Chapitre deuxième

Mathématiques

TEX a été écrit, au départ, par un professeur de mathématiques (et informaticien de génie) qui était mécontent du résultat imprimé de ses livres depuis l'abandon de l'usage du plomb, et l'apparition des premiers procédés de photocomposition. Cela se passait dans la fin des années 70.

Bien que TEX ait été étendu dans bien des voies, par LATEX, et par d'autres ajouts, son usage premier, son domaine de prédilection, ça reste les mathématiques.

Il reste l'outil de choix des grandes sociétés mathématiques dans le monde, parce qu'il est l'un des seuls a savoir mettre en forme proprement des équations, et qu'il sert de référence pour les symboles mathématiques disponibles.

Je vais te présenter dans le chapitre qui vient les rudiments de la saisie d'équations avec TEX, puis une liste la plus complète possible des symboles mathématiques disponibles (il en manque forcément), puis différentes constructions mathématiques.

Ce que je n'exposerai pas, parce que ce n'est pas un manuel de mathématiques, c'est ce à quoi servent ces symboles, et dans quel cas on les utilise.

Chapitre deuxième
Mathématiques

1 Principe, environnements, généralités

1.1 Principe

Comme tu le sais déjà, LaTeX est étudié particulièrement pour la mise en page de textes scientifiques. Il est donc particulièrement étudié pour les équations.

Le principe de base de la saisie de mathématiques sous LaTeX est simple : spécifie-lui que tu passes en mode maths, indique-lui ton équation, indique que tu ressors du mode maths.

La notion de *mode* maths est importante parce que les conventions typographiques (entre autres) sont différentes de la normale.

1.2 Les environnements

Pour passer en mode maths, il y a six méthodes :

- L'environnement `displaymath`
- L'environnement `math`
- `\[...\]`, équivalent à `displaymath`
- `\(...\)`, équivalent à `math`
- `$$...$$`, équivalent[1] à `displaymath`
- `$...$`, équivalent à `math`

La grande différence entre `displaymath` et `math` est que le premier est prévu pour faire apparaître une équation seule centrée sur une ligne, alors que le second est prévu pour mettre une petite équation dans le texte.

On préferera utiliser, par soucis de lisibilité du fichier `.tex` :

- les deux environnements pour tous les passages longs (5 à 6 lignes, voire plus, dans le fichier source)
- `\[...\]` plutôt que `$$...$$` parce qu'on voit clairement si on ouvre ou si on ferme le mode mathématique.
- `$...$` pour citer un nom de variable ou un morceau très court (`$x$` pour x ou `$f(x)$` pour $f(x)$).
- `\(...\)` pour tout passage un peu long (une demi-ligne, par exemple) dans un paragraphe.

De plus, si tu souhaites que les équations « display »[2] soient non pas centrées, mais au fer à gauche[3], alors indique l'option `fleqn` à `\documentclass`. Tu pourras alors faire

```
\mathindent=1cm
```

pour que ces équations se retrouvent à 1cm de la marge gauche.

[1] En fait pas totalement équivalent, mais tu ne verras que très difficilement la différence entre les deux.

[2] Je préfère le mot français de « hors-ligne » même si je ne l'emploie pas systématiquement.

[3] « Au fer à gauche » en typographie ça veut dire « tout à gauche », plus précisément « poussé jusqu'en butée vers le fer à gauche ». Cela vient de l'héroïque époque du plomb.

1.3 Généralités

Les trucs de base à savoir pour taper la majorité des équations sont peu nombreux et assez simples à retenir. Le premier est que LaTeX tape spontanément les maths en italique, ainsi en disant « on a a qui vaut 12 », le deuxième « a » est en italique, on sait que c'est une variable. C'est plus facile à lire. C'est pour ça qu'on peut passer en mode maths facilement, pour le a j'avais tapé `$a$`. Facile, non?

Le second truc à savoir est le problème des exposants et des indices. Pour les indices, le caractère magique est `_`, et pour les exposants, c'est `^`. Exemple : `$x_0^2$` donnera x_0^2. Pour mettre plus d'une lettre en indice ou en exposant, il faut faire `$x^{12}$` : x^{12}.

Chapitre deuxième
Mathématiques

2 Symboles mathématiques

On peut, grossièrement, les regrouper en sept grandes catégories :

1) Les symboles que LaTeX 2$_\varepsilon$ reconnait spontanément.
2) Les symboles que reconnaissait LaTeX 2.09 et que LaTeX 2$_\varepsilon$ ne reconnaît plus tout seul[4].
3) Les symboles accessibles par le package `amsmath`, ou plus exactement son petit frère `amssymb`.
4) Les symboles ajoutés par `stmaryrd`.
5) Les raccourcis claviers et extensions offerts par `qsymbols`.
6) Les six symboles offerts par `ulsy`.
7) Les symboles fournis par `wasysym` qui fourni aussi des symboles pour le mode texte.

2.1 Package `latexsym`

Ce package est écrit par Frank Mittelbach, [108] ; il est maintenu par l'équipe du projet LaTeX3 ; il est livré avec LaTeX.

Le tableau 2.1 te présente les symboles définis dans le package `latexsym`. Ton observation attentive et minutieuse t'amènera très vite à constater que tous ces symboles sont aussi définis dans le package `amssymb` qui est livré avec `amsmath`, mais c'est pas utile de déplacer un mammouth (`wasysym` est un gros package) pour écraser une mouche, ou alors si mais une grosse mouche.

Commande	Sym.	Commande	Sym.	Commande	Sym.
`\mho`	℧	`\Join`	⋈	`\Box`	□
`\Diamond`	◇	`\lhd`	◁	`\unlhd`	⊴
`\rhd`	▷	`\unrhd`	⊵	`\sqsubset`	⊏
`\sqsupset`	⊐	`\leadsto`	⇝		

Tab. 2.1: Symboles définis par `latexsym`

[4] Il s'agit de symboles peu nombreux qui monopolisent une fonte à eux seuls, ce qui gaspille de la place en mémoire si on n'en a pas besoin.

2.2 Symboles LaTeX 2ε

LaTeX 2ε prévoit en standard à peu près les mêmes symboles que LaTeX 2.09 le faisait. Tous les symboles qui sont dans les tableaux 2.2 à 2.10 page 56 ne nécessitent pas de package d'extension.

