# tabularcalc

**v0.1**

**Manuel de l'utilisateur**

Christian Tellechea
unbonpetit@gmail.com

19 mars 2009

***Résumé***

Étant donné une liste de nombres et une ou plusieurs formules à une variable, cette extention, 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 1 | 2,236 05 | 1 | 2,462 2 | 7,071 06 |

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



# 1 Présentation

Ce package tabularcalc nécessite LaTeX, et charge les packages `pgfmath`, `xstring` et `numprint` si ce n'a pas été le cas. Il permet de construire facilement des tableaux de résultats en évaluant des formules dont la variable prend une liste donnée de valeurs. Les tableaux sont affichés avec l'environement standard `tabular`[1]. L'affichage des résultats se fait sous forme décimale.
Le package est appelé avec la traditionnelle instruction `\usepackage{tabularcalc}`

Ce package n'est pas du tout destiné à entrer en concurrence avec `pgfplotstable`, l'excellent package de Christian Feuersänger. Ce dernier est en effet bien plus personnalisable que tabularcalc et ses possibilités sont largement plus étendues, 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 (du moins pour les personnalisations cosmétiques).

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 le package tabularcalc ne réalise pas. Pour cette tâche, il fait appel à un moteur de calcul qui est « pgfmath » par défaut. C'est pourquoi `pgfmath` est chargé s'il ne l'a pas été auparavant. On peut cependant changer de moteur de calcul, voir page 9.

En ce qui concerne l'affichage des nombres décimaux dans le tableau, de l'avis de l'auteur, rien ne le fait mieux que le package `numprint`, ce package est donc également requis. On peut changer le moteur d'affichage des nombres décimaux et/ou entièrement personnaliser l'affichage des nombres (voir page 7).

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. Pour l'instant, l'environement `tabular` est codé en dur dans le package. Rien n'empêchera par la suite de laisser le choix de l'environnement à l'utilisateur `tabularx`, `tabulary`, `supertabular`, etc.

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[`⟨*nombre*⟩`]{`⟨*cellule (0,0)*⟩`}{`⟨*liste val*⟩`}`
`{`⟨*label 1*⟩`}{`⟨*formule 1*⟩`}`
`{`⟨*label 2*⟩`}{`⟨*formule 2*⟩`}`
...
`{`⟨*label n*⟩`}{`⟨*formule n*⟩`}`

où :

- ⟨*nombre*⟩ est le nombre de formules à évaluer. Ce nombre vaut 1 par défaut ;
- ⟨*cell(0,0)*⟩ est le contenu de la cellule (0,0) ;
- ⟨*liste val*⟩ est la liste des valeurs, séparées par une virgule. Noter que les valeurs décimales doivent avoir le point comme séparateur décimal. Si deux virgules se suivent, une colonne vide est affiché ;
- ⟨*label i*⟩ est le ie label ;
- ⟨*formule i*⟩ est la ie formule qui servira à évaluer les résultats de la ie ligne. Dans les formules, `x` est la variable.

La lettre `x` qui représente la variable dans les formules est le développement de `\numberletter` et pour changer la variable dans les formules en `y`, il suffit d'écrire : `\def\numberletter{y}`.

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

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

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

| $x$ | $-4$ | $-2$ | 0 | 2,25 | 7 |
|---|---|---|---|---|---|
| $f(x) = 2x - 3$ | $-11{,}0$ | $-7{,}0$ | $-3{,}0$ | 1,5 | 11,0 |
| $x^2$ | 16,0 | 4,0 | 0,0 | 5,062 5 | 49,0 |
| $h(x) = \sqrt{x^2 + 1}$ | 4,123 1 | 2,236 05 | 1,0 | 2,462 2 | 7,071 06 |

