Troisième partie

Deuxième atelier : tableaux et mises en page avancés

Chapitre 11

Tableaux

11.1 Tableaux et mise en page mathématique

11.1.1 Tableaux

Les tableaux

Ah ! Les tableaux ! C'est en partie à l'aisance avec laquelle un utilisateur construit un tableau compliqué qu'on reconnaît un expert de LaTeX. Autant dire qu'il s'agit là d'une notion un peu délicate à manipuler. C'est logique : construire un tableau est quelque chose de très subtil qui nécessite souvent une bonne dose de réflexion. Dans cette section, nous ne verrons que les macros définies directement par LaTeX ; quelques présentations un peu plus compliquées seront présentées lors de l'étude des extensions (Cf. section 13.2.1).

Un tableau est défini grâce à l'environnement `tabular` où le groupe qui suit le `\begin{tabular}` indique le motif du tableau (nombre et types des colonnes). Il y a quatre type de base pour les colonnes :

l indique une colonne calée à gauche (**l**eft) ;

r indique une colonne calée à droite (**r**ight) ;

c indique une colonne centrée (**c**enter) ;

p{*d***}** indique une colonne qui peut accueillir des **p**aragraphes dont la largeur est donnée par la dimension *d* (Cf. tableau 7.3 page 68 pour les unités acceptées).

Premier exemple de tableau

Nous allons commencer par un tableau très basique : 4 colonnes, une pour chaque type, sans réglures (les lignes horizontales et verticales). Le motif sera alors `lcrp{2cm}` et le tableau complet sera :

```
\begin{tabular}{lcrp{2cm}}
  contenu du tableau
\end{tabular}
```

Le *contenu du tableau* va indiquer le texte qui va apparaître dans chaque cellule du tableau. Pour passer d'une cellule à une autre, on emploie le caractère spécial `&` et pour passer à la ligne suivante la séquence de caractères `\\`.

Voici deux exemples :

g	c	d	paragraphe de deux centi-mètres
xx	yyy	zz	ppp

```
\begin{tabular}{lcrp{2cm}}
  g & c & d &
  paragraphe de deux centim{\'e}tres \\
  xx & yyy & zz & ppp
\end{tabular}
```

g	c	d	paragraphe de deux centi-mètres
xxx	yyy	zzz	pppppp

```
\begin{tabular}{lcrp{2cm}}
  g & c & d &
  {\raggedright paragraphe de
  deux centim{\'e}tres\par} \\
  xxx &
  yyy &
  zzz &
  pppppp
\end{tabular}
```

Remarquons que les deux tableaux ont exactement le même motif (`lcrp{2cm}`). Nous pouvons voir que ce motif ne permet pas d'avoir des tableaux de largeur fixe : le second tableau est plus large que le premier. Pour les colonnes de type `l`, `c` et `r` la largeur sera la largeur de la cellule la plus importante (plus les espaces intercolonnes). En revanche, le rôle même du type `p` est d'avoir une taille fixe.

Problème ...

Sans précaution particulière, le motif p produit des résultats peu esthétiques car sa largeur est généralement très faible et le paragraphe est construit par défaut pour être justifié ce qui est incompatible (voyez le piètre résultat du premier tableau).

g	c	d	paragraphe de deux centi- mètres
xx	yyy	zz	ppp

... et solution

La solution est, souvent, de forcer la composition au fer à gauche grâce à la macro `\raggedright`. N'oublions pas alors de faire agir cette macro sur un vrai paragraphe (obtenu ici grâce à la macro `\par`).

g	c	d	paragraphe de deux centi- mètres
xxx	yyy	zzz	pppppp

La présence des accolades peut sembler mystérieuse ; sans celles-là, il y aurait eu une erreur à la compilation car la macro \\ indique de passer à la ligne suivante mais il s'agit d'une commande interdite lorsqu'on se trouve entre deux paragraphes. Les accolades permettent de créer un paragraphe à l'intérieur sans qu'il soit visible à l'extérieur : tordu n'est-ce pas ? C'est le genre de petite feinte qu'il vaut mieux connaître pour ne pas perdre trop de temps lors des premiers pas sous LaTeX.

Petit détail sans trop d'importance : la dernière ligne d'un tableau n'a pas besoin de la séquence \\, l'environnement en ajoute une de façon automatique si l'utilisateur ne l'a pas spécifié. Plus important : on peut terminer une ligne avant que toutes les colonnes aient été remplies, les cellules correspondantes seront vides. Pour un tableau sans réglures, cela n'a aucune importance, mais avec des réglures, il faut faire attention à ce que l'on fait car ces dernières disparaîtront également (cela peut très bien être le résultat voulu).

Les réglures

Les réglures

Pour obtenir des tableaux qui en soient vraiment, il reste à gérer les réglures.

Les réglures verticales, celles séparant les colonnes, sont déclarées au niveau du motif avec le caractère `|`.

Les réglures horizontales sont déclarées dans le contenu du tableau avec la macro `\hline` qui fera suite à la fin de ligne.

Reprenons l'exemple précédent en ajoutant des réglures :

gauche	centré	droite	paragraphe de deux centi- mètres
xxx	yyy	zzz	ppp
xxx xxx	yyy yyy	zzz zzz	ppp ppp

```
1  \begin{tabular}{|l|c|r ||| p{2cm}|}
2    \hline
3    gauche & centr\'e & droite &
4    {\raggedright paragraphe de deux
5      centim\'etres\par}\\
6    \hline
7    xxx & yyy & zzz & ppp\\
8    \hline\hline
9    xxx xxx & yyy yyy & zzz zzz & ppp ppp\\
10   \hline
11 \end{tabular}
```

L'inclusion d'une triple ligne verticale n'est sans doute pas très heureuse : elle a été faite pour montrer que c'était possible ! De même, on peut réaliser une double réglure horizontale (ou triple, quadruple, ...) en répétant la macro `\hline`. Toujours dans le registre des points inesthétiques, on voit que le `\par` a introduit un espacement vertical parasite ; nous verrons lors du stage de perfectionnement comment y remédier.

multicolumn

multicolumn

On peut vouloir ne pas tout le temps suivre le motif par défaut du tableau : c'est extrêmement fréquent pour les lignes de titres qui peuvent être centrées alors que le reste de la colonne sera calé à gauche par exemple. De plus, il peut arriver qu'on veuille fusionner deux (ou plus) cellules adjacentes. Pour toutes ces manœuvres, la macro `\multicolumn` apporte la solution. Sa syntaxe est un peu lourde :

`\multicolumn{`*nb*`}{`*motif*`}{`*texte*`}`

où *nb* indique le nombre de colonnes devant être fusionnées (pour un changement de motif sans fusion, on prendra $nb = 1$), *motif* est le motif de remplacement pour cette cellule et *texte* est le texte qui sera placé dans cette cellule.

Voici un premier exemple moins artificiel que les précédents où on ne fait que modifier le motif pour obtenir une ligne de titre.

On remarquera le petit jonglage au niveau des formats des macros `\multicolumn` pour garder les mêmes réglures que le tableau.

Corps	Ø (km)	densité
Soleil	1 392 000	1,409
Mercure	4 840	5,50
Vénus	12 390	5,25
Terre	12 760	5,517
Mars	6 800	3,94

```
\begin{tabular}{|l|r|l|}
  \hline
  \multicolumn{1}{|c|}{Corps} &
  \multicolumn{1}{c|}{\O\ (km)} &
  \multicolumn{1}{c|}{densit\'e} \\ \hline
   Soleil   & 1\,392\,000 & 1,409 \\
   Mercure  &     4\,840 & 5,50  \\
   V\'enus  &    12\,390 & 5,25  \\
   Terre    &   12\,760 & 5,517 \\
   Mars     &     6\,800 & 3,94  \\ \hline
\end{tabular}
```

La macro `\,` n'a rien à voir avec les tableaux : il s'agit d'une espace fine (on verra les différents types d'espaces à la section 13.1).

