LaTeX, un peu, beaucoup

Pascal Dupont

Mots clés : LaTeX, tableaux

6. Tableaux

À moins d'écrire de la pure littérature, contes, romans ou essais, tout utilisateur d'un logiciel de traitement de texte va éprouver, à un moment ou à un autre, le besoin de composer un tableau : un tarif dans un feuillet publicitaire, un mois de calendrier, ...

Dans les textes mathématiques que nous rédigeons à l'aide de LaTeX, la nécessité de pouvoir, facilement, encoder des tableaux est bien plus présente encore, car aux besoins « généralistes » s'ajoutent des constructions spécifiques : tableaux de signes et de variations des fonctions, matrices, systèmes d'équations, fonctions définies par morceaux, ...

Dans cet article, nous décrirons la construction de tableaux assez simples ; nous reviendrons sur des aspects plus élaborés de la question dans un article ultérieur.

LaTeX nous propose, à ces fins, deux environnements : *tabular*, destiné aux tableaux que nous créerons en mode texte et *array* pour les tableaux en mode mathématique ([1]). Comme d'habitude, nous entrons dans l'environnement par une commande `\begin{tabular}` ou `\begin{array}`, et nous en sortons par la commande `\end{...}` correspondante.

Considérations préalables

Commençons par mentionner un certain nombre de caractéristiques communes aux environnements *tabular* et *array*.

- Ces deux environnements ne font que *créer* le tableau ; si nous voulons qu'un *tabular* soit placé seul, centré sur une ligne isolée du texte, nous aurons à l'imbriquer dans un environnement *center* ; la commande `\begin{array}` ne commande pas le passage en mode mathématique : celui-ci devra avoir été sélectionné préalablement ; si nous voulons créer une matrice, nous devrons placer des parenthèses (ou des crochets droits, c'est une question de choix) autour de notre *array*, ce que nous ferons avec les commandes `\left(` et `\right)` pour que la taille des parenthèses s'adapte automatiquement à celle du tableau.

- Les tableaux n'ont pas de dimensions prédéfinies : leur hauteur et leur largeur dépendra de ce que nous y placerons.

- Par défaut, ils sont centrés verticalement sur la ligne d'écriture ; si nous voulons que ce soit leur première (resp. dernière) ligne qui se trouve sur la ligne d'écriture, nous utiliserons l'argument optionnel `[t]` (resp. `[b]`) immédiatement après le `\begin{tabular}`/`\begin{array}` qui crée le tableau.

- Ils sont construits ligne par ligne, et les lignes sont séparées par une double controblique \\. Il ne faut pas placer de double controblique après la dernière ligne du tableau (cela aurait pour effet de terminer celui-ci par une ligne vide).

- À l'intérieur de chaque ligne, le séparateur de colonnes est le symbole `&`. Entre deux esperluètes ou entre une esperluète et la double controblique de fin de ligne, ou entre l'ouverture du tableau et la première esperluète, ou..., se trouve donc une « cellule » (ou « case ») du tableau ; celle-ci constitue un « bloc » au sens de LaTeX, comme si elle était contenue dans une paire d'accolades ; cela signifie que toute déclaration (p. ex. `\Large`) que nous faisons dans une cellule cesse ses effets lorsque nous changeons de cellule.

- Outre l'éventuel argument optionnel, les environnements `\begin{tabular}`/`\begin{array}` attendent un argument qui, principalement, leur indiquera combien de colonnes comptera le tableau, mais leur donnera aussi des indications de formatage. Nous l'appellerons le *descriptif de formatage* ;

([1]) Il en existe en vérité un troisième, *tabular**, une variante de *tabular*, qui crée un tableau de largeur prédéfinie ; nous ne le décrirons pas ici.

sa construction sera expliquée dans la section suivante.