Commande	Sym.	Commande	Sym.	Commande	Sym.
\alpha	α	\beta	β	\gamma	γ
\delta	δ	\epsilon	ϵ	\varepsilon	ε
\zeta	ζ	\eta	η	\theta	θ
\vartheta	ϑ	\iota	ι	\kappa	κ
\lambda	λ	\mu	μ	\nu	ν
\xi	ξ	\pi	π	\varpi	ϖ
\rho	ρ	\sigma	σ	\varsigma	ς
\tau	τ	\upsilon	υ	\phi	ϕ
\varphi	φ	\chi	χ	\psi	ψ
\omega	ω	\Gamma	Γ	\Delta	Δ
\Theta	Θ	\Lambda	Λ	\Xi	Ξ
\Pi	Π	\Sigma	Σ	\Upsilon	Υ
\Phi	Φ	\Psi	Ψ	\Omega	Ω

TAB. 2.2: Lettres grecques

Commande	Sym.	Commande	Sym.
\pm	$\pm$	\mp	$\mp$
\times	$\times$	\div	$\div$
\ast	$\ast$	\star	$\star$
\circ	$\circ$	\bullet	$\bullet$
\cdot	$\cdot$	\cap	$\cap$
\cup	$\cup$	\uplus	$\uplus$
\oplus	$\oplus$	\ominus	$\ominus$
\otimes	$\otimes$	\oslash	$\oslash$
\bigcirc	$\bigcirc$	\dagger	$\dagger$
\ddagger	$\ddagger$	\amalg	$\amalg$
\sqcap	$\sqcap$	\sqcup	$\sqcup$
\vee	$\vee$	\wedge	$\wedge$
\setminus	$\setminus$	\wr	$\wr$
\diamond	$\diamond$	\odot	$\odot$
\bigtriangleup	$\bigtriangleup$	\bigtriangledown	$\bigtriangledown$
\triangleleft	$\triangleleft$	\triangleright	$\triangleright$

TAB. 2.3: Opérateurs binaires

Commande	Sym.
\leq	$\leq$
\prec	$\prec$
\preceq	$\preceq$
\subset	$\subset$
\subseteq	$\subseteq$
\sqsubset	$\sqsubset$
\sqsubseteq	$\sqsubseteq$
\ll	$\ll$
\in	$\in$
\smile	$\smile$
\neq	$\neq$
\sim	$\sim$
\vdash	$\vdash$

Commande	Sym.
\geq	$\geq$
\succ	$\succ$
\succeq	$\succeq$
\supset	$\supset$
\supseteq	$\supseteq$
\sqsupset	$\sqsupset$
\sqsupseteq	$\sqsupseteq$
\gg	$\gg$
\ni	$\ni$
\frown	$\frown$
\doteq	$\doteq$
\simeq	$\simeq$
\dashv	$\dashv$

Commande	Sym.
\le	$\le$
\equiv	$\equiv$
\models	$\models$
\perp	$\perp$
\mid	$\mid$
\parallel	$\parallel$
\Join	$\Join$
\bowtie	$\bowtie$
\approx	$\approx$
\asymp	$\asymp$
\cong	$\cong$

TAB. 2.4: Symboles de relation

Commande	Sym.
\leftarrow	$\leftarrow$
\longleftarrow	$\longleftarrow$
\leftrightarrow	$\leftrightarrow$
\Leftarrow	$\Leftarrow$
\Longleftarrow	$\Longleftarrow$
\Leftrightarrow	$\Leftrightarrow$
\uparrow	$\uparrow$
\Uparrow	$\Uparrow$
\updownarrow	$\updownarrow$
\mapsto	$\mapsto$
\nearrow	$\nearrow$
\nwarrow	$\nwarrow$
\leftharpoonup	$\leftharpoonup$
\leftharpoondown	$\leftharpoondown$
\hookleftarrow	$\hookleftarrow$

Commande	Sym.
\rightarrow	$\rightarrow$
\longrightarrow	$\longrightarrow$
\longleftrightarrow	$\longleftrightarrow$
\Rightarrow	$\Rightarrow$
\Longrightarrow	$\Longrightarrow$
\Longleftrightarrow	$\Longleftrightarrow$
\downarrow	$\downarrow$
\Downarrow	$\Downarrow$
\Updownarrow	$\Updownarrow$
\longmapsto	$\longmapsto$
\searrow	$\searrow$
\swarrow	$\swarrow$
\rightharpoonup	$\rightharpoonup$
\rightharpoondown	$\rightharpoondown$
\hookrightarrow	$\hookrightarrow$

TAB. 2.5: Flèches

Commande	Sym.	Commande	Sym.	Commande	Sym.
\ldots	$\ldots$	\cdots	$\cdots$	\vdots	$\vdots$
\forall	$\forall$	\exists	$\exists$	\ddots	$\ddots$
\imath	$\imath$	\jmath	$\jmath$	\angle	$\angle$
\nabla	∇	\triangle	$\triangle$	\Diamond	$\Diamond$
\Re	$\Re$	\Im	$\Im$	\aleph	$\aleph$
\clubsuit	$\clubsuit$	\diamondsuit	$\diamondsuit$	\natural	$\natural$
\heartsuit	$\heartsuit$	\spadesuit	$\spadesuit$	\flat	$\flat$
\top	$\top$	\bot	$\bot$	\partial	∂
\sharp	$\sharp$	\neg	$\neg$	\wp	$\wp$
\Box	$\Box$	\ell	ℓ	\surd	$\surd$
\hbar	$\hbar$	\emptyset	$\emptyset$		
\prime	$\prime$	\infty	∞		

TAB. 2.6: Divers symboles

Commande	Sym.	Commande	Sym.	Commande	Sym.
\sum	$\sum$	\prod	$\prod$	\coprod	$\coprod$
\bigcap	$\bigcap$	\bigcup	$\bigcup$	\biguplus	$\biguplus$
\oint	$\oint$	\int	$\int$	\bigsqcup	$\bigsqcup$
\bigodot	$\bigodot$	\bigotimes	$\bigotimes$	\bigoplus	$\bigoplus$
\bigvee	$\bigvee$	\bigwedge	$\bigwedge$		

TAB. 2.7: Symboles à taille variable

Commande	Sym.	Commande	Sym.	Commande	Sym.
\sin	sin	\cos	cos	\tan	tan
\arcsin	arcsin	\arccos	arccos	\arctan	arctan
\sinh	sinh	\cosh	cosh	\tanh	tanh
\arg	arg	\cot	cot	\coth	coth
\ker	ker	\det	det	\dim	dim
\exp	exp	\log	log	\ln	ln
\lim	lim	\liminf	lim inf	\limsup	lim sup
\max	max	\min	min	\Pr	Pr
\inf	inf	\sup	sup	\hom	hom
\csc	csc	\deg	deg	\gcd	gcd
\lg	lg	\sec	sec		