L'exemple suivant illustre l'utilisation de la macro `multicolumn` pour fusionner plusieurs cellules. En raison de la taille nécessaire, exceptionnellement, le résultat sera présenté sous le source :

```
\begin{tabular}{|l||c|c|c||c|c|c|}
  \hline
  \multicolumn{1}{|c|}{} &
  \multicolumn{6}{c|}{syst\'eme RVB} \\ \hline
   & \multicolumn{3}{c||}{couleur primaire} &
     \multicolumn{3}{c|}{couleur secondaire} \\ \hline
  nom         & rouge & vert & bleu & jaune & magenta & cyan \\ \hline
  composition & R     & V    & B    & RV    & RB      & VB   \\ \hline
\end{tabular}
```

	système RVB					
	couleur primaire			couleur secondaire		
nom	rouge	vert	bleu	jaune	magenta	cyan
composition	R	V	B	RV	RB	VB

Répétition de motif

En premier lieu, on peut remarquer que le motif est un peu répétitif et on conçoit que pour un motif devant désigner 10 cellules centrées avec une ligne verticale entre chaque cellule, l'écriture :

```
|c|c|c|c|c|c|c|c|c|c|
```

sera un peu embêtante à taper, sans compter les risques d'erreur (taper 9 ou 11 cellules au lieu de 10).

Raccourci pour la répétition

LaTeX permet un raccourci d'écriture sous la forme :

```
|*{10}{c|}
```

qui indique que le motif sera constitué d'un " | ", puis d'une répétition de 10 fois le texte " c| " ce qui donnera bien le résultat souhaité. L'astérisque est suivi de deux groupes : le premier indique le nombre de répétitions et le second groupe le sous-motif qui devra être répété. Dans notre exemple, cela vaut à peine le coup puisque il n'y a que deux répétitions. Enfin ! Pour des raisons pédagogiques, voici la traduction du motif précédent :

```
|l|*{2}{|c|c|c|}
```

Pour les esprits torturés

Un esprit torturé pourra remarquer que le motif répété est lui-même constitué d'une répétition de sous-motifs et pourra penser à écrire un motif du type :

```
|l|*{2}{|*{3}{c|}}
```

Cela marchera parfaitement mais on ne peut raisonnablement pas recommander ce genre d'acrobaties en raison de la lisibilité plus que douteuse du source pour un humain normalement constitué !

Plus important en pratique, le tableau précédent aurait pu être plus logiquement présenté sous la forme :

	système RVB					
	couleur primaire			couleur secondaire		
nom	rouge	vert	bleu	jaune	magenta	cyan
composition	R	V	B	RV	RB	VB

Pour cela, il va falloir supprimer certaines parties de quelques réglures. Pour les réglures verticales, il n'y a pas de problème particulier : il suffit d'employer `\multicolumn` en redéfinissant le motif de la cellule pour faire disparaître la réglure. Ainsi, dans l'exemple précédent, les deux cellules vides étaient définies de la façon suivante :

```
\multicolumn{1}{c|}{}
```

Donc sans réglure verticale à gauche mais quand même la réglure verticale à droite et un texte vide (on notera que le choix de `c` au lieu de `l` ou `r` n'a strictement aucune importance).

Pour les réglures horizontales, il va falloir disposer d'une autre macro que `\hline` puisque celle-ci trace une réglure sur toute la largeur du tableau. La macro magique qui permet de réaliser des réglures horizontales partielles est `\cline` suivi par un groupe indiquant sur quelles cellules adjacentes doit s'étendre la réglure ; ceci est précisé avec la syntaxe `{`*colonne_début*`-`*colonne_fin*`}` où *colonne_début* et *colonne_fin* sont les numéros de colonne dans le tableau. Dans notre exemple, on veut avoir des réglures horizontales s'étendant de la colonne 2 jusqu'à la colonne 7.

Le source complet est :

```
\begin{tabular}{|l|c|c|c||c|c|c|}
  \cline{2-7}
  \multicolumn{1}{c|}{} &
  \multicolumn{6}{c|}{syst\'eme RVB} \\ \cline{2-7}
  \multicolumn{1}{c|}{} &
  \multicolumn{3}{c||}{couleur primaire} &
  \multicolumn{3}{c|}{couleur secondaire} \\ \hline
  nom         & rouge & vert & bleu & jaune & magenta & cyan \\ \hline
```

```
  composition & R     & V    & B    & RV    & RB      & VB   \\ \hline
\end{tabular}
```

Un peu de magie

Cette section a été un peu longue mais patience, il ne reste plus qu'un point à découvrir sur la gestion de base des tableaux. Considérons le tableau suivant :

Le nombre 2	est	premier	Le nombre 3	est	premier
Le nombre 4	est non	premier	Le nombre 5	est	premier
Le nombre 6	est non	premier	Le nombre 7	est	premier
Le nombre 8	est non	premier	Le nombre 9	est non	premier
Le nombre 10	est non	premier	Le nombre 11	est	premier

Cela n'a rien de particulièrement excitant ! De plus, avec ce qui a été présenté jusqu'à maintenant, il est tout à fait possible de le composer même si cela va entraîner un côté un peu pénible.

En fait, le côté un peu magique de l'affaire et qui ne transparaît pas au niveau de la sortie est que le corps de ce tableau a été tapé sous la forme :

```
2  &     & 3  &     \\
4  & non & 5  &     \\
6  & non & 7  &     \\
8  & non & 9  & non \\
10 & non & 11 &     \\
```

On voit que seules les éléments variables ont été spécifiés. Les " colonnes " présentant toujours le même matériel (texte ou espacement) ont été spécifiées au niveau du motif. La méthode consiste à remplacer l'espace intercolonne par ce qu'on veut grâce à la syntaxe `@{`*matériel*`}`. Cela ne pose pas trop de problème, la seule chose à bien se souvenir est qu'il s'agit d'un *remplacement* de l'espace intercolonne ce qui fait que celui n'existe plus et qu'il faudra prévoir les espaces nécessaires. Voyez, par exemple, le " est " entouré de deux espaces ; les espaces suivant ou précédant le caractère `&` disparaissent au niveau de la sortie ce qui fait qu'il est préférable de les spécifier au niveau du motif si on ne veut pas à avoir à taper une ribambelle de `\␣` (macro espace).

Voici le motif qui a permis la construction du tableau précédent :

```
*{2}{@{Le nombre }l@{ est }r@{ premier\hspace{1cm}}}
```

Ceci peut être très utile pour des systèmes d'équations :

2	x + 0 y + 3	z =	5		
4	x +	y + 5	z =	0	
6	x + 3 y + 7	z =	- 2		
8	x + 7 y + 9	z =	4		
10	x + 3 y + 11	z =	- 5		

```
\begin{tabular}{l@{ x + }l@{ y + }l@{ z = }r}
  2  &   0  & 3 &   5  \\
  4  &  & 5 &   0  \\
  6  & 3 & 7 &  − 2  \\
  8  & 7 & 9 & 4 \\
  10 & 3 & 11 & − 5  \\
\end{tabular}
```

Chapitre 12

Extensions

12.1 Extensions utiles

Présentation des extensions

Ce chapitre présentera, de façon relativement succincte parfois, des extensions particulièrement utiles et assez souvent employées. Ces extensions (ou packages) font obligatoirement partie de toutes les distributions ; vous êtes donc assuré de les avoir si vous installez un système TEX sur votre ordinateur.

Les extensions présentées sont celles que l'auteur utilise. Il existe assez souvent d'autres extensions permettant d'obtenir des résultats équivalents. Choisir telle ou telle extension est affaire d'habitude, de goût, de hasard, ...

