tabularcalc

v0.2

Manuel de l'utilisateur

Christian Tellechea
unbonpetit@gmail.com

21 avril 2009

Résumé

Étant donné une liste de nombres et une ou plusieurs formules à une variable, cette extension, à l'aide d'une syntaxe simple, construit un tableau de valeurs, c'est-à-dire un tableau dont la première ligne contient les nombres les autres lignes les résultats pris par la (ou les) formules pour chacun des nombres de la liste :

x	-4	-2	0	2,25	7
$f(x)=2x-3$	-11	-7	-3	1,5	11
x^2	16	4	0	5,062 5	49
$h(x)=\sqrt{x^2+1}$	4,123 106	2,236 068	1	2,462 214	7,071 068

Le tableau peut être construit horizontalement ou verticalement, et il est entièrement personnalisable, autant du point de vue des filets, que de la hauteur des lignes ou que des types de colonnes. De plus, le contenu de n'importe quelle cellule du tableau peut être masqué.

Des effets encore plus précis sont possibles puisqu'une commande permet de faire exécuter n'importe quel code dans une cellule particulière.

Table des matières

J'adresse mes remerciements à Derek O'Connor pour l'intérêt qu'il a porté à cette extension et pour les tests qu'il a effecté sur les versions beta que je lui ai envoyées. Ses suggestions — pertinentes — de nouvelles fonctionnalités m'ont été précieuses. Sans ses conseils, tabularcalc ne serait pas ce qu'il est.

Merci également à Le Huu Dien Khue qui m'a gentiment proposé de traduire ce manuel en vietnamien.

1 Introduction

1.1 Présentation

Cette extension permet de construire facilement des tableaux de résultats en évaluant des formules dont la variable prend des valeurs données dans une liste. Les tableaux sont affichés avec l'environnement standard `tabular`. L'affichage des nombres se fait sous forme décimale.
tabularcalc fonctionne sous LaTeX et charge les extensions `fp`, `xstring` et `numprint` si ce n'a pas été le cas.
Cette extension n'est pas du tout destinée à entrer en concurrence avec `pgfplotstable`, l'excellente extension de Christian Feuersänger. Cette dernière est en effet bien plus personnalisable que tabularcalc au prix cependant d'une syntaxe et d'une difficulté d'utilisation plus grande. tabularcalc se veut plus modeste et privilégie la facilité d'utilisation combinée à des possibilités de personnalisation aisément accessibles.

En ce qui concerne l'affichage des nombres décimaux dans le tableau, de l'avis de l'auteur, rien ne le fait mieux que l'extension `numprint` qui est donc également requise. On peut changer le moteur d'affichage des nombres décimaux et/ou entièrement personnaliser l'affichage des nombres (voir page 9).

1.2 L'extension `fp`

Du côté des calculs, l'évaluation d'une expression numérique comme `2*x*x-5*x+7` lorsque `x = 2.7` est, avec TEX, une chose complexe que tabularcalc ne réalise pas. Pour cette tâche, il fait appel à un moteur de calcul fourni par l'extension « `fp` » qui offre toutes les fonctions arithmétiques, scientifiques et trigonométriques habituelles.

On peut utiliser indifféremment la syntaxe infixée ou postfixée. Consulter le fichier `README` de l'extension `fp` pour la liste exhaustive des fonctions disponibles selon la notation employée.

Sans demander l'autorisation de l'auteur ni même l'en informer[1], je me suis permis de corriger 2 problèmes de l'extension `fp`, tous deux dans la macro `\FPpow`, s'occupant du calcul des puissances :

- tout d'abord, un espace indésirable était ajouté lors du calcul d'une puissance par `fp` ;
- il y a plus gênant puisque pour calculer a^b, `fp` utilise la formule $\mathrm{e}^{b \ln a}$. Il y a un petit problème lorsque b est entier et a est négatif puisqu'alors : $(-3)^2 = \mathrm{e}^{2 \ln(-3)}$ et le logarithme d'un nombre négatif est indéfini.

