaite utiliser le connecteur logique « et » de `l3fp` 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` ».

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

Dans cet exemple, 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

Dans cet autre exemple, 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

3.4 Références relatives

Pour faire référence à une cellule, on peut 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\\                                                                  1
    [0,-1] & [-1,-1]\\                                                  1   1
    [0,-1] & [-1,-1]+[0,-1] & [-1,-1]\\                                 1   2   1
    [0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]\\                1   3   3   1
    [0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]  1   4   6   4   1
\end{spreadtab}
$
```

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

3.5 Cellules de texte

Si l'on veut mettre du texte dans une cellule, 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[2] de faire référence nulle part ailleurs dans le tableau.

La clé `text mark` permet de modifier le marqueur de texte. Par exemple

```
\STset{text mark=`}
```

2. Il y a une exception à cette règle, voir la page 12.

fera qu'une cellule contenant le caractère « ` » sera comprise comme étant une cellule de texte. Il est cependant important de noter que la clé `text mark` détokenize sa valeur, ce qui rend le marqueur de texte constitué de token(s) de catcode 12 : pour être reconnus par spreadtab, ces tokens doivent avoir ce catcode 12 au moment de la lecture du tableau.

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

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

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

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

3.6 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 ;
- 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*⟩`}`⟨*champ textuel*⟩

  Avec la clé `numeric mark`, on peut changer ce marqueur en « `\=` » avec :

  ```
  numeric mark={\=}
  ```

  Dans ce cas, la définition 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.

Pour fixer les idées, voici un exemple très simple :

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

val 1 : 50	val 2 : 29	moy : **39.5**

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 « `@` ».

3.7 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 la 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*⟩[3]. La commande `\STcopy` *ne doit pas* se trouver dans une cellule où un marqueur de champ numérique « `:=` » est présent.

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. 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. Le caractère « ! » peut être changé avec `freeze char =` ⟨*caractère*⟩. Il faut noter que le ⟨*caractère*⟩ est détokénisé.

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.

On peut ainsi 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} &                      &                   & & & & & & & \\
                     &                      &                   & & & & & & & \\
                     &                      &                   & & & & & & & \\
                     &                      &                   & & & & & & & \\
                     &                      &                   & & & & & & & \\
                     &                      &                   & & & & & & & \\
                     &                      &                   & & & & & & & \\
                     &                      &                   & & & & & & & \\
                     &                      &                   & & & & & & & \\\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.

4 Mise en forme du tableau

4.1 Codes exécutés avant ou après le tableau

Les clés `pretab code =` ⟨*code*⟩ et `posttab code =` ⟨*code*⟩ définissent des codes arbitraires exécutés juste avant et juste après l'affichage du tableau. *New v0.6*

```
Une addition
\begin{spreadtab}[pretab code=\begingroup\color{red},
                  posttab code=\endgroup]{{tabular}[t]{r}}
    149\\
    287\\\hline
    a1+a2
\end{spreadtab} posée.
```

Une addition
149
287
436
posée.

3. 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.

4.2 Séparateur décimal

Comme `l3fp` donne 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 en tenaient compte.

La clé `dec sep` = ⟨*char*⟩ permet de changer le séparateur décimal la virgule, mais ce choix n'est pas neutre dans le mode math. En effet, elle est considérée comme une ponctuation ce qui explique qu'elle est suivie d'une espace.

Pour éviter cette faute typographique, on peut

- définir le séparateur décimal entre accolades avec `dec sep={{,}}`
- exécuter le code `\mathcode`\,="013B\relax` avant d'afficher le tableau

Le plus élégant étant d'utiliser un package configurable et spécialisé dans l'affichage des nombres comme `siunitx`.