TAB. 2.8: Noms de « fonctions » (log-like)

Commande	Sym.
(	(
[	[
\{	{
\lfloor	⌊
\langle	⟨
\lceil	⌈
/	/
|	‖

Commande	Sym.
)	)
]	]
\}	}
\rfloor	⌋
\rangle	⟩
\rceil	⌉
\backslash	\
\\|	‖

Commande	Sym.
\uparrow	↑
\Uparrow	⇑
\downarrow	↓
\Downarrow	⇓
\updownarrow	↕
\Updownarrow	⇕

TAB. 2.9: Délimiteurs

Commande	Résultat
\widetilde{abc}	$\widetilde{abc}$
\widehat{abc}	$\widehat{abc}$
\overleftarrow{abc}	$\overleftarrow{abc}$
\overrightarrow{abc}	$\overrightarrow{abc}$
\overline{abc}	$\overline{abc}$
\underline{abc}	$\underline{abc}$
\overbrace{abc}	$\overbrace{abc}$
\underbrace{abc}	$\underbrace{abc}$
\sqrt{abc}	$\sqrt{abc}$
\sqrt[n]{abc}	$\sqrt[n]{abc}$
f'	f'
\frac{abc}{xyz}	$\frac{abc}{xyz}$

TAB. 2.10: Constructions mathématiques

2.3 Le package amssymb

Les symboles des tableaux 2.11 page ci-contre à 2.18 page 60 nécessitent d'utiliser le package amssymb qui est livré avec amsmath.

Commande	Sym.	Commande	Sym.
\leftleftarrows	⇇	\rightrightarrows	⇉
\rightleftarrows	⇄	\leftrightarrows	⇆
\upuparrows	⇈	\downdownarrows	⇊
\twoheadleftarrow	↞	\twoheadrightarrow	↠
\Lleftarrow	⇚	\Rrightarrow	⇛
\leftarrowtail	↢	\rightarrowtail	↣
\curvearrowleft	↶	\curvearrowright	↷
\circlearrowleft	↺	\circlearrowright	↻
\looparrowleft	↫	\looparrowright	↬
\Lsh	↰	\Rsh	↱
\leftrightsquigarrow	↭	\rightsquigarrow	⇝
\upharpoonleft	↿	\upharpoonright	↾
\downharpoonleft	⇃	\downharpoonright	⇂
\leftrightharpoons	⇋	\rightleftharpoons	⇌
\multimap	⊸		

TAB. 2.11: Flèches (ajout amssymb)

Commande	Sym.	Commande	Sym.
\nleftarrow	↚	\nrightarrow	↛
\nLeftarrow	⇍	\nRightarrow	⇏
\nleftrightarrow	↮	\nLeftrightarrow	⇎

TAB. 2.12: Flèches négatives (ajout amssymb)

Commande	Sym.	Commande	Sym.	Commande	Sym.
\leqslant	⩽	\geqslant	⩾	\leqq	≦
\geqq	≧	\lll	⋘	\ggg	⋙
\eqslantless	⪕	\eqslantgtr	⪖	\lesseqgtr	⋚
\gtreqless	⋛	\lesseqqgtr	⪋	\gtreqqless	⪌
\lesssim	≲	\gtrsim	≳	\lessapprox	⪅
\gtrapprox	⪆	\precapprox	⪷	\succapprox	⪸
\sqsubset	⊏	\sqsupset	⊐	\subseteqq	⫅
\supseteqq	⫆	\Subset	⋐	\Supset	⋑
\smallsmile	⌣	\smallfrown	⌢	\therefore	∴
\because	∵	\precsim	≾	\succsim	≿
\curlyeqprec	⋞	\curlyeqsucc	⋟	\preccurlyeq	≼
\succcurlyeq	≽				

TAB. 2.13: Symboles de relation classiques (ajout amssymb)

Commande	Sym.
\lessdot	⋖
\risingdotseq	≓
\trianglelefteq	⊴
\vartriangleleft	◁
\blacktriangleleft	◀
\shortmid	∣
\backsim	∽
\vDash	⊨
\Vdash	⊩
\bumpeq	≏
\eqcirc	≖
\triangleq	≜
\thickapprox	≈
\varpropto	∝

Commande	Sym.
\gtrdot	⋗
\fallingdotseq	≒
\trianglerighteq	⊵
\vartriangleright	▷
\blacktriangleright	▶
\shortparallel	∥
\backsimeq	⋍
\Vvdash	⊪
\between	≬
\Bumpeq	≎
\circeq	≗
\thicksim	∼
\pitchfork	⋔
\backepsilon	϶

TAB. 2.14: Symboles de relation moins classiques (ajout amssymb)

Commande	Sym.
\nless	≮
\ngeq	≱
\nleqq	≦̸
\gneq	≩
\lnapprox	⪉
\nsucc	⊁
\precnsim	⋨
\succnapprox	⪊
\nmid	∤
\nshortparallel	∦

Commande	Sym.
\ngtr	≯
\nleqslant	⩽̸
\ngeqq	≧̸
\lneqq	≨
\gnapprox	⪊
\npreceq	⪯̸
\succnsim	⋩
\nsim	≁
\nshortmid	∤

Commande	Sym.
\nleq	≰
\ngeqslant	⩾̸
\lneq	⪇
\gneqq	≩
\nprec	⊀
\nsucceq	⪰̸
\precnapprox	⪹
\nparallel	∦

TAB. 2.15: Symboles de relation négatifs classiques (ajout amssymb)

Commande	Sym.	Commande	Sym.
\lvertneqq	≨	\gvertneqq	≩
\lnsim	⋦	\gnsim	⋧
\nsubseteq	⊈	\nsupseteq	⊉
\subsetneq	⊊	\supsetneq	⊋
\nsubseteqq	⫅̸	\nsupseteqq	⫆̸
\subsetneqq	⫋	\supsetneqq	⫌
\varsubsetneq	⊊	\varsupsetneq	⊋
\varsubsetneqq	⫋	\varsupsetneqq	⫌
\ntriangleleft	⋪	\ntriangleright	⋫
\ntrianglelefteq	⋬	\ntrianglerighteq	⋭
\nvdash	⊬	\nvDash	⊭
\nvDash	⊭	\nVDash	⊯

TAB. 2.16: Symboles de relation négatifs moins classiques (ajout amssymb)