Pour laisser tabularcalc corriger ces 2 problèmes, on peut passer l'option « `fixFPpow` » au package :

```
\usepackage[fixFPpow]{tabularcalc}
```

1.3 Ce qui est nouveau

Hélas, on m'a fait part d'incompatibilité avec d'autres packages car le nom de certaines macros de tabularcalc était déjà utilisé. J'ai donc décidé — la mort dans l'âme — de renommer les macros publiques, au risque certain de créer une incompatibilite avec la version 0.1 ; que les utilisateurs m'excusent pour ce désagrément !
Pour éviter tout nouveau risque, je mets donc « `tc` » (comme tabularcalc[2]) devant le nom de presque toutes les macros publiques. Pour ceux qui utilisent déjà tabularcalc, voici tous les changements de nom :

Ancien nom	Nouveau nom
`\noshowmark`	`\tcnoshowmark`
`\startline`	`\tcatbeginrow`
`\resetcellcode`	`\tcresetcellcode`
`\listsep`	`\tclistsep`
`\printvalue`	`\tcprintvalue`
`\printresult`	`\tcprintresult`
`\sethrule`	`\tcsethrule`
`\resethrule`	`\tcresethrule`
`\setcoltype`	`\tcsetcoltype`
`\resetcoltype`	`\tcresetcoltype`

Autres nouveautés pour ceux qui sont familiers avec cette extension :

- les calculs effectués par `pgfmath` étant beaucoup trop imprécis, cette extension est abandonnée au profit de `fp` qui offre une précision bien plus grande ;
- les valeurs peuvent maintenant être calculées ;
- on peut désormais écrire le code correspondant à un tableau dans un fichier pour le modifier à la main.

1.4 Vocabulaire

Pour que des points de vocabulaire soient clairs par la suite, dans les tableaux triviaux ci-dessous, les nombres en rouge sont les « valeurs », les nombres en bleu sont les « résultats », et les textes en brun sont les « labels ». La cellule en haut à gauche est la « cellule (0,0) ». Ce vocabulaire sera employé ensuite.

1. Je crois que l'auteur a depuis longtemps délaissé le monde de LATEX.
2. Je préfère préciser qu'il s'agit d'un racourci pour tabularcalc et non de mes initiales : je ne suis pas narcissique à ce point !

Tableau horizontal

cellule (0,0)	-5	-1	0	3	10
x	-5	-1	0	3	10
$2x$	-10	-2	0	6	20
$3x$	-15	-3	0	9	30

Tableau vertical

cellule (0,0)	x	$2x$	$3x$
-5	-5	-10	-15
-1	-1	-2	-3
0	0	0	0
3	3	6	9
10	10	20	30

2 Fonctions basiques

2.1 Tableaux horizontaux

La commande \htablecalc permet de construire un tableau de valeur horizontal, dont la 1re ligne contiendra les valeurs, et les lignes suivantes les résultats. La syntaxe est :

\htablecalc[⟨*n*⟩]{⟨*cellule (0,0)*⟩}{⟨*variable=liste val*⟩}
{⟨*label 1*⟩}{⟨*formule 1*⟩}
{⟨*label 2*⟩}{⟨*formule 2*⟩}
...
{⟨*label n*⟩}{⟨*formule n*⟩}

où :

- ⟨*n*⟩ est le nombre de formules à évaluer. Ce nombre vaut 1 par défaut ;
- ⟨*cellule(0,0)*⟩ est le contenu de la cellule (0,0) ;
- ⟨*variable*⟩ est la variable qui interviendra dans les ⟨*formules i*⟩ servant à calculer les résultats.
- ⟨*liste val*⟩ est la liste des valeurs, séparées par une virgule. Noter que les valeurs décimales doivent avoir le point comme séparateur décimal ;
- ⟨*label i*⟩ est le ie label ;
- ⟨*formule i*⟩ est la ie formule qui servira à évaluer les résultats de la ie ligne.

Dans la liste de valeurs, le séparateur par défaut est la virgule qui est le développement de \tclistsep, et pour changer ce séparateur en « | » par exemple, il faut écrire : \def\tclistsep{|}

À titre d'exemple, voici un premier essai pour obtenir le tableau de la première page :

```
\htablecalc [3]{$x$}{x=-4,-2,0,2.25,7}
                  {$f(x)=2x-3$}{2*x-3}
                  {$x^2$}{x*x}
                  {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

x	-4	-2	0	2,25	7
$f(x) = 2x - 3$	-11	-7	-3	1,5	11
x^2	16	4	0	5,062 5	49
$h(x) = \sqrt{x^2 + 1}$	4,123 106	2,236 068	1	2,462 214	7,071 068

On peut observer que le tableau n'est pas strictement identique à celui de la première page : les colonnes contenant les résultats ne sont pas toutes de la même largeur et le filet sous la 1re ligne est différent. Nous verrons comment personnaliser tout cela plus loin.

2.2 Tableaux verticaux

La commande \vtablecalc permet de construire un tableau de valeur vertical, dont la 1re colonne contiendra les valeurs, et les colonnes suivantes les résultats. La syntaxe est :

\vtablecalc[⟨*n*⟩]{⟨*cellule (0,0)*⟩}{⟨*variable=liste val*⟩}
{⟨*label 1*⟩}{⟨*formule 1*⟩}
{⟨*label 2*⟩}{⟨*formule 2*⟩}
...

`{`⟨*label n*⟩`}{`⟨*formule n*⟩`}`

où :

- ⟨*n*⟩ est le nombre de formules à évaluer. Ce nombre vaut 1 par défaut ;
- ⟨*cellule(0,0)*⟩ est le contenu de la cellule (0,0) ;
- ⟨*variable*⟩ est la variable qui interviendra dans les ⟨*formules i*⟩ servant à calculer les résultats ;
- ⟨*liste val*⟩ est la liste des valeurs, séparées par une virgule. Noter que les valeurs décimales doivent avoir le point comme séparateur décimal ;
- ⟨*label i*⟩ est le i^{e} label ;
- ⟨*formule i*⟩ est la i^{e} formule qui servira à évaluer les résultats de la i^{e} ligne.

À titre d'exemple, voici le tableau précédent présenté verticalement, mais où l'on prend la variable des formules comme étant `y` :