```
\begin{spreadtab}[dec sep={,}]{{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
\begin{spreadtab}[dec sep={{,}}]{{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
\begin{spreadtab}[dec sep     ={,},
                  pretab code ={\begingroup\mathcode`\,="013B\relax},
                  posttab code={\endgroup}]{{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

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

4.3 Arrondi automatique

Le moteur `l3fp` donne jusqu'à 16 décimales significatives. Si l'on ne souhaite pas autant de précision, on peut arrondir automatiquement le résultat de chaque calcul avec la clé `autoround`. Agir sur cette clé implique donc une *limitation délibérée* de la précision des calculs internes.

```
\begin{spreadtab}{{tabular}{*4c}}       \hline
    @$x$           & 3               & 17 & 751 \\\hline
    @$1/x$         & \STcopy>{1/b1}  &    &     \\\hline
    @$x\cdot(1/x)$ & \STcopy>{b2*b1} &    &     \\\hline
\end{spreadtab}
\bigbreak
\begin{spreadtab}[autoround=6]{{tabular}{*4c}}    \hline
    @$x$           & 3               & 17 & 751 \\\hline
    @$1/x$         & \STcopy>{1/b1}  &    &     \\\hline
    @$x\cdot(1/x)$ & \STcopy>{b2*b1} &    &     \\\hline
\end{spreadtab}
```

x	3	17	751
$1/x$	0.3333333333333333	0.05882352941176471	0.00133155792276964
$x \cdot (1/x)$	0.9999999999999999	1	0.9999999999999996

x	3	17	751
$1/x$	0.333333	0.058824	0.001332
$x \cdot (1/x)$	0.999999	1.000008	1.000332

Pour revenir au comportement par défaut et ne faire aucun arrondi, il faut rendre la valeur vide :

```
autoround={}
```

4.4 Macro \STprintnum

Chaque nombre traité par spreadtab est donné à la macro \STprintnum dans le tableau final. Par défaut, cette macro ne procède à aucune modification de son argument et sa définition est la suivante :

```
\def\STprintnum#1{#1}
```

L'utilisateur peut, s'il le souhaite, la modifier selon sa convenance comme ci-dessous avec l'aide du package siunitx et sa macro \num :

```
\def\STprintnum#1{\num[round-mode = places,
                       round-precision = 6,
                       drop-zero-decimal=true,
                       output-decimal-marker = {,}]{#1}%
}
\begin{spreadtab}{{tabular}{*4c}}                \hline
    @$x$           & 3              & 17 & 751 \\\hline
    @$1/x$         & \STcopy>{1/b1}  &    &     \\\hline
    @$x\cdot(1/x)$ & \STcopy>{b1*b2} &    &     \\\hline
\end{spreadtab}
```

x	3	17	751
$1/x$	0,333 333	0,058 824	0,001 332
$x \cdot (1/x)$	1	1	1

Aucune modification de la précision des calculs internes n'est faite dans ce cas.

4.5 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.

```
\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.

4.6 Exporter la valeur d'une cellule

On peut être amené sauvegarder la valeur numérique d'une cellule dans une macro (l'assignation est globale) pour l'afficher dans un champ textuel[4] ou même à l'extérieur du tableau. On doit alors utiliser la clé save list :

```
save list={⟨macroA⟩ = ⟨refA⟩, ⟨macroB⟩ = ⟨refB⟩,...}
```

où chaque référence doit être absolue.

```
\STset{ save list = { \resultC = c1 , \resultB = b1 }}
\begin{spreadtab}{{tabular}{|c|c|c|c|c|}}\hline
    10 & a1+10 & b1+10 & a1+b1+c1 & @cell c1 : \resultC\\\hline
\end{spreadtab}
\par\medskip
Voici la cellule c1 : \resultC\ et b1 : \resultB
```

10	20	30	60	cell c1 : 30

Voici la cellule c1 : 30 et b1 : 20

4. Il est plus facile d'utiliser << et >>, voir section suivante.

Lorsque la clé `aux save list` est utilisée avec la même syntaxe, tout se passe comme précédemment mais l'assignation est aussi écrite dans le fichier auxiliaire. Ainsi, la valeur d'une cellule peut se transmettre entre compilations et on peut donc faire référence à cette valeur avant que le tableau ne soit rencontré.