Commande	Sym.	Commande	Sym.
\barwedge	⊼	\doublebarwedge	⩞
\Cap	⋒	\Cup	⋓
\boxplus	⊞	\boxminus	⊟
\boxtimes	⊠	\boxdot	⊡
\ltimes	⋉	\rtimes	⋊
\curlywedge	⋏	\curlyvee	⋎
\leftthreetimes	⋋	\rightthreetimes	⋌
\circleddash	⊝	\circledast	⊛
\circledcirc	⊚	\centerdot	⋅
\dotplus	∔	\smallsetminus	∖
\veebar	⊻	\divideontimes	⋇
\intercal	⊺		

TAB. 2.17: Opérateurs binaires (ajout amssymb)

Commande	Sym.	Commande	Sym.
\sphericalangle	$\sphericalangle$	\complement	$\complement$
\hbar	$\hbar$	\hslash	$\hslash$
\vartriangle	$\vartriangle$	\triangledown	$\triangledown$
\blacktriangle	$\blacktriangle$	\blacktriangledown	$\blacktriangledown$
\angle	$\angle$	\measuredangle	$\measuredangle$
\square	$\square$	\blacksquare	$\blacksquare$
\lozenge	$\lozenge$	\blacklozenge	$\blacklozenge$
\diagdown	$\diagdown$	\diagup	$\diagup$
\nexists	$\nexists$	\mho	$\mho$
\Game	$\Game$	\Bbbk	$\Bbbk$
\backprime	$\backprime$	\varnothing	$\varnothing$
\bigstar	$\bigstar$	\eth	$\eth$
\circledS	$\circledS$	\Finv	$\Finv$

TAB. 2.18: Divers symboles (ajout `amssymb`)

2.4 Le package `stmaryrd`

Ce package, écrit par Jeremy GIBBONS et Alan JEFFREY [52] donne accès aux symboles de leur propre fonte, celle qu'ils ont appelée « St Mary's Road symbol font[5] ».

Ce package définit une tripotée de symboles, en fait tous ceux des tableaux 2.19 page ci-contre à 2.23 page 62. Note en particulier que sur les délimiteurs (tableau 2.23 page 62), seuls `\llbracket` et `\rrbracket` sont de taille variable et peuvent être utilisés avec `\left` et `\right`.

[5]Faut pas me demander pourquoi. Peut-être qu'ils l'on dessinée dans un couvent ou devant une église.

Commande	Sym.	Commande	Sym.
\Ydown	⟒	\Yleft	⟙
\Yup	⟑	\Yright	⟘
\binampersand	⅋	\bindnasrepma	⅌
\boxbar	◫	\boxbox	⧈
\boxslash	◪	\boxcircle	⧇
\boxbslash	◩	\boxast	⧆
\boxempty	□	\boxdot	⧈
\curlyveedownarrow	⋎	\curlyveeuparrow	⋏
\curlywedgeuparrow	⋏	\curlywedgedownarrow	⋎
\fatslash	⫽	\fatbslash	⫻
\fatsemi	⨟	\interleave	⫴
\leftslice	◅	\rightslice	▻
\sslash	⫽	\nplus	⨣
\minuso	⦵	\moo	⨦
\obar	⦶	\oblong	▭
\obslash	⦸	\ogreaterthan	⧁
\owedge	⦹	\olessthan	⧀
\ovee	⦺		
\talloblong	▯	\varbigcirc	◯
\varcurlyvee	⋎	\varcurlywedge	⋏
\varobar	⦶	\varoast	⊛
\varobslash	⦸	\varocircle	⊚
\varoslash	⊘	\varoplus	⊕
\varominus	⊖	\varotimes	⊗
\varolessthan	⧀	\varogreaterthan	⧁
\varowedge	⦹	\varovee	⦺
\vartimes	⨯		

TAB. 2.19: Opérateurs (ajout stmaryrd)

Commande	Sym.	Commande	Sym.
\bigbox	⨦	\bigsqcap	⨅
\biginterleave	⫼	\bigparallel	⫲
\bigcurlyvee	⋎	\bigcurlywedge	⋏
\bigtriangledown	▽	\bigtriangleup	△
\bignplus	⨁		

TAB. 2.20: Opérateurs à taille variable (ajout stmaryrd)

Commande	Sym.
\inplus	⋵
\subsetplus	⊹
\subsetpluseq	⫅
\trianglelefteqslant	◭
\ntrianglelefteqslant	⬗

Commande	Sym.
\niplus	⋽
\supsetplus	⊺
\supsetpluseq	⫆
\trianglerighteqslant	◮
\ntrianglerighteqslant	⬘

TAB. 2.21: Symboles de relation (ajout `stmaryrd`)

Commande	Sym.
\Mapsfrom	⇐|
\Longmapsfrom	⟸|
\leftarrowtriangle	⇽
\leftrightarrowtriangle	⇿
\shortleftarrow	←
\mapsfrom	↤
\longmapsfrom	⟻
\nnwarrow	↖
\sswarrow	↙
\shortdownarrow	↓

Commande	Sym.
\Mapsto	|⇒
\Longmapsto	|⟹
\rightarrowtriangle	⇾
\leftrightarrowweq	⇼
\shortrightarrow	→
\lightning	↯
\nnearrow	↗
\ssearrow	↘
\shortuparrow	↑

TAB. 2.22: Flèches (ajouts `stmaryrd`)

Commande	Sym.
\Lbag	⟅
\lbag	⟅
\llceil	⌈
\llfloor	⌊
\llbracket	⟦
\llparenthesis	⦇

Commande	Sym.
\Rbag	⟆
\rbag	⟆
\rrceil	⌉
\rrfloor	⌋
\rrbracket	⟧
\rrparenthesis	⦈

TAB. 2.23: Délimiteurs (ajout `stmaryrd`)

2.5 Le package `qsymbols`

Ce package est écrit par Kristoffer H. ROSE [135].

Son utilité n'est pas flagrante : aucun nouveau symbole, aucune nouvelle fonctionnalité. Simplement, une syntaxe abrégée pour accéder à de nombreux symboles. Ces symboles sont ceux usuels sous LaTeX, une partie de ceux du package `amssymb`, du package `stmaryrd` ainsi que certaines possibilités offertes par Xy-pic pour les flèches.