```
\vtablecalc [3]{$x$}{y=-4,-2,0,2.25,7}
                   {$f(x)=2x-3$}{2*y-3}
                   {$x^2$}{y*y}
                   {$h(x)=\sqrt{x^2+1}$}{round(root(2,y*y+1),6)}
```

x	$f(x) = 2x - 3$	x^2	$h(x) = \sqrt{x^2 + 1}$
-4	-11	16	4,123 106
-2	-7	4	2,236 068
0	-3	0	1
2,25	1,5	5,062 5	2,462 214
7	11	49	7,071 068

2.3 Masquer des cellules

On peut masquer le contenu de n'importe quelle cellule, aussi bien dans un tableau horizontal que vertical.

2.3.1 Masquer une valeur

Si on veut masquer une valeur, il suffit de la faire précéder d'un « `@` » dans la liste. Dans l'exemple suivant, on masque la 2^{e} et la 5^{e} valeur :

```
\htablecalc [3]{$x$}{x=-4,@-2,0,2.25,@7}
                   {$f(x)=2x-3$}{2*x-3}
                   {$x^2$}{x*x}
                   {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

x	-4		0	2,25	
$f(x) = 2x - 3$	-11	-7	-3	1,5	11
x^2	16	4	0	5,062 5	49
$h(x) = \sqrt{x^2 + 1}$	4,123 106	2,236 068	1	2,462 214	7,071 068

Le token « `@` » est en réalité le développement de la commande `\tcnoshowmark`. Pour changer le token qui masque une valeur, il suffit de redéfinir cette macro. Par exemple, pour faire tenir ce rôle au signe « = », on écrirait `\def\tcnoshowmark{=}`

2.3.2 Masquer un résultat

Pour une valeur donnée, si on veut masquer les résultats numéro $a_1, a_2, \ldots, a_n$, il suffit de faire suivre cette valeur par $[a_1][a_2]\ldots[a_n]$ où les nombres a_i sont dans l'ordre croissant. Si un des nombres a_j vaut 0, tous les autres a_k où $k > j$ sont ignorés et tous les résultats qui suivront le précédent résultat masqué seront masqués.

Dans l'exemple qui suit, on va :

– masquer le 2e résultat de la première valeur avec « `-4[2]` »
– laisser tous les résultats de la 2e valeur avec « `-2` »
– masquer les résultats no 1 et 3 de la troisième valeur avec « `0[1][3]` »
– masquer tous les résultats de la 4e valeur avec « `2.25[0]` »
– masquer tous les résultats à partir du 2e pour la 5e valeur avec « `7[2][0]` »

```
\htablecalc[3]{$x$}{x=-4[2],-2,0[1][3],2.25[0],7[2][0]}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

x	-4	-2	0	2,25	7
$f(x) = 2x - 3$	-11	-7			11
x^2		4	0		
$h(x) = \sqrt{x^2 + 1}$	4,123 106	2,236 068			

On peut combiner cette syntaxe avec `@` pour masquer à la fois la valeur et certains résultats.

2.4 Hauteur des lignes

Au début de chaque ligne, lors de son affichage, la commande `\tcatbeginrow` est exécutée.
Par défaut, cette commande est définie par : `\def\tcatbeginrow{\rule[-1.2ex]{0pt}{4ex}}`. Cette commande se développe donc par défaut en un « strut » qui fixe la hauteur de la ligne. Voici ce strut, rendu visible devant la lettre a : a

On peut donc faire exécuter un autre strut ou tout autre action au début d'une ligne :

```
\def\tcatbeginrow{%
    {\bfseries\number\tclin)\ }%
}
\htablecalc[3]{$x$}{x=-4,-2,0,2.25,7}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

0) x	-4	-2	0	2,25	7
1) $f(x) = 2x - 3$	-11	-7	-3	1,5	11
2) x^2	16	4	0	5,062 5	49
3) $h(x) = \sqrt{x^2 + 1}$	4,123 106	2,236 068	1	2,462 214	7,071 068

Ici, on ne définit aucun strut (on redonne donc aux lignes leur hauteur naturelles) et on affiche en gras le numéro de la ligne en cours (qui est contenue dans le compteur `\tclin`) à la ligne 2 du code.