12.1.1 geometry

Présentation de geometry

Comme nous l'avons dit en début de manuel, TEX a été construit par un américain pour composer des textes sur des feuilles américaines ce qui ne correspond pas à notre bon vieux format A4. Le package *geometry* permet de ne pas se perdre dans la jungle des registres de dimension permettant de régler les différents paramètres de la page. Son utilisation est très simple et les paramètres les plus importants sont très compréhensibles.

Syntaxe de geometry

Bien évidemment, il faudra charger ce package avec la commande :

```
\usepackage{geometry}
```

au niveau du préambule. Ce package fournit l'unique macro `\geometry` qui s'utilise en indiquant ce qu'on désire dans le groupe qui la suit.

Voici la liste des possibilités les plus utiles. Chaque demande devra être séparée des autres par une virgule :

- `a4paper` indique que la page physique fait $21 \times 29,7$; il existe également tout un tas d'autres format possible (`a0paper`, ..., `a5paper`, `b0paper`, ..., `b5paper`, `letterpaper`, `legalpaper` et `executivepaper`) ;
- `twoside` indique que le document sera composé en recto-verso alors que `oneside` indique que le document sera composé en recto simple ;
- `nohead` supprime l'espace réservé à l'en-tête ; `nofoot` supprime l'espace réservé au pied de page et `noheadfoot` supprime ces deux espaces ;
- `top=`*val*, `bottom=`*val*, `right=`*val* et `left=`*val* fixent l'importance des marges (respectivement supérieure, inférieure, droite et gauche) ;
- `marginparwidth=`*val* indique la largeur des notes marginales.

Il est possible d'en faire encore plus : ces options sont les plus importantes. À titre d'exemple, voici l'appel à cette macro qui a été utilisé pour produire ce manuel :

```
\geometry{a4paper,twoside,left=1.5cm,right=1.5cm,marginparwidth=1.2cm,%
  marginparsep=3mm,top=2cm,bottom=2cm}
```

12.1.2 Extension pour les polices de caractères

times et compagnie

Si vous en avez assez des sempiternelles fontes Computer Modern, il est possible de composer des documents avec d'autres familles de fontes. Le problème est que dans le monde TEX on n'aime pas l'à peu près donc les fontes disponibles sont d'excellentes qualités et par conséquent soit rares, soit chères. Oubliez les fontes True Type qui ne correspondent de toute façon pas aux critères de qualité requis bien qu'il soit possible de transformer ce genre de fontes sous une forme utilisable par TEX.

METAFONT et les fontes PostScript

En résumé et en très gros, deux grands types de fontes sont utilisées dans le monde TEX : les fontes, telles que les Computer Modern, fabriquées à partir du programme METAFONT et les fontes PostScript. Ces fontes sont radicalement différentes. Les fontes produites à partir de METAFONT sont des dessins bitmaps (chaque caractère est décrit sous la forme d'une grille de points) alors que les fontes PostScript sont des fontes vectorielles. La plupart du temps, les fontes METAFONT appartiennent au domaine public (elles sont donc accessibles gratuitement, librement distribuables et/ou modifiables) alors que la plupart des fontes PostScript sont soumises à un copyright classique et sont payantes.

Les fontes PostScript

Les fontes PostScript offrent un vaste choix de possibilités car elles sont produites par des structures commerciales dont c'est, au moins en partie, le rôle et qui disposent d'un personnel hautement qualifié alors que les fontes METAFONT sont fabriquées par des passionnés de façon autonome sur leur temps libre. Or, produire une fonte de qualité requiert une quantité de travail énorme (j'ai essayé) et beaucoup de connaissances ce qui explique la relative rareté des fontes METAFONT, du moins celles de qualité.

La fonte Times

La gestion des fontes est une tâche assez obscure et nous n'allons certainement pas entrer dans les détails de ce qui se passe en coulisse. Des extensions permettent un minimum de travail. Par exemple, pour composer l'ensemble du document en fonte Times, il suffit de spécifier :

```
\usepackage{times}
```

au niveau du préambule et c'est tout.

Autres fontes

Les distributions proposent une vingtaine de package de ce type mais je conseille l'essai des extensions suivantes qui sont les plus complètes :

- `bookman` ;
- `newcent` ;
- `palatino` ;
- `times`.

Les fontes Dingbat

Le package `pifont` est un peu spécial car la fonte appelée renferme des symboles et parce que, contrairement aux autres extensions, il définit quelques macros permettant quelques présentations particulières. Le tableau 12.1 indique tous les symboles existant dans la fonte Dingbat accessibles grâce à la syntaxe :

`\ding{`*numéro*`}`

Les fontes de symboles

De même, on a accès facilement à une fonte dite " de symboles " avec la syntaxe :

`{\Pifont{psy}` *texte* `}`
ou
`\Pisymbol{psy}{`*numéro*`}`

Le tableau 12.2 indique l'ensemble de ces symboles

		32	33 ✁	34 ✂	35 ✃	36 ✄	37 ☎	38 ✆	39 ✇
40 ✈	41 ✉	42 ☛	43 ☞	44 ✌	45 ✍	46 ✎	47 ✏	48 ✐	49 ✑
50 ✒	51 ✓	52 ✔	53 ✕	54 ✖	55 ✗	56 ✘	57 ✙	58 ✚	59 ✛
60 ✜	61 ✝	62 ✞	63 ✟	64 ✠	65 ✡	66 ✢	67 ✣	68 ✤	69 ✥
70 ✦	71 ✧	72 ★	73 ☆	74 ✪	75 ✫	76 ✬	77 ✭	78 ✮	79 ✯
80 ✰	81 ✱	82 ✲	83 ✳	84 ✴	85 ✵	86 ✶	87 ✷	88 ✸	89 ✹
90 ✺	91 ✻	92 ✼	93 ✽	94 ✾	95 ✿	96 ❀	97 ❁	98 ❂	99 ❃
100 ❄	101 ❅	102 ❆	103 ❇	104 ❈	105 ❉	106 ❊	107 ❋	108 ●	109 ❍
110 ■	111 ❑	112 ❒	113 ❑	114 ❒	115 ▲	116 ▼	117 ◆	118 ❖	119 ◗
120 ❘	121 ❙	122 ❚	123 ❥	124 ❦	125 ❝				
	161 ❡	162 ❢	163 ❣	164 ❤	165 ❥	166 ❦	167 ❧	168 ♣	169 ♦
170 ♥	171 ♠	172 ①	173 ②	174 ③	175 ④	176 ⑤	177 ⑥	178 ⑦	179 ⑧
180 ⑨	181 ⑩	182 ❶	183 ❷	184 ❸	185 ❹	186 ❺	187 ❻	188 ❼	189 ❽
190 ❾	191 ❿	192 ①	193 ②	194 ③	195 ④	196 ⑤	197 ⑥	198 ⑦	199 ⑧
200 ⑨	201 ⑩	202 ❶	203 ❷	204 ❸	205 ❹	206 ❺	207 ❻	208 ❼	209 ❽
210 ❾	211 ❿	212 ➔	213 →	214 ↔	215 ↕	216 ➘	217 ➙	218 ➚	219 ➛
220 ➜	221 ➝	222 ➞	223 ➟	224 ➠	225 ➡	226 ➢	227 ➢	228 ➤	229 ➥
230 ➦	231 ➧	232 ➨	233 ➩	234 ➪	235 ➫	236 ➬	237 ➭	238 ➮	239 ➯
	241 ➱	242 ➲	243 ➳	244 ➴	245 ➵	246 ➶	247 ➷	248 ➸	249 ➹
250 ➺	251 ➻	252 ➼	253 ➽	254 ➾					

TAB. 12.1 – Caractères de la fonte ZapfDingbat

Talon Réponse

On peut également construire des listes et des suites de symboles occupant une partie d'une ligne :

✻ petite liste charmante ;
✻ peut-être un peu niaise.