Saisie	Résultat																
x	a	b	c	d	e	f	g	h	i	j	k	l	m	n	p	q	r	s
‘x	α	β	χ	δ	ϵ	ϕ	γ	η	ι	ψ	κ	λ	μ	ν	π	θ	ρ	σ
x	t	w	x	y	z			D	F	G	J	L	P	Q	S	W	X	Y
‘x	τ	ω	ξ	υ	ζ			Δ	Φ	Γ	Ψ	Λ	Π	Θ	Σ	Ω	Ξ	Υ
x	+	*	:	;	/	U	C	_	o	T	.	=	~	E	A	!	^	V
‘x	$\pm$	$\times$	$\in$	$\notin$	$\setminus$	$\cup$	$\complement$	$\perp$	$\circ$	$\top$	$\cdot$	$\equiv$	$\simeq$	$\exists$	$\forall$	$\neg$	$\wedge$	$\vee$
x		+	-	‘*	/	|	‘/	‘.	*	‘o	‘~	‘V	<	>	?	!	:-	a
‘(x)	◯	⊕	⊖	⊗	⊘	⊙	⊚	⊛	⊜	⊝	⊚	⊛	⊜	⊝	∘	•	☺	ⓐ
‘[x]	□	⊞	⊟	⊠	◪	◫	◪	⊡	⊛	⊡	⊠	⊞	⊟	⊠	▫	▪	☹	ⓐ
‘<x>															◇		☺	
‘{x}																	☹	

TAB. 2.24: Symboles simples avec `qsymbols`

Symbole		Ouvert					Fermé						
∈	∋	‘‘∈	‘‘/∈	‘‘∈=	‘‘/∈=	‘‘‘∈	‘‘‘‘∈	‘‘∋	‘‘/∋	‘‘∋=	‘‘/∋=	‘‘‘∋	‘‘‘‘∋
<	>	$<$	$\not<$	$\leq$	$\not\leq$	$\wedge$	$\bigwedge$	$>$	$\not>$	$\geq$	$\not\geq$	$\vee$	$\bigvee$
(	)	$\subset$	$\not\subset$	$\subseteq$	$\not\subseteq$	$\cap$	$\bigcap$	$\supset$	$\not\supset$	$\supseteq$	$\not\supseteq$	$\cup$	$\bigcup$
[	]	$\sqsubset$	$\not\sqsubset$	$\sqsubseteq$	$\not\sqsubseteq$	$\sqcap$	$\bigsqcap$	$\sqsupset$	$\not\sqsupset$	$\sqsupseteq$	$\not\sqsupseteq$	$\sqcup$	$\bigsqcup$
\{	\}	$\prec$	$\not\prec$	$\preceq$	$\not\preceq$	$\curlywedge$	$\bigcurlywedge$	$\succ$	$\not\succ$	$\succeq$	$\not\succeq$	$\curlyvee$	$\bigcurlyvee$
\<	\>	$\lhd$	$\not\lhd$	$\trianglelefteq$	$\not\trianglelefteq$	$\triangle$	$\bigtriangleup$	$\rhd$	$\not\rhd$	$\trianglerighteq$	$\not\trianglerighteq$	$\bigtriangledown$	$\bigtriangledown$
~	\~	$\sim$	$\not\sim$	$\simeq$	$\not\simeq$	$\wr$		$\backsim$	$\not\backsim$	$\eqsim$	$\neq$	$\int$	$\int$
(-	-)	$\in$	$\notin$			$\pitchfork$		$\ni$					
(+	+)	$\Subset$	$\not\Subset$	$\Subset$	$\not\Subset$	$\Cap$	$\bigCap$	$\Supset$	$\not\Supset$	$\Supset$	$\not\Supset$	$\uplus$	$\biguplus$

TAB. 2.25: Symboles de relation avec `qsymbols`

2.5.1 Les symboles simples

On les obtient à l'aide du caractère ‘ comme l'indique le tableau 2.24.

Note qu'on peut encadrer ou encercler même des symboles qui ne sont pas prévus (c'est le cas du a). Le gras d'un symbole s'obtient en le faisant précéder de `‘@`[6], par exemple `‘@‘a` donne : $\boldsymbol{\alpha}$.

Tu peux ajouter un nouveau symbole à la liste :

```
\newqsymbol{‘code}{signification}
```

Où `code` peut être une lettre ou une lettre entre (), [], {}.

2.5.2 Symboles de relation

On les obtiendra souvent en utilisant deux ‘‘, comme dans le tableau 2.25.

Relevons également les symboles à taille variable du tableau 2.26 page suivante.

[6] Dans les version précédentes du package `"@` était utilisé, mais un nouveau symbole lui a été préféré pour simplifier la gestion du guillemet double puisque celui-ci est également utilisé pour les flèches.

x	U	V	S	P
``x	$\bigcup$	$\bigvee$	$\sum$	$\prod$

TAB. 2.26: Symboles à taille variable avec `qsymbols`

Gauche		Double		Droite	
`"<-"`	$\leftarrow$	`"<->"`	$\leftrightarrow$	`"->"`	$\rightarrow$
`"<="`	$\Leftarrow$	`"<=>"`	$\Leftrightarrow$	`"=>"`	$\Rightarrow$
`"<3"`	$\Lleftarrow$			`"3>"`	$\Rrightarrow$
`"</-"`	$\nleftarrow$	`"</->"`	$\nleftrightarrow$	`"-/>"`	$\nrightarrow$
`"<-/"`	$\not\leftarrow$	`"<-/>"`	$\not\leftrightarrow$	`"/->"`	$\not\rightarrow$
`"</="`	$\not\Leftarrow$	`"</=>"`	$\not\Leftrightarrow$	`"=/>"`	$\not\Rightarrow$
`"<=/"`	$\not\Lleftarrow$	`"<=!!!/>"`	$\not\Longleftrightarrow$	`"/=>"`	$\not\Rrightarrow$
`"<-'"`	$\hookleftarrow$			`"'->"`	$\hookrightarrow$
`"<<-"`	$\twoheadleftarrow$	`"<<-!!!>>"`	$\twoheadleftarrow\!\!\!\!\!\!\rightarrow$	`"->>"`	$\twoheadrightarrow$
`"<<="`	$\Lleftarrow$	`"<<=!!!>>"`	$\Lleftarrow\!\!\!\!\!\!\Rightarrow$	`"=>>"`	$\Rrightarrow$
`"<-<"`	$\leftarrowtail$			`">->"`	$\rightarrowtail$
`"<|-"`	$\leftarrow\!\!\!\!\!\!\rightarrow$	`"<|-|>"`	$\leftarrow\!\!\!\!\!\!\rightarrow$	`"-|>"`	$\rightarrow\!\!\!\!\!\!\rightarrow$
`"o-"`	$\multimap$	`"o-o"`	$\multimap\!\!\!\!\!\!\multimap$	`"-o"`	$\multimap$
`"^<-"`	$\leftarrow\!\!\!\!\!\!\leftarrow$	`"^<->"`	$\leftarrow\!\!\!\!\!\!\leftarrow\!\!\!\!\!\!\rightarrow$	`"^->"`	$\rightarrow\!\!\!\!\!\!\rightarrow$
`"_<-"`	$\leftarrow\!\!\!\!\!\!\rightarrow$	`"_<->"`	$\leftarrow\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!\rightarrow$	`"_->"`	$\rightarrow\!\!\!\!\!\!\rightarrow$
`"<~"`	$\leftsquigarrow$	`"<~>"`	$\leftrightsquigarrow$	`"~>"`	$\rightsquigarrow$

TAB. 2.27: Flèches standard avec `qsymbols`

2.5.3 Flèches standard

Les flèches les plus courantes avec LaTeX sont accessibles avec une syntaxe simplifiée[7] comme indiqué par le tableau 2.27.