2.5 Filets horizontaux

tabularcalc permet de définir 3 types de filets horizontaux. Pour cela, la commande `\tcsethrule` admet 3 arguments :
– le « filet 0 » affiché en haut et en bas du tableau ;
– le « filet 1 » affiché sous la première ligne ;
– les « autres filets » affichés sous les lignes de résultats (sauf la dernière qui reçoit le filet de bas de tableau).
Voici la syntaxe :
`\tcsethrule{`⟨*filet 0*⟩`}{`⟨*filet 1*⟩`}{`⟨*autres filets*⟩`}`

Par défaut, les 3 arguments valent `\hline`.

Voici un exemple où le filet sous la première ligne est double et les autres filets sont supprimés :

```
\tcsethrule{\hline}{\hline\hline}{}
\htablecalc[3]{$x$}{x=-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

x	-2	-1	0	1	2	3
$2x$	-4	-2	0	2	4	6
$3x$	-6	-3	0	3	6	9
$4x$	-8	-4	0	4	8	12

La commande `\tcresethrule` permet de revenir aux filets horizontaux définis par défaut.

2.6 Personnalisation des colonnes

2.6.1 Filets verticaux

tabularcalc permet de définir 2 types de colonnes : le type de la colonne de gauche et le type des autres colonnes. La commande `setcoltype` admet un argument optionnel et 2 arguments obligatoires :
– l'argument optionnel, vide par défaut, définit les filets verticaux « | » affichés à la droite du tableau ;
– le type 1 de la première colonne qui est prédéfini à « |c| » ;
– le type 2 des autres colonnes qui est prédéfini à « c| »

La syntaxe est :
`\tcsetcoltype[`⟨*filets de droite*⟩`]{`⟨*type 1*⟩`}{`⟨*type 2*⟩`}`

Voici un exemple :

```
\tcsetcoltype[||]{||c||}{c}
\htablecalc[3]{$x$}{x=-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

x	-2	-1	0	1	2	3
$2x$	-4	-2	0	2	4	6
$3x$	-6	-3	0	3	6	9
$4x$	-8	-4	0	4	8	12

La commande `\tcresetcoltype` permet de revenir aux types de colonne définis par défaut.

2.6.2 Largeur des colonnes

Au lieu de l'habituel spécificateur de colonne « c » que nous avons utilisé jusqu'à présent, on peut spécifier la largeur des colonnes avec le spécificateur « m » de l'extension `array` de cette façon : « `m{1.5cm}` ».

Voici un exemple où la 1re colonne est centrée à droite, les colonnes de résultats sont centrées et mesurent 1,5 cm de large :

```
\usepackage{array}
\tcsetcoltype{|r|}{>{\centering\arraybackslash}m{1.5cm}|}
\htablecalc[3]{$x$}{x=-4,-2,0,2.25,7}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

x	-4	-2	0	2,25	7
$f(x) = 2x - 3$	-11	-7	-3	1,5	11
x^2	16	4	0	5,062 5	49
$h(x) = \sqrt{x^2 + 1}$	4,123 106	2,236 068	1	2,462 214	7,071 068

3 Calculer les valeurs

Lorsque les valeurs sont assez nombreuses et suivent une règle mathématique, il peut être plus simple de saisir la formule que toutes les valeurs. Ainsi la syntaxe :

```
\htablecalc [2]{$x$}{x=-3,-1,1,3,5,7,9,11,13}
                   {$2x$}{2*x}
                   {$x^2$}{x*x}
```

peut être saisie ainsi :