```
La valeur de d1 est \cellD.\par
\STset{ aux save list = { \cellD = d1 }}
\begin{spreadtab}{{tabular}{|c|c|c|c|}}\hline
    1 & a1+10 & b1+10 & a1+b1+c1\\\hline
\end{spreadtab}
```

La valeur de d1 est 33.

1	11	21	33

Dans les deux cas, la valeur des clés `save cell` et `aux save cell` est vidée après l'affichage du tableau qui suit `\STset`.

Il est également possible, avec la macro-fonction `tag`, d'exporter une valeur dans le fichier auxiliaire, voir page 13.

4.7 Afficher la valeur d'une cellule

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 relative ou absolue. Si ce qui est entre `<<` et `>>` n'est pas une référence, alors rien n'est fait. La ⟨*référence*⟩ ne doit contenir aucun espace. Si on écrit `<< a1>>` alors, l'espace fait que la référence n'est pas reconnue.

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

23	32	Moy $= \frac{23+32}{2} = 27.5$

Les caractères qui délimitent la référence valent `<<` et `>>` par défaut. Ils peuvent être modifiés avec la clé `display marks`. En écrivant `display marks={|;|}`, on devra écrire `|`⟨*référence*⟩`|` :

```
\begin{spreadtab}[display marks={|;|}]{{tabular}{|c|c||c|}}\hline
    23 & 32 & Moy $= \frac{|a1|+|b1|}{2} = :={(a1+b1)/2}$\\\hline
\end{spreadtab}
```

23	32	Moy $= \frac{23+32}{2} = 27.5$

4.8 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 totalité de la cellule, c'est-à-dire `\multicolumn{2}{|c|}{`⟨*formule*⟩`}` serait considérée comme étant une formule.

5 Macro-fonctions

5.1 Macro-fonctions mathématiques

5.1.1 Sommer des cellules

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

sa syntaxe est `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*⟩.

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`.

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)}}\\                                  somme=31
    [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]\\          1   4   6   4   1
    [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]       &       \\          1   3   3   1
    [0,1] & [-1,1]+[0,1] & [-1,1]       &              &       \\          1   2   1
    [0,1] & [-1,1]       &              &              &       \\          1   1
    1     &              &              &              &                   1
\end{spreadtab}
```

5.1.2 Macro-fonction `sumprod`

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

Sa syntaxe est : `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}}
    @Âges   & 10 & 11 & 12 & 13 & 14 & 15\\
    @Nombre & 5  & 8  & 20 & 55 & 9  & 3 \\\hline
    \multicolumn{7}{c}{Moyenne = :={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.

5.1.3 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*⟩`)`

```
\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

5.1.4 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 :

1. 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 majuscule ou minuscule ;
2. une référence au champ *textuel* d'une cellule qui doit contenir des caractères obéissants à la syntaxe vue au point précédent.

```
\begin{spreadtab}{{tabular}{|r|r|}}\hline
    @Écritures scientifiques & @Écritures décimales \\\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 `l3fp` 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`.

5.1.5 Identité

La plus simple des macros fonctions est `id` puisque `id(`⟨*nombre*⟩`)` 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 de 1 à 10               & 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 de 1 à 10	1	2	3	4	5	6	7	8	9	10
Moyenne des 8 premiers entiers	4.5									

5.2 Macro-fonctions de test

3 macro-fonctions de test sont disponibles et seront prochainement supprimées puisque `l3fp` 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 `l3fp` 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

5.3 Macro-fonctions de date

5.3.1 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 [5]. 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 « `:={}` »

```
\begin{spreadtab}{{tabular}{cc}}
    @14/7/1789             & frshortdatetonum(a1)\\
    1/1/2001 :={}          & frshortdatetonum(a2)\\
\end{spreadtab}
```

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

Une autre macro-fonction existe, elle transforme une date longue du type « 25 décembre 2024 » en un nombre.

```
\begin{spreadtab}{{tabular}{cc}}
    @1 janvier 2001   & frlongdatetonum(a1)\\
    @25 décembre 2024 & frlongdatetonum(a2)
\end{spreadtab}
```