Parmi ces flèches, certaines sont succeptible de s'étendre indéfiniement pour s'adapter à la taille d'un indice ou d'un exposant, comme par exemple dans `"<-{_{ax+by+cz+d}}"` qui produira $\xleftarrow[ax+by+cz+d]{}$. Les flèches reconnaissant cette syntaxe et capable de s'y adapter sont `"<-"`, `"->"`, `"<="`, `"=>"`, `"<3"`, `"3>"`, `"<-|"`, `"|->"`, `"<=|"`, `"|=>"`, `"<|-"`, `"<|-|>"`, `"-|>"`, `"«-"`, `"-»"`, `"«="`, et `"=»"`.

D'autre part, certaines peuvent être allongées artificiellement en ajoutant un ou plusieurs `!` dans le corps.

2.5.4 Flèches étendues

Les tableaux 2.28 page suivante à 2.29 page ci-contre t'indiqueront comment obtenir des flèches de plus en plus complexes avec le même type de syntaxe[8].

[7] Ça c'est l'avis de l'auteur du package, et pas tout à fait le mien, parce que bon, faut pas pousser, c'est pas toujours intuitif.

[8] Là, j'aime de moins en moins, en particulier les flèches du tableau 2.29 page suivante. Je les trouve ignobles à taper ces trucs, mais c'est mon opinion et elle n'engage que moi.

x	`‘-/>`	`-//>`	`=>{_?}`	`‘->\{^+\}`
"x"	$\not\curvearrowright$	$\not\Rightarrow$	$\overset{?}{\Rightarrow}$	$\overset{+}{\hookrightarrow}$

TAB. 2.28: Flèches non-standard avec `qsymbols`

x	`-{_1}>`	`={^{\sin}}!!!>>`	`={|{‘b}}!!!|>`	`(1)>`	`[2]>`
"x"	$\overset{}{\underset{1}{\rightarrow}}$	$\overset{\sin}{\Longrightarrow}$	$\overset{\beta}{=\!\!\Rightarrow}$	$\xrightarrow{1}$	$\xrightarrow{2}$
x	`-(1)!!>`	`-[1]!!>`	`-(1)!!>>`	`-[‘a]!!!>>`	`<<|=[‘c]!!!>>`
"x"	$\xrightarrow{1}$	$\xrightarrow{1}$	$\xrightarrow{1}\!\!\!\rightarrow$	$\xrightarrow{a}\!\!\!\rightarrow$	$\overset{c}{\Longleftarrow}\!\!\!\Longrightarrow$

TABLE 2.29: Flèches complexes avec `qsymbols`

2.5.5 Digression

J'ai été amené, durant plusieurs années, a faire évoluer régulièrement ce livre, pour le corriger, pour le complèter, pour suivre les évolutions de ce que j'y documente.

Le package `qsymbols` a toujours été pour moi une source de problèmes, et je ne parviens toujours pas à savoir s'il me fallait le laisser ou non.

Il a toujours fonctionné bizarement, j'y ai régulièrement relevé de petits bugs, et il est incompatible avec pas mal d'autres. Mais, je trouve que pour quelqu'un qui a beaucoup de maths à taper, il représente un vrai atout. Et puis, le JMPL n'est pas vraiment un document normal, il utilises trop de packages trop bizares pour pouvoir être considéré comme normal. Mon jugement en est peut-être faussé.

Si tu souhaites t'en servir, souviens toi que `qsymbols` est une source d'ennuis possibles.

2.6 Le package `ulsy`

Ce petit package, que nous devons à Ulrich GOLDSCHMITT [54] définit 6 nouveaux symboles. Ce ne sont pas des symboles mathématiques en ce sens qu'ils peuvent être appelés n'importe où dans le texte.

Ces six symboles sont définis dans le tableau 2.30.

Commande	Sym.
`\odplus`	⊕
`\blitzc`	↯

Commande	Sym.
`\blitza`	↯
`\blitzd`	↯

Commande	Sym.
`\blitzb`	↯
`\blitze`	↯

Attention, ces symboles sont des symboles accessibles en mode texte uniquement. Pour les obtenir en mode math il faudra donc faire appel à `\hbox{\odplus}` par exemple.

TAB. 2.30: Symboles définis par le package `ulsy`

2.7 Le package `wasysym`

Ce package, que nous devons à Axel KIELHORN [73] définit une floppée de nouveaux symboles pour LaTeX dont certains sont des symboles mathématiques[9] et d'autres sont des symboles normaux utilisables dans le texte.

La fonte éponyme (`wasy`) est l'œuvre de Roland WALDI [155]. `wasy` veut dire, dixit l'auteur, « Waldi Symbol ».

Commande	Sym.	Commande	Sym.	Commande	Sym.
`\Join`	⋈	`\apprle`	≲	`\Box`	□
`\apprge`	≳	`\Diamond`	◇	`\wasypropto`	∝
`\leadsto`	⇝	`\invneg`	⌐	`\sqsubset`	⊏
`\ocircle`	○	`\sqsupset`	⊐	`\logof`	⊛
`\lhd`	◁	`\varint`	∫	`\unlhd`	⊴
`\iint`	∬	`\LHD`	◀	`\iiint`	∭
`\rhd`	▷	`\varoint`	∮	`\unrhd`	⊵
`\oiint`	∯	`\RHD`	▶		

TAB. 2.31: Symboles mathématiques ajoutés par `wasysym`

Commande	Sym.	Commande	Sym.	Commande	Sym.
`\male`	♂	`\RIGHTarrow`	▶	`\female`	♀
`\LEFTarrow`	◀	`\currency`	¤	`\UParrow`	▲
`\phone`	☎	`\DOWNarrow`	▼	`\recorder`	⌕
`\diameter`	⌀	`\clock`	🕒	`\invdiameter`	⍉
`\lightning`	↯	`\varangle`	⦟	`\pointer`	➪
`\wasylozenge`	⌑	`\kreuz`	✠	`\ataribox`	◘
`\smiley`	☺	`\cent`	¢	`\frownie`	☹
`\permil`	‰	`\blacksmiley`	☻	`\brokenvert`	¦
`\sun`	☼	`\wasytherefore`	∴	`\checked`	✓
`\Bowtie`	⋈	`\bell`	🔔	`\agemO`	℧

Attention : le symbole `\lightning` est un symbole mathématique en ce sens qu'il doit apparaître en mode math uniquement, ceci dit, je ne comprends pas bien le sens qu'il peut avoir dans une équation. C'est pour ça que je l'ai mis dans les symboles généraux.