• Les lignes du tableau peuvent être séparées par des filets horizontaux ; il suffit de placer la commande `\hline` *immédiatement* après la double controblique qui commande le passage à la ligne. Un filet horizontal avant la première ligne s'obtient en plaçant `\hline` *immédiatement* après le descriptif de formatage ; un filet horizontal après la dernière ligne nécessite que celle-ci, par exception, soit suivie d'une double controblique. Ces filets horizontaux peuvent être doublés : il suffit de répéter la commande `\hline`.

La commande `\cline{`i`-`j`}` produira un filet horizontal s'étendant sur les colonnes i à j.

Formatage des colonnes

• Par colonne que devra comporter notre tableau, il faudra que le descriptif de formatage comprenne une des lettres `c`, pour une colonne centrée, `l`, pour une colonne justifiée à gauche, ou `r`, pour une colonne justifiée à droite.

• Entre ces lettres peuvent se trouver des barres verticales `|` pour indiquer que les colonnes en question doivent être séparées par des filets verticaux. Deux de ces barres verticales produisent un filet double.

Exemple

Huit	Un	Six
Trois	Cinq	Sept
Quatre	Neuf	Deux

et

$f(x)$	$f'(x)$
e^x	e^x
$\sin x$	$\cos x$
$\cos x$	$-\sin x$

Exemple

seront codés comme suit :

```
\begin{tabular}{|rcl|}
\hline
Huit&Un&Six\\
Trois&Cinq&Sept\\
Quatre&Neuf&Deux\\
\hline
\end{tabular}
```

et

```
$\begin{array}{l||l}
f(x)&f'(x)\\
\hline
e^x&e^x\\
\sin x&\cos x\\
\cos x&-\sin x
\end{array}$
```

• Un quatrième symbole peut se trouver dans le descriptif de formatage pour représenter une colonne : `p{`*`dim`*`}`, où *`dim`* est une dimension ; ceci signifie une colonne de largeur fixée, dans laquelle le texte sera composé comme s'il était placé dans une *parbox* de largeur *`dim`*, en allant à la ligne et en justifiant à gauche et à droite ; en particulier, que ce soit dans un *tabular* ou dans un *array*, le contenu sera composé en mode texte et non en mode mathématique. Il est à noter toutefois qu'une cellule formatée de cette manière ne peut contenir de commande de fin de ligne (\\), sauf si le contenu de la cellule est *explicitement* placé dans une *parbox*.

Exemple

10 plus	Dix, onze, douze, treize, quatorze, quinze, seize, dix-sept, dix-huit, dix-neuf
10 fois	Dix, vingt, trente, quarante, cinquante, soixante, septante, huitante, nonante
$10+n$	Dix, onze, douze, treize, quatorze, quinze, seize, dix-sept, dix-huit, dix-neuf
$10n$	Dix, vingt, trente, quarante, cinquante, soixante, septante, huitante, nonante

Exemple

Voici les codages de ces deux exemples :

```
\begin{tabular}{rp{67mm}}
10 plus&
Dix, onze, douze, treize, quatorze, quinze,
seize, dix-sept, dix-huit, dix-neuf\\
10 fois&
Dix, vingt, trente, quarante, cinquante,
soixante, septante, huitante, nonante
\end{tabular}
```

et

```
$\begin{array}{rp{67mm}}
10+n&
Dix, onze, douze, treize, quatorze, quinze,
seize, dix-sept, dix-huit, dix-neuf\\
10n&
Dix, vingt, trente, quarante, cinquante,
soixante, septante, huitante, nonante
\end{array}$
```

Dans les deux cas, la deuxième colonne se voit attribuer une largeur de 67 mm.

• Le groupe `@{`*`texte`*`}`, dans le descriptif de formatage, signifie que, dans chacune des lignes du

tableau, *texte* viendra s'intercaler entre les deux colonnes correspondantes (ou avant la première colonne, ou après la dernière) ; ce *texte* sera interpété en mode texte dans un *tabular* et en mode mathématique dans un *array*. On notera aussi que ce texte *remplace* l'espace normalement présent entre les colonnes ; un usage possible de ceci est donc de supprimer tout espace entre deux colonnes, en intercalant `@{}`.