On peut observer que le tableau n'est pas strictement identique à celui de la permière page : les résultats entiers comportent une décimale nulle, 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[⟨nombre⟩]{⟨cellule (0,0)⟩}{⟨liste val⟩}
                     {⟨label 1⟩}{⟨formule 1⟩}
                     {⟨label 2⟩}{⟨formule 2⟩}
                     ...
                     {⟨label n⟩}{⟨formule n⟩}
```

où :

- ⟨*nombre*⟩ est le nombre de formules à évaluer. Ce nombre vaut 1 par défaut ;
- ⟨*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 colonne. Dans les formules, `x` est la variable.

À titre d'exemple, voici le tableau précédent présenté verticalement :

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

| $x$ | $f(x) = 2x - 3$ | $x^2$ | $h(x) = \sqrt{x^2 + 1}$ |
|---|---|---|---|
| $-4$ | $-11{,}0$ | 16,0 | 4,123 1 |
| $-2$ | $-7{,}0$ | 4,0 | 2,236 05 |
| 0 | $-3{,}0$ | 0,0 | 1,0 |
| 2,25 | 1,5 | 5,062 5 | 2,462 2 |
| 7 | 11,0 | 49,0 | 7,071 06 |

## 2.3 Masquer des cellules

On peut masquer le contenu de n'importe quelle celule, 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 2e et la 5e valeur :

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

| $x$ | $-4$ | | 0 | 2,25 | |
|---|---|---|---|---|---|
| $f(x) = 2x - 3$ | $-11{,}0$ | $-7{,}0$ | $-3{,}0$ | 1,5 | 11,0 |
| $x^2$ | 16,0 | 4,0 | 0,0 | 5,062 5 | 49,0 |
| $h(x) = \sqrt{x^2 + 1}$ | 4,123 1 | 2,236 05 | 1,0 | 2,462 2 | 7,071 06 |

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

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

| $x$ | $-4$ | $-2$ | 0 | 2,25 | 7 |
|---|---|---|---|---|---|
| $f(x) = 2x - 3$ | $-11{,}0$ | $-7{,}0$ | | | 11,0 |
| $x^2$ | | 4,0 | 0,0 | | |
| $h(x) = \sqrt{x^2 + 1}$ | 4,123 1 | 2,236 05 | | | |

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 `\startline` est exécutée.
Par défaut, cette commande est définie par : `\def\startline{\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\startline{%
    {\bfseries\number\tclin)\ }%
}
\htablecalc[3]{$x$}{-4,-2,0,2.25,7}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{sqrt(x*x+1)}
```

| **0)** $x$ | $-4$ | $-2$ | 0 | 2,25 | 7 |
|---|---|---|---|---|---|
| **1)** $f(x) = 2x - 3$ | $-11{,}0$ | $-7{,}0$ | $-3{,}0$ | 1,5 | 11,0 |
| **2)** $x^2$ | 16,0 | 4,0 | 0,0 | 5,062 5 | 49,0 |
| **3)** $h(x) = \sqrt{x^2 + 1}$ | 4,123 1 | 2,236 05 | 1,0 | 2,462 2 | 7,071 06 |

Ici, on supprime le strut (on donne 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 `\sethrule` 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 :
\sethrule{⟨*filet 1*⟩}{⟨*filet 2*⟩}{⟨*autres filets*⟩}

Par défaut, les 3 arguments valent \hline.

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

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

| $x$ | $-2$ | $-1$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| $2x$ | $-4{,}0$ | $-2{,}0$ | 0,0 | 2,0 | 4,0 | 6,0 |
| $3x$ | $-6{,}0$ | $-3{,}0$ | 0,0 | 3,0 | 6,0 | 9,0 |
| $4x$ | $-8{,}0$ | $-4{,}0$ | 0,0 | 4,0 | 8,0 | 12,0 |

La commande \resethrule 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 :
\setcoltype[⟨*filets de droite*⟩]{⟨*type 1*⟩}{⟨*type 2*⟩}

Voici un exemple :

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

| $x$ | $-2$ | $-1$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| $2x$ | $-4{,}0$ | $-2{,}0$ | 0,0 | 2,0 | 4,0 | 6,0 |
| $3x$ | $-6{,}0$ | $-3{,}0$ | 0,0 | 3,0 | 6,0 | 9,0 |
| $4x$ | $-8{,}0$ | $-4{,}0$ | 0,0 | 4,0 | 8,0 | 12,0 |