COUPON À RENVOYER

✂ ✂ ✂ ✂ ✂ ✂ ✂ ✂ ✂ ✂ ✂ ✂

La section du Canon (κατατομη κανόνος)

```
1  \parskip=4pt plus2pt minus1pt \parindent=8mm
2  \begin{dinglist}{93}
3    \item petite liste charmante ;
4    \item peut-\^etre un peu niaise.
5  \end{dinglist}
6
7  {\centering COUPON \'A RENVOYER\par}
8  \dingline{34}
9
10 \emph{La section du Canon}
11 ({\Pifont{psy}katatomh kan\'onoV})
```

				32		33	!	34	∀	35	#	36	∃	37	%	38	&	39	∍
40	(	41	)	42	∗	43	+	44	,	45	−	46	.	47	/	48	0	49	1
50	2	51	3	52	4	53	5	54	6	55	7	56	8	57	9	58	:	59	;
60	<	61	=	62	>	63	?	64	≅	65	Α	66	Β	67	Χ	68	Δ	69	Ε
70	Φ	71	Γ	72	Η	73	Ι	74	ϑ	75	Κ	76	Λ	77	Μ	78	Ν	79	Ο
80	Π	81	Θ	82	Ρ	83	Σ	84	Τ	85	Υ	86	ς	87	Ω	88	Ξ	89	Ψ
90	Ζ	91	[	92	∴	93	]	94	⊥	95	_	96	‾	97	α	98	β	99	χ
100	δ	101	ε	102	ϕ	103	γ	104	η	105	ι	106	φ	107	κ	108	λ	109	μ
110	ν	111	ο	112	π	113	θ	114	ρ	115	σ	116	τ	117	υ	118	ϖ	119	ω
120	ξ	121	ψ	122	ζ	123	{	124	|	125	}								
		161	ϒ	162	′	163	≤	164	⁄	165	∞	166	*f*	167	♣	168	♦	169	♥
170	♠	171	↔	172	←	173	↑	174	→	175	↓	176	°	177	±	178	″	179	≥
180	×	181	∝	182	∂	183	•	184	÷	185	≠	186	≡	187	≈	188	…	189	|
190	—	191	↵	192	ℵ	193	ℑ	194	ℜ	195	℘	196	⊗	197	⊕	198	∅	199	∩
200	∪	201	⊃	202	⊇	203	⊄	204	⊂	205	⊆	206	∈	207	∉	208	∠	209	∇
210	®	211	©	212	™	213	∏	214	√	215	·	216	¬	217	∧	218	∨	219	⇔
220	⇐	221	⇑	222	⇒	223	⇓	224	◊	225	⟨	226	®	227	©	228	™	229	∑
230	⎛	231	⎜	232	⎝	233	⎡	234	⎜	235	⎣	236	⎛	237	⎧	238	⎝	239	⎜
		241	⟩	242	⌠	243	⌡	244	⎮	245	⎭	246	⎞	247	⎜	248	⎭	249	⎤
250	⎜	251	⎦	252	⎞	253	⎫	254	⎭										

TAB. 12.2 – Caractères de la fonte Symbol

Chapitre 13

Mathématiques avancées

13.1 Mathématiques complexes

Mathématiques complexes

Voici le deuxième chapitre consacré aux mathématiques. Nous verrons encore quelques points supplémentaires lors de la présentation de l'extension *amsmath* à la section 13.2. Pour l'instant, nous en resterons à ce que LaTeX offre de manière standard.

13.1.1 Fontes mathématiques

Fontes mathématiques

Les textes mathématiques ne sont pas composés de la même façon que les textes courant, les macros gérant ces deux mondes sont différentes et même la façon d'agencer les lettres les unes à côté des autres n'est pas la même. Voici le mot difficile écrit en mode mathématique et en italique, la différence saute aux yeux :

mode mathématique : `\(difficile\)` $difficile$
mode texte : `\textit{difficile}` *difficile*

Les caractères sont les mêmes mais les espacements ne sont pas du tout les mêmes et les ligatures ne se font plus dans les modes mathématiques puisque pour LaTeX, la formule `\(difficile\)` est comprise comme étant en fait `\(di^3f^2cle\)` (di^3f^2cle), c'est-à-dire en tant que produit de plusieurs variables !

Les fontes mathématiques présentent les mêmes possibilités de modification que celles du texte courant mais en modifiant le nom des macros. Ainsi, certaines macros `\text...` vues à la section 7.0.4 ont un équivalent dans les modes mathématiques en changeant le " `text` " en " `math` ".

Le tableau 13.1 montre l'effet de toutes ces macros existant en mode mathématique : on voit que

Source	Résultat
`\(f\in ab\log[\alpha](\Phi_1\cup F)\)`	$f \in ab\log[\alpha](\Phi_1 \cup F)$
`\(f\in \mathbf{ab\log[\alpha](\Phi_1\cup F)}\)`	$f \in \mathbf{ab}\log[\alpha](\mathbf{\Phi_1 \cup F})$
`\(f\in \mathit{ab\log[\alpha](\Phi_1\cup F)}\)`	$f \in ab\log[\alpha](\Phi_1 \cup F)$
`\(f\in \mathrm{ab\log[\alpha](\Phi_1\cup F)}\)`	$f \in \mathrm{ab}\log[\alpha](\Phi_1 \cup \mathrm{F})$
`\(f\in \mathsf{ab\log[\alpha](\Phi_1\cup F)}\)`	$f \in \mathsf{ab}\log[\alpha](\mathsf{\Phi_1 \cup F})$
`\(f\in \mathtt{ab\log[\alpha](\Phi_1\cup F)}\)`	$f \in \mathtt{ab}\log[\alpha](\mathtt{\Phi_1 \cup F})$

TAB. 13.1 – Fontes mathématiques

seules les lettres romaines, les chiffres et les majuscules grecques sont affectés par ces macros. Les exemples montrent également la différence entre rien du tout et `\mathit{}` car, par défaut, les chiffres et les majuscules grecques ne sont pas en italique.

Problème d'exposants

TeX gère automatiquement la taille des caractères dans les formules en fonction de leurs emplacements (exposant, exposant d'exposant, composantes de fraction, etc.) et du mode hors texte ou en texte. En tout, il y a quatre tailles prédéfinies : la taille hors texte (display), texte (text), scripte (script) et sous-scripte (scriptscript). La taille peut alors être forcée grâce aux macros `\displaystyle`, `\textstyle`, `\scriptstyle` et `\scriptscriptstyle` (la macro `\displaystyle` avait été présentée brièvement à la section 5.0.2). Tout ce qui se trouve après ces macros va prendre la taille spécifiée,

à moins que des automatismes de LaTeX entrent en jeu pour réduire cette taille (fraction, exposant, ...).

Voici un exemple de leur utilisation :

Un nombre énorme : $2^{2^{2^{2^{2^{2^{2}}}}}} \approx 3{,}4 \times 10^{38}$

Un nombre énorme : $2^{2^{2^{2^{2^{2^{2}}}}}} \approx 3{,}4 \times 10^{38}$

```
Un nombre \'enorme :
\(2^{2^{2^{2^{2^{2^{2^{2}}}}}}}\approx
3{,}4\times10^{38}\)

Un nombre \'enorme :
\(2^{2^
  {\scriptstyle2^
    {\scriptstyle2^
      {\scriptstyle2^
        {\scriptstyle2^
          {\scriptstyle2^
            {\scriptstyle2}}}}}}}\approx
3{,}4\times10^{38}\)
```

ou encore, plus parlant :

Le début du développement du nombre d'or en fraction continue donne :

$$1+\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1}}}}=1{,}6$$

Le début du développement du nombre d'or en fraction continue donne :

$$1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1}}}}=1{,}6$$

```
Le d\'ebut du d\'eveloppement du nombre d'or en
fraction continue donne :
\[1+\frac{1}{1+
  \frac{1}{1+
    \frac{1}{1+
      \frac{1}{1}}}}=1{,}6\]

Le d\'ebut du d\'eveloppement du nombre d'or en
fraction continue donne :
\[1+\displaystyle\frac{1}{1+
  \displaystyle\frac{1}{1+
    \displaystyle\frac{1}{1+
      \displaystyle\frac{1}{1}}}}=1{,}6\]
```

13.1.2 Grands opérateurs

Grands opérateurs

Les grands opérateurs sont des opérateurs pouvant changer de taille en fonction du mode mathématique et s'écrivant souvent avec des indices et/ou des exposants. Le tableau 5.4 page 54 en dresse la liste complète ; nous allons maintenant voir comment les utiliser.