Voici par exemple comment définir une fonction par morceaux :

Exemple

$$|x| = \begin{cases} x \text{ si } x \geqslant 0, \\ -x \text{ si } x < 0. \end{cases}$$

Exemple

Nous avons codé

```
\[|x|=
\left\{\begin{array}{r@{\text{ si }}l}
x&x\geqslant0,\\
-x&x<0.
\end{array}\right.\]
```

Notons la commande `\text`, sans laquelle le « si » serait traité en mode mathématique, puisque nous sommes dans un *array* ; l'argument de cette commande inclut les blancs précédant et suivant le « si », sans lesquels celui-ci serait collé aux x qui l'entourent, puisque le `@{...}` supprime tout espace entre les colonnes. Par ailleurs, cet *array* est encadré par deux *délimiteurs*, c'est-à-dire deux symboles dont la taille s'adapte automatiquement à leur contenu : ici, une accolade à gauche, codée `\{`, et le délimiteur vide à droite, ce qui est indiqué par le point ; le rôle de délimiteur est attribué par les commandes `\left` et `\right` qui les précèdent, et qui doivent obligatoirement être utilisées en couple.

• Enfin, si le descriptif de formatage comprend une partie répétitive, il est possible de l'abréger en codant `*{`n`}{`*sous-ensemble*`}`, qui équivaut à n répétitions de *sous-ensemble*. Comme d'habitude, si n est formé d'un seul chiffre, il est permis de ne pas l'entourer d'accolades, et de même avec *sous-ensemble* s'il est un symbole unique ; par exemple, `*6c` équivaut à `cccccc` ; *sous-ensemble* peut lui-même contenir des expressions avec `*`.

Fusion de cellules

Il est possible de fusionner plusieurs cellules contiguës d'une même ligne en utilisant la commande `\multicolumn{`n`}{`*descr*`}{`*contenu*`}`. Celle-ci doit soit venir en tête de ligne soit suivre *immédiatement* un `&`. Voici son mode d'emploi. D'abord, n est le nombre de cellules à fusionner. Ensuite, *descr* est le descriptif de formatage qui remplacera celui des n colonnes concernées ; logiquement, ce descriptif doit contenir un *unique* symbole `l`, `c`, `r` ou `p{...}`, mais il peut contenir en outre des `|` ou des expressions `@{...}`. La partie remplacée du descriptif de formatage défini en tête de tableau comprend n symboles `l`, `c`, `r` ou `p{...}` et les `|` ou `@{...}` qui les suivent, et, *uniquement dans le cas de la première colonne*, les `|` ou `@{... }` qui la précèdent. Par exemple, si le descriptif de formatage du tableau est `|rl|rl|`, l'instruction `\multicolumn2c{`*contenu*`}` dans la première cellule d'une ligne fera composer cette ligne comme si le descriptif était `crl|` ; la même instruction dans la deuxième cellule d'une ligne (après le premier `&`, donc) fera composer cette ligne comme si le descriptif était `|rcl|` la même instruction dans la troisième cellule d'une ligne (après le second `&`) fera composer cette ligne comme si le descriptif était `|rl|c`.

Voici, en un seul exemple, deux cas typiques de l'utilisation de cette commande.

Exemple

	AM		PM	
	08–10	10:30–12:30	14–16	16:15–18:15
L	Math	Anglais	Chimie	Éd. Phys.
M	Math	Physique	Anglais	Natation
N	Math	Néerl.	—	—
J	Math	Français	Géogr.	Biologie
V	Math	Anglais	Histoire	Éd. Phys.