1 janvier 2001	730791
25 décembre 2024	739550

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

5.3.2 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 30 jours avant puis 30 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
    -30 & numtofrshortdate(frshortdatetonum(a1)+[-1,0])\\
    -30 & numtofrlongdate(frshortdatetonum(a1)+[-1,0]) \\
    -30 & numtofrmonth(frshortdatetonum(a1)+[-1,0])    \\
    -30 & numtofrday(frshortdatetonum(a1)+[-1,0])      \\\hline
    30  & numtofrshortdate(frshortdatetonum(a1)+[-1,0])\\
    30  & numtofrlongdate(frshortdatetonum(a1)+[-1,0]) \\
    30  & numtofrmonth(frshortdatetonum(a1)+[-1,0])    \\
    30  & numtofrday(frshortdatetonum(a1)+[-1,0])
\end{spreadtab}
```

1/6/2009	
-30	2/5/2009
-30	2 mai 2009
-30	mai
-30	samedi
30	1/7/2009
30	1 juillet 2009
30	juillet
30	mercredi

5.4 Macro fonctions de coordonnées

5.4.1 `tag`, `cell`, `value` et `\STtag`

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` a pour syntaxe `tag(`⟨*nom*⟩`)`. Cela 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 globalement via un `\gdef` 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 renvoie 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, 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 `value(⟨nom⟩)`

```
\begin{spreadtab}{{tabular}{|c|c|c|}}\hline
    100 & 75 & a1+b1tag(ab:sum)\\\hline
\end{spreadtab}

\begin{spreadtab}{{tabular}{|c|c|}}\hline
    value(ab:sum) & 1.20*value(ab:sum)\\\hline
\end{spreadtab}
```

100	75	175

175	210

Si la clé `tag to aux` est ⟨*true*⟩, les assignations sont également écrites dans le fichier auxiliaire, ce qui les rend transmissibles d'une compilation à l'autre. Il devient alors possible de placer `\STtag{⟨nom⟩}` avant le tableau où `tag(⟨nom⟩)` est spécifié.

5.4.2 `row` et `col`

spreadtab fournit également les macros fonctions `row(⟨tagname⟩)` et `col(⟨tagname⟩)` qui renvoient le numéro de la ligne ou de la colonne où se trouvait `tag(⟨tagname⟩)`. Voici par exemple une façon de dénombrer des valeurs pour en calculer la moyenne : on nomme `first` la première valeur et `last` la dernière, et le nombre de valeurs est donc `row(last)-row(first)+1` :

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

moyenne =

7
9
15
6
20
13
11
55
17

6 Précautions particulières

6.1 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	7	8
1	1	2	3	5	8	13	21	34

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

```
\newcommand\dline{\hline\hline}
\def\xtracol{&&&&&&}
\begin{spreadtab}{{tabular}{*9c}}
    0     & 1 & \STcopy>{b1+1} \STxp\xtracol\\\dline
    1     & 1 & \STcopy>{a2+b2}\STxp\xtracol
\end{spreadtab}
```

En effet la compilation échoue et dans le log, on lit « `! Improper alphabetic constant.` »

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

```
\dline 1
```

Pour pouvoir compiler ce code sans erreur, la cellule `b1` *doit* contenir un marqueur de champ numérique :

```
\newcommand\dline{\hline\hline}
\def\xtracol{&&&&&&}
\begin{spreadtab}{{tabular}{*9c}}
    0     & 1 & \STcopy>{b1+1} \STxp\xtracol\\\dline
    :={1} & 1 & \STcopy>{a2+b2}\STxp\xtracol
\end{spreadtab}
```

0	1	2	3	4	5	6	7	8
1	1	2	3	5	8	13	21	34

6.2 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.

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

6.3 Messages émis par spreadtab

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

- en cas de référence circulaire. 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 émet par défaut des messages d'information (dans le fichier de log), sur ce qu'il est en train de faire. La clé `messages`, selon qu'elle vaut `true` ou `false` permet ou pas l'émission de messages d'information.

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 :

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

L'environnement spécifié par l'utilisateur est repris (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 la ligne se termine par `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 bien, elles ne font références qu'à des cellules déjà calculées.

6.4 Débogage

Pour faciliter l'utilisation de spreadtab, un mode de débogage est disponible. Il est activé lorsque la clé `debug` n'est pas vide et contient au moins un des mots ⟨*formula*⟩, ⟨*text*⟩ ou ⟨*cellcode*⟩, séparés par des virgules. Ces mots provoquent chacun l'affichage d'un tableau de débogage :

— ⟨*formula*⟩ : affichage des champs numériques de toutes les cellules et les fins de ligne ;

— ⟨*text*⟩ : affichage des champs textuels de toutes les cellules ;

— ⟨*cellcode*⟩ : 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. −1 s'il s'agit d'une cellule fusionnée par une commande `\multicolumn` ;
2. 0 si la cellule est vide ou purement textuelle ;
3. 1 si le champ numérique de la cellule contient une formule qui sera à évaluer plus tard ;
4. 2 si la champ numérique de la cellule contient un nombre.

On peut ajouter la valeur ⟨*show tab*⟩ pour forcer l'affichage du tableau obtenu qui sinon, n'est pas affiché.

Voici un tableau pour lequel on affiche les 3 tableaux de débogage et aussi le tableau final obtenu :