Regardons tout ceci avec l'opérateur `\sum`, les autres opérateurs se comportent exactement de la même façon mis à part `\int` et `\oint` sur lesquels il faudra un petit complément.

Indice et exposant d'opérateurs

Ces opérateurs peuvent avoir un indice et/ou un exposant qui seront placés de façon traditionnelle dans une formule en texte et au-dessus et au-dessous du symbole dans une formule hors texte :

On sait que $\sum_{n=0}^{\infty} \frac{1}{n!}$ peut être la définition de e.
On sait que

$$\sum_{n=0}^{\infty} \frac{1}{n!}$$

peut être la définition de e.

```
On sait que
  \(\sum_{n=0}^\infty \frac{1}{n!}\)
peut \^etre la d\'efinition de \(e\).

On sait que
\[\sum_{n=0}^\infty \frac{1}{n!}\]
peut \^etre la d\'efinition de \(e\).
```

Comme il a été précisé à la section 5.0.2, on peut être tenté d'écrire quelque chose du type : " On sait que $\sum_{n=0}^{\infty} \frac{1}{n!}$ peut être la définition de e ". Toujours pour ne pas massacrer l'interligne, cette façon de faire est fortement déconseillée ! Incidemment, le code pour la formule qui a permis cette horreur a été :

```
\(\displaystyle\sum_{n=0}^\infty \frac{1}{n!}\)
```

Indice et exposant d'intégrales

Les deux opérateurs `\int` et `\oint` ne se comportent pas comme les autres au niveau du placement des exposants et indices. En effet, il est assez peu courant de voir quelque chose du style :

$$\int_0^2 t^2\,dt$$

et même en mode hors texte, l'exposant et l'indice occupent leur position habituelle, contrairement aux autres grands opérateurs.

Par exemple :

Comme $\int_0^2 t\,dt = 2$, un triangle ...
Comme

$$\int_0^2 t\,dt = 2$$

un triangle ...

```
Comme \(\int_0^2 t\,dt=2\), un triangle \ldots
Comme
\[\int_0^2 t\,dt=2\]
un triangle \ldots
```

limits ...

Mais alors, comment $\int_0^2 t^2\,dt$ a-t-il été tapé ? En fait, on peut forcer la position des exposants et indices au-dessus et au-dessous d'un opérateur quelconque quelle que soit sa taille mathématique en le faisant suivre de la macro `\limits`. L'exemple donné ci-dessus a été composé de cette façon :

```
\(\int\limits_0^2 t^2\,dt\)
```

... et nolimits

De même, on peut forcer la position des exposants et indices à leurs positions habituelles en faisant suivre l'opérateur de la macro `\nolimits`. Par exemple :

$$\sum_{i=1}^{n} a_i = \frac{\sum_{i=1}^{n} i}{\sum_{i=1}^{n} i^2}$$

```
\[\sum\nolimits_{i=1}^n a_i
  =\frac{\sum_{i=1}^n i}{\sum_{i=1}^n i^2}\
```

13.1.3 Délimiteurs

Délimiteurs

Les délimiteurs sont des symboles destinés à encadrer des sous-formules. La liste complète des délimiteurs a été présentée au tableau 5.5 page 54. Si ces symboles n'étaient que cela, ils ne seraient guère utiles et n'auraient de sens que pour l'humain. En réalité, les délimiteurs peuvent voir leur taille modifiée de façon fine, soit manuellement, soit automatiquement. Nous ne verrons ici que la façon automatique (de toute façon plus utile) et nous renverrons le lecteur avide au stage de perfectionnement ou à des ouvrages spécialisés !

Dans la formule `\(f(x)\)`, les deux parenthèses sont des délimiteurs. Soyons logique : la parenthèse gauche est appelée " délimiteur gauche " et je vous laisse deviner comment on appelle la parenthèse droite. La sous-formule, dans le cas présent est composée de la seule lettre x. Construisons maintenant une formule avec des délimiteurs encadrant une sous-formule un peu plus complexe, par exemple :

$$f(\frac{1+x}{x}) = x^2 + (\frac{1}{x})^2$$

```
\[f(\frac{1+x}{x})=x^2+(\frac{1}{x})^2\]
```

left ... right

Le résultat est tout à fait ridicule. Les deux macros qui permet de régler automatiquement la taille des délimiteurs à la hauteur de la sous-formule sont `\left` à mettre immédiatement avant le délimiteur gauche et `\right` à mettre immédiatement avant le délimiteur droit. Le même exemple donnera alors :

$$f\left(\frac{1+x}{x}\right) = x^2 + \left(\frac{1}{x}\right)^2$$

```
\[f\left( \frac{1+x}{x} \right)
  =x^2+\left( \frac{1}{x} \right)^2\]
```

On peut imbriquer des couples `\left`... `\right` mais il faut toujours qu'un `\left` s'équilibre avec un `\right` sous peine de voir LaTeX protester avec véhémence.

13.1.4 Espacements

Espaces

Comme nous l'avons déjà dit, un espace tapé au niveau d'une formule ne sert à rien sauf à stopper un nom de macro ou bien à présenter les choses de façon lisible pour un humain. Or, il peut arriver que l'on veuille insérer des espaces au niveau d'une formule. Les macros d'espacement horizontal vues pour le texte — `\␣` (macro espace), `\hspace` — ainsi que l'espace insécable `~` fonctionnent également en mode mathématique.

Sept types d'espaces

Le tableau 13.2 montre sept types d'espaces que LaTeX permet en mode mathématique (dont quatre disponibles en mode texte) :

Macro	Signification	Exemple	Mode texte
`\!`	espace négative	‖	non
`\,`	espace fine	| |	oui
`\:`	espace moyenne	| |	non
`\;`	grande espace	| |	non
`\␣`	espace intermot	| |	oui
`\quad`	espace cadratin	| |	oui
`\qquad`	double espace cadratin	| |	oui

TAB. 13.2 – Macros d'espacements

Méfiez-vous de la racine carrée !

Le tableau 13.3 montre quelques utilisations classiques de ces différentes espaces qu'il est bon de connaître pour obtenir une lecture facile d'un texte. On arrive vraiment à de la typographie très fine et il n'est pas obligatoire de se souvenir de toutes ces présentations. D'autre part, certaines constructions sont sujettes à discussions (parfois enflammées) donc vous prenez et vous en faites ce que vous voulez ! Tous ces exemples ont été repris tels quels dans le TeXbook. Pensez-y surtout pour les intégrales multiples et méfiez-vous de la racine carrée : avec ces deux conseils, vous pourrez éviter les présentations malheureuses.