Exemple

Voici comment obtenir ce tableau.

```
\begin{tabular}{|*5{c|}}
\cline{2-5}
\multicolumn1{c|}{}&
\multicolumn2{c|}{AM}&
\multicolumn2{c|}{PM}\\
\cline{2-5}
\multicolumn1{c|}{}&
08--10&10\string :30--12\string :30&
14--16&16\string :15--18\string :15\\
\hline
\textbf L&Math&Anglais&Chimie&Éd. Phys.\\
\hline
\textbf M&Math&Physique&Anglais&Natation\\
\hline
```

```
\textbf N&Math&Néerl.&---&---\\
\hline
\textbf J&Math&Français&Géogr.&Biologie\\
\hline
\textbf V&Math&Anglais&Histoire&Éd. Phys.\\
\hline
\end{tabular}
```

Dans ce codage, remarquons les deux instructions `\multicolumn2{c|}{... }` dont le but est, sans surprise, de regrouper des colonnes deux par deux ; mais nous avons aussi utilisé, par deux fois, `\multicolumn1{c|}{}` pour modifier le formatage de la première cellule, dans les deux premières lignes ; en effet, si nous nous étions contenté de laisser vides ces cellules, le filet du bord gauche du tableau aurait commencé tout en haut, alors que nous souhaitions qu'il ne commence qu'à la troisième ligne.

D'autres commentaires, au passage :

- Pour que les deux-points servant de séparateur entre les heures et les minutes ne soient pas précédés d'un espace, nous les avons préfixés, dans le code, par la commande `\string` ;
- Le tiret moyen utilisé pour indiquer un intervalle de temps ou une gamme de valeurs (par exemple, il séparera la date de naissance et la date de mort, dans une notice biographique) est obtenu avec deux tirets consécutifs dans le code ;
- Le tiret long, que nous avons utilisé ici pour les cases vides, est codé `---` ; c'est ce même tiret long qui doit être utilisé comme signe de ponctuation pour isoler une incise ou encore pour pour indiquer le changement d'interlocuteur dans la transcription d'un dialogue.

Lignes trop longues ou incomplètes

Que se passe-t-il si une ligne compte plus ou moins de cellules que le descriptif de formatage n'a prévu de colonnes ?

Dans le premier cas, le compilateur s'arrêtera sur un message d'erreur « `Extra alignment tab has been changed to \cr.` ». Autrement dit, il tente de s'en sortir en imaginant que l'esperluète surnuméraire aurait dû être une double contreoblique ; ce n'est généralement pas la bonne solution, et votre tableau ne ressemblera pas à grand-chose (la cause la plus probable de votre erreur est tout simplement que vous avez oublié une double contreoblique en bout de ligne…) ; mais soit, vous êtes fixé et il vous reste à corriger.

Dans le second cas, ce n'est pas grave ; c'est même probablement délibéré de votre part ; personne ne vous oblige à terminer vos lignes. Mais ce que vous avez à savoir, c'est que les `|` et les `@{...}` qui suivent les `l`, `c`, `r` ou `p{...}` inutilisés seront ignorés ; par exemple, si vous avez prévu de fermer votre *tabular* à droite par un filet vertical, celui-ci se trouvera interrompu à hauteur de la ligne incomplète.

Quelques paramètres de style

Les quatre paramètres suivants sont des dimensions ; il nous est impossible d'indiquer ici leurs valeurs par défaut, parce qu'elles dépendent du style de document et de la taille des caractères.

- `\tabcolsep` est *la moitié* de l'espace entre deux colonnes d'un *tabular* ; lorsqu'un filet vertical est placé entre deux colonnes, il y a donc un espace égal à `\tabcolsep` de part et d'autre. Attention, avant la première et après la dernière colonne, ce sont des `\tabcolsep` simples qui sont placés.

- `\arraycolsep` est son analogue pour les *array*.

- `\arrayrulewidth` est l'épaisseur des filets créés par les commandes `|`, `\vline` ([2]), `\hline` et `\cline` (aussi bien dans un *tabular* que dans un *array*). Ces filets restent toutefois présumés de largeur nulle, donc en fait leur largeur sera défalquée des espaces entre colonnes ou entre lignes.