La commande \resetcoltype 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 » du package 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}
\setcoltype{|r|}{>{\centering\arraybackslash}m{1.5cm}|}
\htablecalc[3]{$x$}{-4,-2,0,2.25,7}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{sqrt(x*x+1)}
```

| $x$ | $-4$ | $-2$ | 0 | 2,25 | 7 |
|---|---|---|---|---|---|
| $f(x) = 2x - 3$ | $-11{,}0$ | $-7{,}0$ | $-3{,}0$ | 1,5 | 11,0 |
| $x^2$ | 16,0 | 4,0 | 0,0 | 5,062 5 | 49,0 |
| $h(x) = \sqrt{x^2 + 1}$ | 4,123 1 | 2,236 05 | 1,0 | 2,462 2 | 7,071 06 |

# 3 Personnalisation avancée

## 3.1 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, tandisque 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 du package 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]{$x$}{-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

| $x$ | $-2$ | $-1$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| $2x$ | $-4{,}0$ | $-2{,}0$ | 0,0 | 2,0 | 4,0 | 6,0 |
| $3x$ | $-6{,}0$ | $-3{,}0$ | 0,0 | 3,0 | 6,0 | 9,0 |
| $4x$ | $-8{,}0$ | $-4{,}0$ | 0,0 | 4,0 | 8,0 | 12,0 |

On peut observer que la cellule (1 , 4) qui contient 2,0 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` [2]. 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]{$x$}{-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

| $x$ | $-2$ | $-1$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| $2x$ | $-4{,}0$ | $-2{,}0$ | 0,0 | 2,0 | 4,0 | 6,0 |
| $3x$ | $-6{,}0$ | $-3{,}0$ | 0,0 | 3,0 | 6,0 | 9,0 |
| $4x$ | $-8{,}0$ | $-4{,}0$ | 0,0 | 4,0 | 8,0 | 12,0 |

## 3.2 Personnaliser l'affichage

### 3.2.1 Les macros `\printvalue` et `\printresult`

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

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

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

Pour afficher un résultat, la commande `\printresult` est appelée. Elle admet **deux** arguments ; le premier est le résultat provenant de `pgfcalc` 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, `\printresult` est définie par le code suivant :

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

On peut observer que l'argument `#2` (la valeur) n'est pas exploitée par `\printresult`. 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 :

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

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

| longueur | 0,7 | −10 | 3 | −2 | 5 | 12 |
|---|---|---|---|---|---|---|
| aire du carré | 0,489 99 $\mathrm{cm}^2$ | X | 9,0 $\mathrm{cm}^2$ | X | 25,0 $\mathrm{cm}^2$ | 144,0 $\mathrm{cm}^2$ |

Une remarque : l'aire du carré de 0,7 cm de côté est légèrement fausse et devrait être 0,49 $\mathrm{cm}^2$ au lieu de 0,489 99 $\mathrm{cm}^2$ ! Le package `pgfmath` n'est pas destiné à faire du calcul scientifique mais du dessin ce qui explique pourquoi la précision est des résultats est limitée au $\frac{1}{100\ 000}$, et les résultats sont souvent entâchés d'erreurs d'arrondis comme celui-ci.

### 3.2.2 Gérer les arrondis

En ce qui concerne l'affichage des nombres entiers, le package `pgfmath` quoiqu'excellent, mais il a un défaut irritant : lorsque le résultat du calcul est un entier, il renvoie un nombre brut ayant 1 décimale nulle, comme on le voit dans le tableau ci-dessus. On peut donc tester si le résultat est un entier dans `\printresult`. Pour cela, on peut par exemple se servir de la commande `\IfInteger` du package `xstring` et ne transmettre à `\numprint` la valeur du compteur `\integerpart` qui est la partie entière du nombre testé :

```
\def\printresult#1#2{%
    \IfInteger{#1}%
        {\numprint{\number\integerpart}}%
        {\numprint{#1}}%
}
\htablecalc{$x$}{-3,1.56,2.5,3.608}{$2x$}{2*x}
```

| $x$ | −3 | 1,56 | 2,5 | 3,608 |
|---|---|---|---|---|
| $2x$ | −6 | 3,12 | 5 | 7,216 |

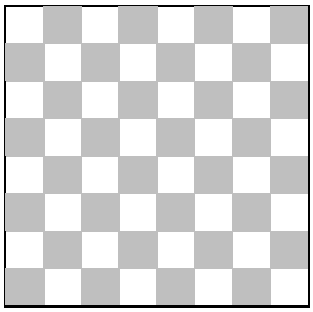
On peut toujours demander à `numprint` d'afficher un certain nombre de chiffres après la virgule avec la commande `\nprounddigits` dont l'argument est le nombre de chiffres de la partie décimale. Mais des 0 inutiles sont rajoutés si nécessaire.

### 3.2.3 Pour le fun

On peut imaginer des détournements de ce package, 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 2 pour ne pas fausser la dimension voulue de 0,5 cm ;
- l'affichage de toutes les valeurs et résultats est neutralisé à la ligne 3 ;
- ensuite on ne dessine que les filets horizontaux du haut et du bas du tableau (ligne 4), et les filets de gauche et droite (ligne 5) ;
- on définit un strut de hauteur 0,5 cm au début de chaque ligne (ligne 6) ;
- et on teste si la somme de la ligne et de la colonne en cours est impaire (ligne 8) auquel cas, on colore la case en gris (ligne 9).