Formule	Résultat	sans espace
`\((2n)!/(n!\,(n+1)!)\)`	$(2n)!/(n!\,(n+1)!)$	$(2n)!/(n!(n+1)!)$
`\[\frac{52!}{13!\,13!\,26!}\]`	$\frac{52!}{13!\,13!\,26!}$	$\frac{52!}{13!\,13!\,26!}$
`\(\sqrt{2}\,x\)`	$\sqrt{2}\,x$	$\sqrt{2}x$
`\(\sqrt{\,\log x}\)`	$\sqrt{\log x}$	$\sqrt{\log x}$
`\(O(1/\sqrt{n}\,)\)`	$O(1/\sqrt{n}\,)$	$O(1/\sqrt{n})$
`\([\,0,1)\)`	$[\,0,1)$	$[0,1)$
`\(\log n\,(\log\log n)^2\)`	$\log n\,(\log\log n)^2$	$\log n\,(\log\log n)^2$
`\(x^2\!/2\)`	$x^2\!/2$	$x^2/2$
`\(n/\!\log n\)`	$n/\!\log n$	$n/\log n$
`\(\Gamma_{\!2}+\Delta^{\!2}\)`	$\Gamma_{\!2}+\Delta^{\!2}$	$\Gamma_2+\Delta^2$
`\(R_i{}^j{}_{\!kl}\)`	$R_i{}^j{}_{\!kl}$	$R_i{}^j{}_{kl}$
`\(\int_0^x\!\int_0^y dF(u,v)\)`	$\int_0^x\!\int_0^y dF(u,v)$	$\int_0^x\int_0^y dF(u,v)$
`\[\int\!\!\int_D dx\,dy\]`	$\iint_D dx\,dy$	$\int\int_D dxdy$

TAB. 13.3 – Exemples classiques de correction d'espacement

13.1.5 Empiler verticalement

Deux dimensions

Jusqu'ici, mises à part les fractions, les éléments d'une formule mathématique étaient composés de gauche à droite mais les mathématiciens aiment beaucoup travailler en deux dimensions et la fin de ce chapitre montre comment procéder verticalement dans une formule de math.

Barre au-dessus d'une formule

Nous avons déjà vu les accents mathématique et le tableau 5.9 page 55 en dressait la liste complète. Il existait un accent permettant de surligner une lettre (`\bar`) mais son emploi pour placer une barre au-dessus d'une formule plus longue n'est pas correct :

$\bar{x}+\bar{y}$

```
\(\bar x \bar + \bar y\)
```

Pour réaliser cela, il faut employer la macro `\overline` suivie du groupe sur lequel on veut placer la barre :

$\overline{x+y}$

```
\(\overline{x+y}\)
```

Barre sous la formule

De même, il existe la macro `\underline` qui permet de placer une barre sous la formule :

$\underline{x+y}$

```
\(\underline{x+y}\)
```

Cette dernière macro est également disponible en mode texte mais le soulignement d'un texte est généralement une mauvaise idée : en fait une mauvaise habitude qui provient du temps des machines à écrire. Mettre un texte en évidence se fait **toujours** en le composant dans une autre fonte que le texte alentour (soit en gras, soit en italique par exemple).

Accolade horizontale

Les deux macros `\overbrace` et `\underbrace` permettent respectivement de surligner ou de souligner avec une accolade horizontale à la place de la barre. Le groupe qui suit la macro indique ce qui doit être inclus au niveau de l'accolade. On peut écrire quelque chose au-dessus de l'accolade supérieure ou au-dessous de l'accolade inférieure en utilisant respectivement l'exposant ou l'indice. L'exemple suivant montre tout cela :

Pour n pair, $2^n = \overbrace{2\times\cdots\times 2}^{\frac{n}{2}} \times \underbrace{2\times\cdots\times 2}_{\frac{n}{2}}$

```
Pour \(n\) pair, \(2^n=
  \overbrace{2\times\cdots\times2}
    ^{\textstyle\frac{n}{2}}
\times
  \underbrace{2\times\cdots\times2}
    _{\textstyle\frac{n}{2}}\)
```

Mélanger accolades supérieures et inférieures

Encore plus fort, on peut mélanger des accolades supérieures et inférieures. Il n'y a que la sagesse de l'auteur qui limite les possibilités car la lisibilité devient franchement mauvaise et la frappe un peu délicate à mener sans se tromper ! Un premier exemple sage :

$2^n = \overbrace{2\times\underbrace{2\times\cdots\times2}_{n-2}\times2}^{n}$

```
\(2^n=\overbrace{2\times\underbrace{
    2\times\cdots\times2}_{n-2}\times2}
  ^{n}\)
```


Tout est permis !

Et un exemple délirant pour voir que tout est permis !

$\overbrace{\overbrace{\overbrace{2\times2}^{2}\times\overbrace{2\times2}^{2}}^{4}\times\overbrace{\overbrace{2\times2}^{2}\times\overbrace{2\times2}^{2}}^{4}}^{8}$

```
\(\overbrace{\overbrace{\overbrace
    {2\times2}^2
  \times\overbrace
    {2\times2}^2
  }^{\scriptstyle 4}
\times\overbrace{\overbrace
    {2\times2}^2
  \times\overbrace
    {2\times2}^2
  }^{\scriptstyle 4}
}^{\scriptstyle 8}\)
```

stackrel

On peut placer n'importe quoi au-dessus de n'importe quoi pour produire une relation (donc avec des espacements autour comme pour " = "). Pour cela, LaTeX propose la macro `\stackrel` (stack signifie pile et rel et là pour rappeler qu'on construit une relation). Cette macro place l'élément qui la suit immédiatement au-dessus de l'élément qui est spécifié ensuite. Il est plus sage de ne pas trop réfléchir et de placer deux groupes à la suite de cette macro.

Voici deux exemples utiles :

$\vec{x} \stackrel{\text{déf}}{=} (x_1, x_2, \ldots, x_n)$
$x \xmapsto{f} f(x)$

```
\(\vec x\stackrel{\mathrm{d\acute ef}}{=}
  (x_1,x_2,\ldots,x_n)\)

\(x\stackrel{f}{\longmapsto} f(x)\)
```

Le premier exemple montre qu'on peut être conduit assez souvent à mettre du texte véritable à l'intérieur d'une formule de math. Pour cela, il existe la macro `\mbox` qui construit le contenu de ce qui suit comme s'il s'agissait de texte ordinaire sauf qu'aucune coupure ne pourra être effectuée à ce niveau.

Ainsi, le premier exemple aurait pu être tapé sous la forme :

```
\(\vec x\stackrel{\mbox{d\'ef}}{=}(x_1,x_2,ldots,x_n)\)
```

13.1.6 Matrices et environnement array

environnement array

De façon générale, les matrices sont considérées comme des tableaux, la seule différence étant que les tableaux étaient construits en utilisant l'environnement `tabular` alors que les matrices seront construites avec l'environnement `array`. L'énorme avantage est d'être en mode mathé'matique.

Matrices

Un premier exemple pour voir que ce n'est vraiment pas différent :

$$\begin{array}{cccc} 1-\lambda & 2 & 3 & 4 \\ 2 & 3-\lambda & 4 & 1 \\ 3 & 4 & 1-\lambda & 2 \\ 4 & 1 & 2 & 3-\lambda \end{array}$$

```
\[ \begin{array}{cccc}
  1-\lambda & 2 & 3 & 4 \\
  2 & 3-\lambda & 4 & 1 \\
  3 & 4 & 1-\lambda & 2 \\
  4 & 1 & 2 & 3-\lambda \\
\end{array} \]
```

Pour les " vraies " matrices, il faudra ajouter des parenthèses sous la forme de délimiteurs extensibles (donc avec les préfixes `\left` et `\right`).