```
\htablecalc [2]{$x$}{x=a;a=-3:13[2]}
                   {$2x$}{2*x}
                   {$x^2$}{x*x}
```

La présence du « `;` » change la façon d'analyser l'argument contenant les valeurs : on exprime à droite du signe « `;` » que la variable muette « `a` » va parcourir l'intervalle $-3 \ldots 13$ avec un incrément de 2. Les valeurs seront donc des entiers *impair*. À gauche de « `;` », on exprime que la variable intervenant dans les formules — ici `x` — va être égale à `a` donc prendra elle aussi les valeurs impaires de -3 à 13. On aurait pu aussi générer l'argument `{x=-3,-1,1,3,5,7,9,11,13}` avec `{x=2*a+1:a=-2:6}`. Cette fois-ci l'incrément vaut 1 par défaut. Il y plusieurs façons de parvenir à ces mêmes valeurs, celle-ci par exemple : `{x=2*a-3;a=0:8}`

Lorsqu'on utilise un argument avec « `;` », on perd la possibilité de masquer des cellules comme vu à la page 4. Il faut aussi prendre garde au nombres de valeurs générées afin que le tableau ne devienne pas gigantesque.

Avec cette syntaxe, l'argument contenant les valeurs se présente sous cette forme :

⟨*variable 1*⟩`=`⟨*formule*⟩`;`⟨*variable 2*⟩`=`⟨*min*⟩`:`⟨*max*⟩`[`⟨*incrément*⟩`]`

où :

- ⟨*variable 1*⟩ est la variable intervenant dans les formules servant à évaluer les résultats ;
- ⟨*variable 2*⟩ est la variable intervenant dans les formules servant à évaluer les valeurs. Elle doit être différente de ⟨*variable 1*⟩.
- ⟨*formule*⟩ est la formule qui va évaluer les différentes valeurs. La variable intervenant dans cette formule est ⟨*variable 2*⟩ ;
- ⟨*min*⟩`:`⟨*max*⟩ représentent l'intervalle dans lequel doit varier la ⟨*variable 2*⟩ ;
- ⟨*incrément*⟩ est le pas d'incrémentation, il est facultatif et vaut `1` par défaut. Il ne doit pas être nul.

Comme on l'a vu, il y a de plusieurs façons différentes de générer les mêmes valeurs. Ainsi, les valeurs `{0,1,2,3,4,5,6,7,8,9,10}` sont générées par :

- `{z=x;x=0:10}` et « `z` » sera la variable des formules ;
- `{n=2*a;a=0:5[0.5]}` et « `n` » sera la variable des formules ;
- `{x=y/10;y=0:100[10]}` et « `x` » sera la variable des formules ;

Il va sans dire que l'intervalle et l'incrément doivent être cohérents entre-eux. Un argument `0:10[-1]` va provoquer message d'erreur de tabularcalc !

À titre d'exemple, voici un tableau utilisant les fonctions trigonométriques de `fp` :

```
\htablecalc[6]{$x$\ [deg]}{x=a;a=15:75[15]}
              {$\sin x$}{round(sin(x*pi/180),6)}
              {$\cos x$}{round(cos(x*pi/180),6)}
              {$\tan x$}{round(tan(x*pi/180),6)}
              {$\sin^2x$}{round(sin(x*pi/180)^2,6)}
              {$\cos^2x$}{round(cos(x*pi/180)^2,6)}
              {$\tan^2x$}{round(tan(x*pi/180)^2,6)}
```

x [deg]	15	30	45	60	75
$\sin x$	0,258 819	0,5	0,707 107	0,866 025	0,965 926
$\cos x$	0,965 926	0,866 025	0,707 107	0,5	0,258 819
$\tan x$	0,267 949	0,577 35	1	1,732 051	3,732 051
$\sin^2 x$	0,066 987	0,25	0,5	0,75	0,933 013
$\cos^2 x$	0,933 013	0,75	0,5	0,25	0,066 987
$\tan^2 x$	0,071 797	0,333 333	1	3	13,928 203

Et un autre tableau qui affiche des puissances de 10, leur logarithme décimal, leur racine carrée et leur inverse :