```
\usepackage{colortbl,xcolor}
\arraycolsep=0pt\tabcolsep=0pt
\def\printvalue#1{}\def\printresult#1#2{}
\sethrule{\hline}{}{}
\setcoltype[|]{|m{0.5cm}}{m{0.5cm}}
\def\startline{\rule[-0.2cm]{0pt}{0.3cm}}
\edefcellcode{}{}{%
    \ifodd\numexpr\tccol+\tclin\relax
        \noexpand\cellcolor{lightgray}\fi
}
\htablecalc[7]{}{ , , , , , , }
    {}{}{}{}{}{}{}{}{}{}{}{}{}{}
```



# 4 Changer de moteur de calcul

`pgfmath`, le moteur de calcul mathématique de `pgf` est utilisé par défaut, mais on peut en choisir un autre, bien qu'il n'y ait pas grand choix : à ma connaissance, il existe les packages `fp` et `xlop` qui effectuent des calculs mathématiques sous LaTeX. Ils ont chacun des inconvénients, c'est pour cela qu'ils ne sont pas sélectionnés par défaut :

- « `fp` » utilise la notation polonaise inverse ce qui est peu habituel. On peut cependant utiliser la notation infixe mais le signe d'opposé « − » devant un nombre n'est pas géré : très embêtant pour les valeurs négatives ;
- « `xlop` » n'est pas seulement un package de calcul, il affiche aussi les résultats. De plus, il ne dispose pour l'instant que des opérations arithmétiques.

La macro `\tc@evalexpr` est chargé d'effectuer les calculs. Elle n'admet pas d'argument et son fonctionnement est très simple : elle prend `\tc@currentresult` dont le développement est une expression numérique à évaluer, par exemple « `4*2.5*2.5-3*2.5-7` ». Elle fait appel à un moteur de calcul qui évalue cette expression (ici, cela donnerait `10.5`), et enfin, elle assigne ce résultat à `\tc@currentresult`. Par défaut, la macro `\tc@evalexpr` est définie par le code suivant :

```
\def\tc@evalexpr{%
    \expandafter\pgfmathparse
    \expandafter{\tc@currentresult}%
    \let\tc@currentresult\pgfmathresult}
```

Mettons que l'on choisisse le package `fp` pour exécuter nos calculs : redéfinissons `\tc@evalexpr` :

```
\makeatletter
\def\tc@evalexpr{%
    \expandafter\FPeval\expandafter
    \tc@currentresult\expandafter{\tc@currentresult}%
    \FPclip\tc@currentresult\tc@currentresult
}
\makeatother
```

On a utilisé la commande `\FPclip` pour enlever tous les zéros inutiles du résultat (voir la documentation de `fp`).
Ça y est, le moteur de calcul de tabularcalc est maintenant `fp` :

```
\usepackage{fp}
\htablecalc{$x$}{0.5,1,2.5,10}
    {$2x^2-\frac{1}{2}$}{2*x*x-0.5}
```

| $x$ | 0,5 | 1 | 2,5 | 10 |
|---|---|---|---|---|
| $2x^2 - \frac{1}{2}$ | 0 | 1,5 | 12 | 199,5 |

Le gros problème est que `fp` ne gère pas les signes $-$ devant les nombres comme étant un signe d'opposé, et une valeur de « `-3` » aurait fait planter `fp` ! On peut éviter cette limitation en utilisant la notation polonaise inverse mais il faut à nouveau modifier `\tc@evalexpr` pour lui dire de faire le calcul en notation polonaise inverse (d'où l'utilisation de `\FPupn`), et faire quelques adaptations dans `\printvalue` évaluer les valeurs qui sont aussi écrites en notation polonaise inverse (on écrit `3 neg` pour `-3`) :

```
\usepackage{fp}
\makeatletter
\def\tc@evalexpr{%
    \expandafter\FPupn\expandafter
    \tc@currentresult\expandafter{\tc@currentresult}%
    \FPclip\tc@currentresult\tc@currentresult
}
\makeatother
\def\printvalue#1{%
    \FPupn\tempval{#1}%
    \FPclip\tempval\tempval
    \numprint\tempval
}
\htablecalc{$x$}{3 neg,0.5,1,2.5,10}
    {$2x^2-\frac{1}{2}$}{x x mul 2 mul 0.5 sub}
```

| $x$ | $-3$ | 0,5 | 1 | 2,5 | 10 |
|---|---|---|---|---|---|
| $2x^2 - \frac{1}{2}$ | 17,5 | 0 | 1,5 | 12 | 199,5 |

⋆
⋆ ⋆

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

Christian Tellechea