Déterminants

La forme pour les déterminants suivra le même principe en prenant des barres verticales comme délimiteurs. L'exemple précédent devient alors :

$$\left|\begin{array}{cccc} 1-\lambda & 2 & 3 & 4 \\ 2 & 3-\lambda & 4 & 1 \\ 3 & 4 & 1-\lambda & 2 \\ 4 & 1 & 2 & 3-\lambda \end{array}\right|$$

```
\[
%  \varphi(\lambda) =
\left|
  \begin{array}{cccc}
  1-\lambda & 2 & 3 & 4 \\
  2 & 3-\lambda & 4 & 1 \\
  3 & 4 & 1-\lambda & 2 \\
  4 & 1 & 2 & 3-\lambda \\
  \end{array}
\right| \]
```

délimiteur vide

LaTeX définit le délimiteur " vide " qui ne produit aucun symbole. Il est souvent utile dans certains cas faisant appel aux tableaux. Par exemple :

$$\left.\begin{array}{c} x \in E \\ \text{ou} \\ x \in F \end{array}\right\} \iff x + x^2 = 0$$

```
\[\left.\begin{array}{c}
    x\in E \\
    \quad\mbox{ou} \\
    x\in F \\
  \end{array}\right\}\iff x+x^2=0
\]
```

Un autre exemple fréquent est donné par la construction :

$$P_{i,j} = \begin{cases} 0 & \text{si } i-j \text{ est impair,} \\ i!(-1)^{(i-j)/2} & \text{si } i-j \text{ est pair.} \end{cases}$$

```
\[P_{i,j} = \left\{
  \begin{array}{ll}
    0 & \mbox{si $i-j$ est impair,}\\
    i!(-1)^{(i-j)/2} &
      \mbox{si $i-j$ est pair}.
  \end{array}\right.\]
```

Cet exemple sera repris lors de l'étude de l'extension *amsmath* à la section 13.2 page 122. Notons ici l'utilisation d'un mode mathématique en texte (le `\(i-j\)`) à l'intérieur d'un mode texte (les boîtes `\mbox`) à l'intérieur d'un mode mathématique hors texte (la formule complète) !

13.2 amsmath

L'American Mathematical Society

L'American Mathematical Society est la dépositaire du logiciel TeX. Il s'agit d'une importante association américaine de mathématiciens. Lorsque ceux-ci ont eu le programme de Knuth en leur possession, ils se sont empressé de construire toute une série de macros permettant des constructions mathématiques complexes dont ils avaient besoin. Toutes ces macros ont été réunies dans plusieurs extensions dont la plus importante est *amsmath*. Il est hors de question de présenter toutes les possibilités de ce package en raison de la profusion de possibilités mais nous verrons les possibilités offertes les plus utiles.

Listes des extensions AMS

Les extensions disponibles sont *amsmath*, *amstext*, *amsfonts*, *amssymb* et *amscd*. L'extension *amscd* permet la construction de diagrammes commutatifs, *amssymb* et *amsfonts* permettent d'avoir un accès à quelques lettres hébraïques, à quelques symboles très spécialisés, à une fonte gothique, une fonte pour l'écriture des ensembles " comme à la main ", des fontes grasses, *amstext* permet de placer du texte normal dans des formules mathématiques de façon intelligente. Tous les exemples qui suivent n'ont eu besoin que des packages *amsmath* et *amsfonts*.

Exemples

Voici d'abord quelques exemples d'utilisation de macros particulières :

$$\int_C \vec{V}\cdot d\vec{M} = \iint_D \left(\frac{\partial Q}{dx} - \frac{d\partial P}{dy}\right) dx\,dy$$

$$\overrightarrow{A(x)B(x)} = \overrightarrow{g_x}$$

$$\underline{\lim_{n\to\infty}}\, u_n = \overline{\lim_{n\to\infty}}\, u_n = \boldsymbol{a^\lambda} \iff \lim_{n\to\infty} u_n = \boldsymbol{a^\lambda}$$

```
\[\int_C \vec V\cdot d\vec M
  = \iint_D \left(\frac{\partial Q}{dx}
    -\frac{d\partial P}{dy}\right) dx\,dy
\] % int 'egrales multiples automatiques

\[\overrightarrow{A(x)B(x)}
  = \overrightarrow{g_x}\] % grandes fl 'eches

\[\varliminf_{n\to\infty} u_n
  = \varlimsup_{n\to\infty} u_n
  = \boldsymbol{a^\lambda}
  \iff \lim_{n\to\infty} u_n
  = \boldsymbol{a^\lambda}\]
```

array

Une série de macros permettent de construire des structures classiques faisant normalement appel à l'environnement `array`. Voici un exemple permettant les structures de choix :

$$|x| = \begin{cases} x & \text{si } x \geq 0 \\ -x & \text{si } x \leq 0 \end{cases}$$

```
\[|x| =
\begin{cases}
  x  & \text{si $x \geq 0$}\\
  -x & \text{si $x \leq 0$}\\
\end{cases}\]
```

Matrices

Le package *amsmath* définit cinq environnements permettant de construire des matrices (`matrix` qui ne met aucun délimiteur, `pmatrix` qui entoure la matrice avec des parenthèses, `bmatrix` avec des crochets, `vmatrix` avec des traits verticaux et `Vmatrix` avec des double traits verticaux) :

$$\begin{pmatrix} 1 & \cos x \\ -\cos x & 1 \end{pmatrix}$$

$$\begin{vmatrix} \cos(x)-X & -\sin x \\ \sin x & \cos(x)-X \end{vmatrix}$$

```
\[\begin{pmatrix}
  1 & \cos x \\
  -\cos x & 1 \\
\end{pmatrix}\]

\[\begin{vmatrix}
  \cos(x)-X & -\sin x \\
  \sin x & \cos(x)-X \\
\end{vmatrix}\]
```

13.2.1 Tableaux évolués

Tableaux évolués

Vous vous êtes peut-être aperçu que la construction de tableaux est quelquefois un peu pénible et des problèmes surgiront presqu'obligatoirement si vous vous mettez à composer des textes de façon soutenue avec LaTeX. Ces problèmes ont été évoqués sur des forums de discussion et certaines personnes, très à l'aise dans la programmation de LaTeX ont créé des packages résolvant automatiquement les problèmes rencontrés. Là aussi, il existe beaucoup d'extensions dont le but est de résoudre certains problèmes liés aux tableaux ; nous ne ferons pas vœux d'exhaustivité et nous ne présenterons que les plus classiques. Seul le package *array* sera présenté avec quelques détails, les autres packages ne seront qu'évoqués, éventuellement avec un exemple représentatif.

Commandes de motif

Le package *array* offre la possibilité de dire plus de choses au niveau du motif d'un tableau. Le tableau 13.4 donne le récapitulatif des commandes qu'on peut indiquer au niveau du motif d'un tableau.

Un petit exemple va permettre de voir ceci en pratique. Il est bien entendu hors de question de voir toutes les subtilités du package *array*. [COM] consacre une quarantaine de pages aux extensions dédiées aux tableaux (en fait, un chapitre entier) et la description du package *array* en prend une quinzaine.

Anciennes commandes	
`l`	Colonne alignée à gauche
`c`	Colonne centrée
`r`	Colonne alignée à droite
`p{`*larg.*`}`	Paragraphe de largeur *larg.* dont le haut sera aligné avec le haut de la ligne du tableau.
`{`*instr.*`}`	Remplace l'espace intercolonne par *instr.*
Ancienne commande modifiée	
`|`	Insère une réglure verticale. Contrairement à ce qui se passait avec la commande LaTeX, la largeur de la colonne sera agrandie de la largeur de cette réglure.
Nouvelles commandes	
`m{`*larg.*`}`	Paragraphe de largeur *larg.* centrée verticalement dans la ligne du tableau.
`b{`*larg.*`}`	Paragraphe de largeur *larg.* dont le bas sera aligné avec le bas de la ligne du tableau.
`>{`*instr.*`}`	Utilisée avant les commandes `l`, `r`, `c`, `p`, `m` ou `b`, elle insère *instr.* au début de la colonne.
`<{`*instr.*`}`	Utilisée après les commandes `l`, `r`, `c`, `p`, `m` ou `b`, elle insère *instr.* à la fin de la colonne.
`!{`*instr.*`}`	Peut être utilisée n'importe où comme la commande `|` sauf qu'au lieu d'obtenir une réglure verticale, c'est *instr.* qui sera inséré. En particulier, cette commande ne supprime pas l'espace intercolonne.