- `\doublerulesep` est l'espace qui sépare les lignes doubles (aussi bien dans un *tabular* que dans un *array*) ; attention, c'est plus exactement la distance entre les *axes* de ces lignes ; si celles-ci sont épaisses, la différence peut être perceptible.

Pour modifier la valeur de ces dimensions, il suffit de déclarer, p. ex., `\arrayrulewidth1.5mm`.

Par ailleurs, la commande `\arraystretch` est un *facteur* qui vient multiplier l'espace normal entre les lignes d'un *tabular* ou d'un *array* ; sa valeur par défaut est 1 ; il se modifie à l'aide de l'instruction `\renewcommand` ([3]) : par exemple, `\renewcommand{\arraystretch}{1.5}` aura pour effet d'augmenter de 50 % l'espace entre deux lignes du tableau. Il est à noter que les commandes de

([2]) La commande `\vline`, placée dans une cellule du tableau, crée à l'endroit où elle se trouve un filet vertical dont la hauteur est exactement celle de cette cellule

([3]) Présentée dans le n° 5 de cette série (Losanges 26).

changement de ligne peuvent aussi recevoir une dimension en argument optionnel pour ajouter de l'espace entre deux lignes (p. ex. : `\\[4mm]`) ; ceci ne concerne alors qu'un seul interligne.

Toutes ces (re)définitions obéissent aux règles de localité habituelles : elles restent en vigueur jusqu'à révocation par un nouvel ordre, ou jusqu'à la fermeture de l'environnement où elles ont été proclamées.

De jolies matrices

Une fois que les *array* sont maitrisés, rien de plus simple, sans doute, que de construire des matrices. Peut-être… Mais un peu de soin est-il sans doute tout de même de mise.

Voici un exemple.

Exemple

$$M_1 = \begin{pmatrix} -1 & 0 \\ 4 & -3 \end{pmatrix}.$$

Exemple

Codage :

```
\[M_1=
\left(\begin{array}{rr}
-1&0\\
4&-3
\end{array}\right).\]
```

Pour le résultat le plus esthétique (à nos yeux), les nombres doivent être centrés verticalement, mais sans tenir compte des signes moins, visuellement légers, qui donc « ne comptent pas » dans l'alignement vertical ; nous nous en sommes sorti ci-dessus avec des colonnes alignées à droite (descripteurs `r`), parce que les éléments de la matrice ont tous le même nombre de chiffres. En revanche, dans la matrice

Exemple

$$M_2 = \begin{pmatrix} -100 & 0 \\ 4 & -30 \end{pmatrix}.$$

Exemple

la même solution ne fonctionne plus, et nous devons recourir à un artifice : nous utilisons des colonnes centrées (descripteur `c`) et les nombres positifs se voient munis d'un signe moins, mais écrit à l'encre transparente, grâce à la commande `\phantom` :

```
\[M_2=
\left(\begin{array}{cc}
-100&\phantom-0\\
\phantom-4&-30
\end{array}\right).\]
```

(Notons que `\phantom+` fournirait exactement le même résultat, car les deux signes ont la même largeur et sont régis par les mêmes règles d'espacement, et donnerait sans doute moins de sueurs froides au mathématicien !)

Mentionnons encore l'existence des environnements *matrix* (matrice « nue »), *pmatrix* (matrice avec parenthèses), *bmatrix* (matrice avec crochets) et de quelques-uns de leurs cousins. Par exemple, avec le codage

```
\[M_3=
\begin{pmatrix}
-100&\phantom-0\\\phantom-4&-30
\end{pmatrix}.\],
```

nous obtenons la variante suivante de M_2 :

Exemple

$$M_3 = \begin{pmatrix} -100 & 0 \\ 4 & -30 \end{pmatrix}.$$

Exemple

Un avantage de ces environnements est de ne pas nécessiter de descriptif de formatage, ni même d'indication du nombre de colonnes ; mais, revers de la médaille, il est impossible de disposer de colonnes autres que centrées.