```
\STset{debug={text,formula,cellcode,show tab}}
\begin{spreadtab}{{tabular}{|rr|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}
```

	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	
1				\hline
2	22	54	a2+b2	\\\hline \hline
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

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

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 de débogage, les cellules contenant les coordonnées ne sont grisées que si le package `colortbl` a été chargé.

7 Liste des clés

Tous les réglages concernant spreadtab se font à partir de la version 0.6 par un système de ⟨*clés*⟩=⟨*valeur*⟩ qui peuvent se trouver : *New v0.6*

1. dans l'argument de `\STset` auquel cas les réglages spécifiés concernent tous les tableaux à venir ;
2. dans l'argument optionnel de `\begin{spreadtab}` ou `spreadtab` et dans ce cas, les réglages spécifiés ne concernent que le tableau en cours.

Voici la liste des ⟨*clés*⟩ et leurs valeurs par défaut :

⟨*clé*⟩	⟨*valeur*⟩	voir page	Description
`tabline sep`	⟨\\⟩	3	séparateur entre les lignes du tableau
`text mark`	⟨@⟩	4	la ⟨*valeur*⟩ est détokénisée : si elle est présente dans une cellule, spreadtab la comprend comme une cellule de texte
`numeric mark`	⟨:=⟩	5	l'argument entre accolades qui suit est le champ numérique de la cellule

⟨*clé*⟩	⟨*valeur*⟩	voir page	Description
`freeze char`	⟨!⟩	6	caractère détokénisé qui, s'il est présent devant la lettre ou le nombre d'une référence, ne l'incrémente pas lors de la copie de la formule
`pretab code`	◊ (vide)	6	code exécuté juste avant l'affichage du tableau
`posttab code`	◊ (vide)	6	code exécuté juste après l'affichage du tableau
`dec sep`	⟨.⟩	7	séparateur décimal des nombres après calcul du tableau
`autoround`	◊ (vide)	7	Une absence de valeur est le comportement normal sinon, un ⟨*entier*⟩ positif est attendu et indique le rang de l'arrondi des calculs internes
`save list`	◊ (vide)	8	liste csv de la forme ⟨*macro*⟩=⟨*référence absolue*⟩ : chaque nombre contenu dans les cellules référencées est stockée (globalement) dans une macro. Cette clé est réinitialisée à vide après chaque tableau
`aux save list`	◊ (vide)	9	même syntaxe et comportement que ci-dessus, mais les assignations sont également écrites dans le fichier auxiliaire. Cette clé est réinitialisée à vide après chaque tableau
`display marks`	⟨<<;>>⟩	9	marqueurs autour d'une référence absolue dans un champ textuel pour afficher le nombre contenu dans la cellule référencée
`tag to aux`	⟨*false*⟩	14	lorsque ⟨*true*⟩, les assignations faites par la macro-fonction `tag` sont également écrites dans le fichier auxiliaire
`messages`	⟨*false*⟩	16	permet à spreadtab d'émettre des messages dans le fichier log sur ce qu'il est en train de faire
`debug`	◊ (vide)	16	doit contenir sous forme csv des mots parmi ⟨*text*⟩, ⟨*formula*⟩ ou ⟨*cellcode*⟩ pour afficher le tableau de débogage correspondant. Si le mot ⟨*show tab*⟩ est ajouté, le tableau est également affiché. Cette clé est réinitialisée à vide après chaque tableau.

Pour revenir aux valeurs par défaut, on peut à tout moment exécuter la macro `\STreset`.

* *
*

Merci de me signaler par `email` tout bug, régression ou toute nouvelle fonctionnalité à implémenter que vous pensez utile.