TAB. 2.32: Symboles généraux ajoutés par `wasysym`

[9] Relevons au passage que ces symboles font double emploi avec ceux introduits par le package `latexsym`, ou du moins la majorité d'entre eux. Même si le dessin exact de ces symboles n'est pas forcément tout à fait le même, tu ne devrais pas voir la différence dans ce livre.

Commande	Sym.	Commande	Sym.	Commande	Sym.
\AC	∼	\HF	≈	\photon	∼∼∼
\VHF	≋	\gluon	

Ces symboles ne sont pas des symboles mathématiques. Ils peuvent donc apparaître où bon te semblera dans le texte.

TAB. 2.33: Symboles de physique et d'électricité ajoutés par `wasysym`

Commande	Sym.	Commande	Sym.	Commande	Sym.
\Square	□	\XBox	⊠	\pentagon	⬠
\CheckedBox	☑	\hexagon	⬡	\hexstar	✳
\varhexagon	⬡	\varhexstar	✳		
\octagon	⯃	\davidsstar	✡		

Ça non plus, c'est pas symboles maths, donc c'est utilisable en mode texte.

TAB. 2.34: Polygônes et étoiles (`wasysym`)

Commande	Sym.	Commande	Sym.	Commande	Sym.
\eighthnote	♪	\quarternote	♩	\halfnote	𝅗𝅥
\fullnote	𝅝	\twonotes	♫		

Ces symboles aussi sont utilisables en mode texte.

TAB. 2.35: Notes de musique (`wasysym`)

Commande	Sym.	Commande	Sym.	Commande	Sym.
\Circle	○	\CIRCLE	●	\Leftcircle	◖
\LEFTCIRCLE	◖	\Rightcircle	◗	\RIGHTCIRCLE	◗
\leftturn	↺	\LEFTcircle	◐	\rightturn	↻
\RIGHTcircle	◑				

Encore une fois, au risque de me répéter, ces symboles sont utilisables en mode texte.

TAB. 2.36: Cercles divers (`wasysym`)

Commande	Sym.	Commande	Sym.	Commande	Sym.
\thorn	þ	\dh	ð	\Thorn	Þ
\Dh	Đ	\openo	ɔ	\inve	ə

Je voudrais pas avoir l'air de me répéter, mais ça non plus, c'est pas exclusivement réservé aux mathématiques. C'est librement utilisable dans le texte.

TAB. 2.37: Symboles phonétiques (wasysym)

Commande	Sym.	Commande	Sym.	Commande	Sym.
\vernal	♈	\astrosun	☉	\ascnode	☊
\mercury	☿	\descnode	☋	\venus	♀
\fullmoon	○	\earth	♁	\newmoon	●
\mars	♂	\leftmoon	☾	\jupiter	♃
\rightmoon	☽	\saturn	♄	\uranus	♅
\neptune	♆	\pluto	♇		

T'as deviné? C'est aussi utilisable en mode texte.

TAB. 2.38: Symboles d'astronomie (wasysym)

Commande	Sym.	Commande	Sym.	Commande	Sym.
\aries	♈	\scorpio	♏	\taurus	♉
\sagittarius	♐	\gemini	♊	\capricornus	♑
\cancer	♋	\aquarius	♒	\leo	♌
\pisces	♓	\virgo	♍	\conjunction	☌
\libra	♎	\opposition	☍		

Pour changer, ça aussi c'est utilisable en mode texte.

TAB. 2.39: Symboles d'astrologie (wasysym)

Commande	Sym.
\APLstar	⋆
\APLlog	⊛
\APLbox	□
\APLup	△
\APLdown	▽
\APLinput	⍞
\APLinv	⌹
\APLminus	¯

Commande	Sym.
\APLdownarrowbox	⍗
\APLleftarrowbox	⍇
\APLrightarrowbox	⍈
\notbackslash	⍀
\notslash	⌿
\APLcomment	⍝
\APLuparrowbox	⍐

Tout ça c'est du mode texte, pour changer.

TAB. 2.40: Symboles APL (wasysym)

Commande	Sym.
\APLnot{a}	$\not{a}$
\APLcirc{a}	$\circledast a$
\APLvert{a}	$\not\mid a$

Ça aussi, c'est du texte.

TAB. 2.41: Commandes APL (wasysym)

Chapitre deuxième
Mathématiques

3 Constructions mathématiques

3.1 Sommes

Et comment qu'on fait pour obtenir

$$\sum_{i=0}^{n} u_i$$

me demanderas-tu, lecteur curieux et impatient.

C'est fastoche :

```
\[\sum_{i=0}^{n}u_i\]
```

De nombreux symboles peuvent être utilisés comme cela, plus exactement, tous les symboles à taille variable recensés dans le tableau 2.7 page 55.

Relevons au passage la différence entre `math` et `displaymath`. Dans un cas on produit $\sum_{i=0}^{n} u_i$ dans la ligne avec `\(\sum_{i=0}^\nu_i\)`. Alors que dans l'autre on produit l'exemple précédent.

Sache toutefois que si tu souhaites voir un truc du genre $\sum_{i=0}^{n} u_i$ en plein dans le texte comme ici, alors il faudra faire appel à la commande `\displaystyle` :

```
\(\displaystyle\sum_{i=0}^nu_i\)
```

Les autres commandes du même accabit sont :

- `\textstyle` équation dans le texte $\sum_{i=0}^{n} u_i$
- `\scripstyle` équation en indice $\sum_{i=0}^{n} u_i$
- `\scriptscriptstyle` équation en indice d'indice $\sum_{i=0}^{n} u_i$

J'aimerai attirer ton attention sur le cas de l'intégrale. LaTeX, la met en forme, par exemple, comme ça :

$$\int_0^1 x\,dx = \left[\frac{x^2}{2}\right]_0^1 = \frac{1}{2}$$

Ce résultat est assez acceptable. Je trouve qu'il l'est moins quand on complique un peut l'équation, pour rire :

$$\int_{i+j-1}^{i+j+1} x\,dx$$

Je préfère souvent ajouter un peu d'espace négatif pour rapprocher l'ensemble, en un résultat qui me plait plus, même si c'est une question de goût et que mon goût en la matière est discutable :

$$\int_{i+j-1}^{i+j+1} x\,dx \qquad \int_0^1 x\,dx$$

Ce résultat est obtenu à partir du source suivant :

```
\[ \int_{i+j-1}^{i+j+1} \!\!\!\!\!\!\!\!\!\!\!\!\!x\,dx
\qquad \int_0^1 \!\! x\,dx\]
```

Pour les commandes sur la gestion de l'espacement, reporte toi à la section 3.7 page 75, et plus particulièrement au tableau 3.2 page 76.