```
\htablecalc[3]{Puissances de 10}{x=round(10^n,4);n=-3:3}
              {Logarithme d\'ecimal}{ln(x)/ln(10)}
              {Racine carr\'ee}{round(root(2,x),3)}
              {Inverse}{1/x}
```

Puissances de 10	0,001	0,01	0,1	1	10	100	1 000
Logarithme décimal	−3	−2	−1	0	1	2	3
Racine carrée	0,032	0,1	0,316	1	3,162	10	31,623
Inverse	1 000	100	10	1	0,1	0,01	0,001

4 Personnalisation avancée

4.1 Exécution d'un code dans une cellule

La macro `\defcellcode` permet d'exécuter un code donné dans une cellule, une colonne entière ou une ligne entière de son choix. Pour cela, les cellules du tableau sont repérées avec des coordonnées que voici :

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)

En interne, la 1re coordonnée — le numéro de la ligne — est contenu dans le compteur `\tclin`, tandis que le numéro de la colonne est contenu dans le compteur `\tccol`.

Voici la syntaxe de cette commande :
`\defcellcode{`⟨*nombre 1*⟩`}{`⟨*nombre 2*⟩`}{`⟨*code*⟩`}`

où :

- `{`⟨*nombre 1*⟩`}` est la première coordonnées, le numéro de la ligne ;
- `{`⟨*nombre 2*⟩`}` est la deuxième coordonnées, le numéro de la colonne ;
- `{`⟨*code*⟩`}` est le code qui sera exécuté lorsque la cellule choisie sera affichée : ce code ne sera développé qu'à ce moment là ;
- Si `{`⟨*nombre 1*⟩`}` est vide, toutes les lignes sont concernées ;

– Si {⟨*nombre 2*⟩} est vide, toutes les colonnes sont concernées ;

Il faut noter que le code ainsi défini est exécuté *lors de l'affichage de la cellule*, et à ce moment là, la valeur du compteur \tccol ne contient plus le numéro de la colonne de la cellule : il n'est donc pas question d'impliquer \tccol dans le code défini par la commande \defcellcode. Par contre, le compteur \tclin contient bien le numéro de la ligne en cours d'affichage.
Si la commande \defcellcode est appelée plusieurs fois pour définir des codes différents, et si plusieurs codes concernent une même cellule, les codes seront exécutés dans l'ordre où ils ont été définis.

Voici un exemple où, à l'aide de l'extension xcolor, on choisit d'écrire en bleu le contenu de la cellule (2 , 3), d'écrire en rouge le contenu de la ligne 1, et en brun le contenu de la colonne 4.

```
\usepackage{color}
\defcellcode{2}{3}{\color{blue}}
\defcellcode{1}{}{\color{red}}
\defcellcode{}{4}{\color{brown}}
\htablecalc[3]{$x$}{x=-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

x	-2	-1	0	1	2	3
$2x$	-4	-2	0	2	4	6
$3x$	-6	-3	0	3	6	9
$4x$	-8	-4	0	4	8	12

On peut observer que la cellule (1 , 4) qui contient 2 a été colorée en brun. En effet, elle a d'abord été colorée en rouge (ligne 3 du code) *puis* colorée en brun (ligne 4 du code).

Il existe une autre commande similaire qui exécute du code qui est \edefcellcode : cette fois ci, le code est exécuté une première fois lors de la construction de la cellule, alors que le compteur \tccol contient bien le numéro de la colonne de la cellule. Lors de cette première exécution, le code est développé au maximum avec un \edef [3]. Le développement obtenu est exécuté une seconde fois lors de l'affichage de la cellule.

Voici un exemple où le texte dans toutes les colonnes supérieures à la colonne n° 2 est coloré en bleu :

```
\usepackage{color}
\edefcellcode{}{}{%
    \ifnum\tccol>2 \noexpand\color{blue}\fi}
\htablecalc[3]{$x$}{x=-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

x	-2	-1	0	1	2	3
$2x$	-4	-2	0	2	4	6
$3x$	-6	-3	0	3	6	9
$4x$	-8	-4	0	4	8	12

4.2 Personnaliser l'affichage

4.2.1 Les macros \printvalue et \printresult

Pour afficher une valeur, la commande \tcprintvalue est appelée. Elle admet un argument qui est le nombre décimal à afficher qui provient de fp et se présente sous forme brute c'est-à-dire 12345.6789 par exemple pour 12 345,678 9.
Par défaut, \tcprintvalue est définie par le code suivant :

```
\def\tcprintvalue#1{\numprint{#1}}
```

3. Il convient de mettre un \noexpand devant les commandes que l'on ne veut pas développer à ce moment là.

On voit que la commande `\numprint` est appelée pour donner un affichage soigné.

Pour afficher un résultat, la commande `\tcprintresult` est appelée. Elle admet **deux** arguments ; le premier est le résultat provenant de `fp` et le deuxième est la valeur qui a servi à calculer le résultat, telle qu'elle a été saisie dans la liste de valeurs.
Par défaut, `\tcprintresult` est définie par le code suivant :

```
\def\tcprintresult#1#2{\numprint{#1}}
```

On peut observer que l'argument `#2` (la valeur) n'est pas exploitée par `\tcprintresult`. On peut cependant imaginer un exemple où elle le serait. Sur cet exemple, on affiche un `X` rouge lorsque la dimension du côté du carré (qui est l'argument `#2`) est négative. Sinon, on affiche le résultat avec l'unité. En plus, on affiche en bleu tout résultat inférieur à 10 :