Aux dires de leurs défenseurs, ces environnements offrent une meilleure gestion des espaces que dans les matrices construites comme des *array*. Nous ne partageons pas ce point de vue, et préférons les matrices plus aérées : M_2 plutôt que M_3, sauf peut-être pour les matrices à une seule colonne (et encore cela dépend-il des cas). Le lecteur se fera sa propre opinion, pour son propre usage…

Des problèmes

Les tableaux créés avec les environnements *tabular* et *array* ne sont toutefois pas parfaits (notamment, lorsque des filets doubles horizontaux et verticaux se croisent, le résultat est assez inesthétique) ; ils souffrent aussi de certaines limitations.

Mais ces problèmes étant anciens et bien connus, nombre d'entre eux sont résolus par des modules

qui augmentent les possibilités de ces deux environnements, et qui en proposent d'autres.

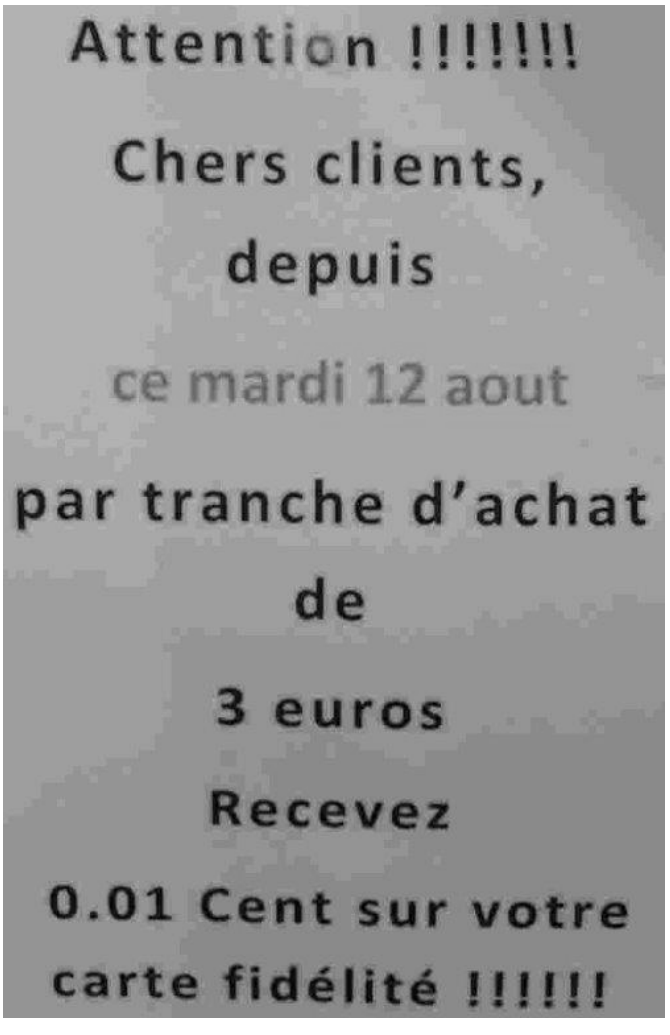
Comme annoncé plus haut, nous évoquerons quelques-unes de ces solutions dans une future rubrique.

$$\begin{array}{r|l} 63732 & 27 \\ \cline{2-2} -54 & 2360 \\ \cline{1-1} 97 & \\ -81 & \\ \cline{1-1} 163 & \\ -162 & \\ \cline{1-1} 12 & \\ 0 & \\ \cline{1-1} 12 & \end{array}$$

Exercice

Nous vous proposons de mettre à contribution votre connaissance (toute nouvelle ou éprouvée) de l'environnement *array* pour coder la « potence » suivante :

Pascal Dupont est chargé de cours à HEC•ULg. ✉ pascal.dupont@ulg.ac.be.



Photographié dans un hypermarché... © Guy Noël