TAB. 13.4 – Commandes des motifs de tableaux

ABEL	Niels Henrick	Il montre l'impossibilité de résoudre les équations de degré 5 par radicaux.	1802 ap. J-C. – 1829 ap. J-C.
ALEMBERT	Jean LE ROND D'	Collaborateur à l'encyclopédie de Diderot. Il a effectué d'importants travaux sur les dérivées partielles, les nombres complexes et la mécanique	1717 ap. J-C. – 1783 ap. J-C.
APOLLONIUS	de Perge	Il a étudié les sections planes du cône (les coniques). En fait, on sait très peu de choses sur ce mathématicien.	262 av. J-C. – $\sim$ 180 av. J-C.

```
\begin{tabular}{|>{\scshape}llm{5.65cm}%
    r<{ \scshape j-c.}@{ -- }l<{ \scshape j-c.}|}
  \hline
  Abel & Niels Henrick &
  Il montre l'impossibilit\'e de r\'esoudre les \'equations de degr\'e 5 par radicaux. &
  1802 ap. & 1829 ap. \\ \hline
  Alembert & Jean \textsc{le Rond d'} &
  Collaborateur \'a l'encyclop\'edie de Diderot. Il a effectu\'e d'importants travaux sur les
  d\'eriv\'ees partielles , les nombres complexes et la m\'ecanique &
  1717 ap. & 1783 ap. \\ \hline
  Apollonius & de Perge &
  Il a \'etudi\'e les sections planes du c\^one (les coniques). En fait, on sait tr\'es peu de
  choses sur ce math\'ematicien. &
  262 av. & \(\sim\) 180 av. \\ \hline
\end{tabular}
```

tabularx

L'extension *tabularx* permet de composer des tableaux ayant une largeur totale précise. Pour cela, ce package fournit l'environnement `tabularx` qui fonctionne strictement de la même façon que le `tabular` classique de LaTeX sauf que l'appel comporte un argument supplémentaire donnant le largeur totale du tableau et qu'il existe la commande `X` qu'on peut mettre dans le motif du tableau, cette commande indiquant la ou les colonnes qui seront extensibles afin d'amener le tableau à la taille voulue. Si plusieurs colonnes sont spécifiées " `X` ", elles se partageront la place restant et seront donc de même largeur. Il existe une astuce permettant de surmonter cette limitation mais sa présentation sortirait du cadre de ce stage.

Ainsi, le tableau 13.4 commençait de cette façon :

```
\noindent
\begin{tabularx}{\linewidth}{|c|X|}
\hline
\multicolumn{2}{|c|}{\textbf{Anciennes commandes}} \\\hline
\texttt{l} & Colonne align\'ee \`a gauche \\
\texttt{c} & Colonne centr\'ee \\
\texttt{r} & Colonne align\'ee \`a droite \\
...
```

Ce qui a permis d'avoir un tableau ayant exactement la largeur du corps de la page (`\linewidth`).

supertabular

Le package *supertabular* permet de réaliser des tableaux s'étendant sur plusieurs pages. En effet, un tableau classique ne peut absolument pas franchir la frontière d'une page ce qui peut provoquer soit un grand espace vide soit un débordement dans la marge lorsque des tableaux deviennent volumineux.

dcolumn

Le package *dcolumn* permet des alignements sur le point (ou la virgule) décimal.

hhline

Le package *hhline* propose un grand choix pour la construction de réglures doubles dans les tableaux. Le principe est de remplacer la macro classique `\hline` par la macro `\hhline` qui offre beaucoup plus de possibilité. Sa syntaxe générale est :

`\hhline{`*motif*`}`

On commencera avec un petit exemple de ses possibilités et le tableau 13.5 récapitulera les commandes de motif de cette macro.

a	b	A	+
1	2	α	$-$
3	4	β	$\times$
5	6	Γ	$\div$

```
\setlength{\arrayrulewidth}{1pt}
\begin{tabular}{||cc||c|c||}
    \hhline{|t:==:t:==:t|}
    a & b &    A    & \(+\)      \\
      \hhline{|:==:|-|~||}
    1 & 2 & \(\alpha\) & \(-\)    \\
      \hhline{||--||~:=||}
    3 & 4 & \(\beta\) & \(\times\) \\
      \hhline{||--||-|~||}
    5 & 6 & \(\Gamma\) & \(\div\) \\
      \hhline{|b:==#==:b|}
\end{tabular}
```


Éléments faisant la largeur d'une cellule.	
`=`	Une double ligne horizontale.
`-`	Une ligne horizontale.
`~`	Un vide.
Éléments occupant un coin de cellule.	
`|`	Une (double) ligne horizontale coupée par une ligne verticale.
`:`	Une double ligne horizontale non coupée par une ligne verticale.
`#`	Une double ligne horizontale coupée par une double ligne verticale.
`t`	La ligne supérieure d'une double ligne horizontale.
`b`	La ligne inférieure d'une double ligne horizontale.

TAB. 13.5 – Syntaxe de la macro `\hhline`

L'exemple montre également la façon d'obtenir des réglures n'ayant pas l'épaisseur standard (0,4 point). La dimension `\arrayrulewidth` est déclarée par LaTeX, il ne s'agit pas d'un ajout de ce package.

multirow

Le dernier package que nous évoquerons ici est *multirow* qui permet de fusionner des cellules verticalement (la macro `\multicolumn` de LaTeX permettait une fusion horizontale). Ce package fournit la macro `\multirow` (original n'est-ce pas ?) qui a la syntaxe de base suivante :

`\multirow{`*nbligne*`}{`*largeur*`}{`*contenu*`}`

où *nbligne* désigne le nombre de lignes qui devront être fusionnées, *largeur* la largeur de la grande cellule créée et *contenu* le contenu de la cellule qui sera considérée comme un paragraphe (donc pourra être composée sur plusieurs ligne de texte.

Voici un exemple de tableau composé grâce à cette macro :

Planète	Satellite(s)
Terre	Lune
Mars	Phobos Deimos
Jupiter	Io Europe Ganymède Callisto 8 petits

```
\newlength{\Lmax}
\settowidth{\Lmax}{Jupiter}
\begin{tabular}{|l|l|}
  \hline
  Plan\'ete & Satellite(s) \\\hline
  Terre & Lune \\\hline
  \multirow{2}{\Lmax}{Mars} & Phobos \\
                            & Deimos \\\hline
  \multirow{5}{\Lmax}{Jupiter} & Io \\
                               & Europe \\
                               & Ganym\'ede \\
                               & Callisto \\
                               & 8 petits \\
  \hline
\end{tabular}
```

13.2.2 multicol

multicol

Le package *multicol* permet la composition de parties de document sur plusieurs colonnes. En fait, LaTeX offre un embryon de possibilité mais les restrictions sont trop fortes (à mon goût du moins)

puisque le changement de nombre de colonnes provoque un saut de page, que le nombre de colonnes est limité à 2 et que la fin d'un passage en deux colonnes n'est pas équilibrée (la colonne de droite n'est pas remplie comme la colonne de gauche). L'extension *multicol* surmonte ces trois limitations puisque le nombre de colonnes maximum autorisé est de 10 (je vois mal comment dépasser ce nombre sans obtenir un résultat illisible), qu'il est maintenant possible de mélanger des nombres de colonnes différents sur une même page et qu'on peut gérer les problèmes d'équilibre de colonnes (le défaut étant que les colonnes sont parfaitement équilibrées, y compris celles de la dernière page).

syntaxe de multicols

Une fois cette extension chargée, on change le nombre de colonnes grâce à la syntaxe :

`\begin{multicols}{`n`}`
texte ...
`\end{multicols}`

où n représente le nombre de colonnes voulues. Le seul autre point important concernant ce package est la possibilité de modifier la largeur du filet séparateur (une largeur nulle permettant une absence de filet). La dimension à modifier est `\columnseprule`. Pour mémoire, la largeur par défaut des réglures dans LaTeX (par exemple ceux des tableaux) est de 0,4 pt.