```
\usepackage{color}
\def\tcprintresult#1#2{%
    \ifdim#1pt<10pt\color{blue}\fi
    \ifdim#2pt<0pt
        \color{red}\texttt{X}%
    \else
        \numprint[cm^2]{#1}%
    \fi}
\htablecalc{longueur}{x=0.7,-10,3,-2,5,12}
           {aire du carr\'e}{x*x}
```

longueur	0,7	−10	3	−2	5	12
aire du carré	0,49 cm^2	X	9 cm^2	X	25 cm^2	144 cm^2

4.2.2 Gérer les arrondis

Les calculs faits par `fp` ont une grande précision $\left(10^{-12}\right)$ et les développements décimaux ont parfois beaucoup de chiffres après la virgule. Voici par exemple $\sqrt{10}$ calculé par `fp` :

3,162 277 660 168 379 312

Les 11 premiers chiffres sont exacts, le 12$^{\text{e}}$ est arrondi.

Pour les résultats, on peut utiliser la fonction `round(nombre,precision)` de `fp`. Si l'on ne veut pas alourdir la syntaxe des formules, on peut aussi utiliser la commande `\tcprintroundresult` offerte par tabularcalc. Son argument est le nombre de chiffres après la virgule que l'on veut voir affichés. La macro étoilée `\tcprintroundresult*` complète le nombre avec des 0 inutiles si nécessaire. Si l'argument est vide, la macro est neutralisée, et aucun arrondi n'est fait (comprtement par défaut).

```
\tcprintroundresult{3}
\htablecalc{$x$}{x=2,3,4,5}
           {$\sqrt{x}$}{root(2,x)}
```

x	2	3	4	5
$\sqrt{x}$	1,414	1,732	2	2,236

```
\tcprintroundresult*{3}
\htablecalc{$x$}{x=2,3,4,5}
           {$\sqrt{x}$}{root(2,x)}
```

x	2	3	4	5
$\sqrt{x}$	1,414	1,732	2,000	2,236

En ce qui concerne les valeurs calculées, il est **déconseillé** d'utiliser la fonction `round` puisque si tel était le cas, la valeur arrondie serait utilisée pour le calculs des résultats. En voici la démonstration sur cet exemple où l'on prend comme valeurs les valeurs arrondies à 10^{-2} les racines carrées des entiers de 2 à 4 que l'on élève ensuite au carré :

```
\htablecalc{racines}{x=round(root(2,k),2);k=2:4}
           {carr\'e}{x*x}
```

racines	1,41	1,73	2
carré	1,988 1	2,992 9	4

Les erreurs d'arrondis des valeurs se retrouvent dans les résultats.
Il convient donc d'utiliser \tcprintroundvalue dont le fonctionnement et la syntaxe sont les mêmes que \tcprintroundresult :

```
\tcprintroundvalue{2}
\htablecalc{racines}{x=root(2,k);k=2:4}
            {carr\'e}{x*x}
```

racines	1,41	1,73	2
carré	1,999 999 999 999 999 98	2,999 999 999 999 999 935	3,999 999 999 999 999 96

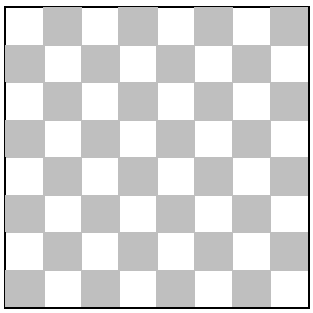
Les résultats — non arrondis, eux ! — sont bien plus proches des entiers attendus.

4.2.3 Pour le fun

On peut imaginer des détournements de tabularcalc, comme par exemple afficher un échiquier dont les cases mesurent 0,5 cm de côté :

- on initialise à 0pt les séparateurs du tableau à la ligne 1 pour ne pas fausser la dimension voulue de 0,5 cm ;
- l'affichage de toutes les valeurs et résultats est neutralisé à la ligne 2 ;
- ensuite on ne dessine que les filets horizontaux du haut et du bas du tableau (ligne 3), et les filets de gauche et droite (ligne 4) ;
- on définit un strut de hauteur 0,5 cm au début de chaque ligne (ligne 5) ;
- et on teste si la somme de la ligne et de la colonne en cours est impaire (ligne 7) auquel cas, on colore la case en gris (ligne 8).

```
\arraycolsep=0pt\tabcolsep=0pt
\def\tcprintvalue#1{}\def\tcprintresult#1#2{}
\tcsethrule{\hline}{}{}
\tcsetcoltype[|]{|m{0.5cm}}{m{0.5cm}}
\def\tcatbeginrow{\rule[-0.2cm]{0pt}{0.3cm}}
\edefcellcode{}{}{%
    \ifodd\numexpr\tccol+\tclin\relax
        \noexpand\cellcolor{lightgray}\fi
}
\htablecalc[7]{}{x=1,2,3,4,5,6,7}
    {}{x}{}{x}{}{x}{}{x}{}{x}{}{x}{}{x}
```



5 Exporter un tableau dans un fichier

Aussi personnalisable qu'il soit, tabularcalc ne peut couvrir tous les cas, et des ajustements fins sont parfois nécessaires dans les tableaux générés. La commande \tcwritetofile{⟨*filename*⟩} admet un argument obligatoire qui est un nom de fichier sans l'extension. Le première instruction \htablecalc ou \vtablecalc qui suit cette commande ne produira pas l'affichage d'un tableau, mais un fichier ⟨*filename*⟩.tex sera créé dans le répertoire courant dont le contenu sera le code TEX du tableau.

Voyons ceci sur un exemple :

```
\tcwritetofile{mytable}
\defcellcode{}{2}{\color{blue}}
\htablecalc[2]{$x$}{x=k;k=0:4}
                {$2x$}{2*x}
                {$x^2$}{x*x}
\tcresetcellcode
```

Un fichier « `mytable.tex` » est créé dans le répertoire courant et son contenu est le code du tableau qui aurait dû être affiché :

```
\begin {tabular}{|c|*{5}{c|}}\hline
\tcatbeginrow $x$&\tcprintvalue {0}&\color {blue}\tcprintvalue {1}&\
     tcprintvalue {2}&\tcprintvalue {3}&\tcprintvalue {4}\\\hline
\tcatbeginrow $2x$&\tcprintresult {0}{0}&\color {blue}\tcprintresult
     {2}{1}&\tcprintresult {4}{2}&\tcprintresult {6}{3}&\tcprintresult
      {8}{4}\\ \hline
\tcatbeginrow $x^2$&\tcprintresult {0}{0}&\color {blue}\tcprintresult
     {1}{1}&\tcprintresult {4}{2}&\tcprintresult {9}{3}&\tcprintresult
      {16}{4}\\ \hline
\end {tabular}
```

Cela laisse à l'utilisateur le loisir de modifier ce code à sa convenance puis d'inclure ce fichier à tout moment dans le code LaTeX d'un document avec :

```
\input{mytable.tex}
```

et voici le résultat :

x	0	1	2	3	4
$2x$	0	2	4	6	8
x^2	0	1	4	9	16

6 Utiliser la notations infixée et postfixée

La notation infixée ou postfixée peut-être utilisée indifféremment puisque tabularcalc utilise `\FPeval` qui accepte les deux. Dans cet exemple, le même tableau est généré avec la notation infixée puis avec la notation postfixée. Les résultats obtenus sont évidemment strictement les mêmes puisque le moteur de calcul est le même ; seule la notation change :

```
\tcprintroundvalue{6}
\tcprintroundresult{6}
Avec la notation infix\'ee\par
\htablecalc[3]{$x=10^k$ o\`u $k\in[-3;3]$}{x=10^k;k=-3:3}
                {$\log x$}{ln(x)/ln(10)}
                {$\sqrt{x}$}{root(2,x)}
                {$\frac{1}{x}$}{1/x}

\medskip
Avec la notation postfix\'ee\par
\htablecalc[3]{$x=10^k$ o\`u $k\in[-3;3]$}{x=k 10 pow;k=-3:3}
                {$\log x$}{x ln 10 ln div}
                {$\sqrt{x}$}{2 x root}
                {$\frac{1}{x}$}{1 x div}
```

Avec la notation infixée

$x = 10^k$ où $k \in [-3;3]$	0,001	0,01	0,1	1	10	100	1 000
$\log x$	-3	-2	-1	0	1	2	3
$\sqrt{x}$	0,031 623	0,1	0,316 228	1	3,162 278	10	31,622 777
$\frac{1}{x}$	1 000	100	10	1	0,1	0,01	0,001

Avec la notation postfixée

$x = 10^k$ où $k \in [-3;3]$	0,001	0,01	0,1	1	10	100	1 000
$\log x$	-3	-2	-1	0	1	2	3
$\sqrt{x}$	0,031 623	0,1	0,316 228	1	3,162 278	10	31,622 777
$\frac{1}{x}$	1 000	100	10	1	0,1	0,01	0,001

On préfèrera — dans la mesure où on la maîtrise — la notation postfixée qui permet bien souvent d'économiser du temps de calcul. Ainsi, le calcul $\cos x(1-\cos x)$ s'écrit ainsi en notation infixée

```
1 cos(x)*(1-cos(x))
```

Ce qui fait que $\cos x$ est calculé deux fois inutilement.

Avec la notation postfixée, on ne le calcule qu'une seule fois :

```
1 x cos copy 1 swap sub mul
```

⋆
⋆ ⋆

C'est tout, j'espère que cette extension vous sera utile !
Merci de me signaler par email tout bug ou toute proposition d'amélioration...

Christian Tellechea