3.2 Opérateurs et fonctions

Ce qu'il est convenu d'appeler les « fonctions » standard (sin, cos,...) sont plus jolies quand elles sont tapées en texte normal alors que le reste des équations (principalement les variables) est en italique. Pour cela, ces fonctions correspondent à des commandes, parmi lesquelles tu retrouveras aussi la limite, puisque ces commandes sont ce que LaTeX appelle des `mathop` ou opérateurs mathématiques. Tous ces opérateurs mathématiques sont regroupés dans le tableau 2.8 page 55. Certains fonctionnent comme la limite, à toi de savoir lesquels...

Cas « particulier » : la limite

$$\lim_{n\to+\infty} u_n = \ell$$

s'obtient en tapant

```
\[ \lim_{n\rightarrow +\infty}u_n=\ell \]
```

LaTeX choisira tout seul, en fonction de l'usage mathématiques classique de l'opérateur, s'il faut le mettre en forme comme une limite ou pas. Voici quelques exemples pour fixer les idées :

$$\lim_x \qquad \cos^2 \qquad \log_{10}$$

Cependant, on peut forcer son choix, ainsi, le même exemple en le forçant à faire les choix contraires :

$$\lim\nolimits_x \qquad \cos\limits^2 \qquad \log\limits_{10}$$

Bien entendu, mon exemple donne un résultat particulièrement idiot, mais tu pourrais croiser un usage moins bête. Voila comment ça s'obtient :

```
\[ \lim\nolimits_x \qquad
   \cos\displaylimits^2 \qquad
   \log\displaylimits_{10} \]
```

3.3 Fractions, racines et accolades

Allons-y à grands coups d'exemples :

$$\sum_{n=0}^{+\infty} \frac{x^n}{n!} = e^x = \sqrt{e^{2x}}$$

$$\sum_{i=0}^{n} u_i = \underbrace{u_0+u_1+\cdots+u_n}_{n+1 \text{ termes}}$$

s'obtiennent à partir de

```
\[ \sum_{n=0}^{+\infty}\frac{x^n}{n!}
   =e^x=\sqrt{e^{2x}} \]
\[ \sum_{i=0}^n u_i =
   \underbrace{u_0+u_1+\cdots+u_n}_{n+1
   \hbox{\scriptsize~termes}} \]
```

Note au passage le `\cdots` qui permet d'aligner les points convenablement au centre de la ligne (pour s'aligner avec le $+$), alors que pour $(x_0, x_1, \ldots, x_n)$, il faut aligner en bas de la ligne, sur la virgule, on tapera alors :

```
\((x_0,x_1,\ldots,x_n)\)
```

Terminons avec

```
\[\overbrace{u_0+u_1+\cdots+u_n}^{x}\]
```

qui donne

$$\overbrace{u_0 + u_1 + \cdots + u_n}^{x}$$

On pourra chercher à combiner les deux variétés d'accolades sans trop de difficulté. Ainsi, pour expliciter la décomposition d'une suite, on peut nommer les termes :

$$\sum_{i=0}^{n} u_i = \overbrace{\underbrace{1}_{u_0} + \underbrace{\frac{x}{1}}_{u_1} + \underbrace{\frac{x^2}{2!}}_{u_2} + \underbrace{\frac{x^3}{3!}}_{u_3} + \cdots + \underbrace{\frac{x^n}{n!}}_{u_n}}^{n+1 \text{ termes}}$$

Cela s'obtient à partir du source suivant :

```
\[
\sum_{i=0}^n u_i = \overbrace{
                  \underbrace{1\phantom{\frac{1}{1}}}_{u_0}
                + \underbrace{\frac{x}{1}}_{u_1}
                + \underbrace{\frac{x^2}{2!}}_{u_2}
                + \underbrace{\frac{x^3}{3!}}_{u_3}
                + \cdots
                + \underbrace{\frac{x^n}{n!}}_{u_n}
                }^{n+1 \text{ termes}}
\]
```

Quelques remarques sur cette équation :

- J'ai utilisé la construction avec `\phantom` pour forcer la hauteur du 1 à la même hauteur qu'une fraction, ainsi les accolades du bas sont alignées.
- La commande `\text` est introduite par les packages de l'$\mathcal{AMS}$ (package `amsmath`), et ne fait pas partie de LaTeX. Elle fait à peu de choses près la même chose que `\hbox{\scriptsize ...}`, à ceci près qu'elle détermine toute seule quelle taille il faut prendre, et quelle fonte de texte il faut prendre (la même que celle utilisée par le texte en dehors de l'équation).
  J'en reparle un peu plus loin, en la comparant à un alphabet mathématique à la section 4.5 page 93.
- L'espace entre $n+1$ et « termes » est celui qui se trouve juste avant « termes » dans l'appel à `\text`. Celui qui se trouve après le 1 de $n+1$ n'est pas pris en compte. En effet, en mode maths, les espaces ne sont pas pris en comptes. Le paramètre de la commande `\text` est traité en mode texte, et non en mode maths, les espaces y sont significatifs.

3.4 Délimiteurs

3.4.1 Pour les feignants

En mathématiques, on aime bien avoir des grandes parenthèses, pour cela, on procèdera comme suit :

```
\[\left(\sum_{i=0}^n u_i\right)\]
```

pour obtenir

$$\left(\sum_{i=0}^{n} u_i\right)$$

Tous les délimiteurs qui peuvent se trouver derrière `\left` ou `\right` sont recensés dans le tableau 2.9 page 56. Rappelons le délimiteur vide (un point) qui est bien utile pour les systèmes d'équations (une accolade à gauche et rien à droite), comme, par exemple :

```
\[\left\{ systeme \right.\]
```

Ainsi le système :

$$f(x) = \left\{ \begin{array}{ccc} x & \text{si} & x \geq 0 \\ -x & \text{si} & x < 0 \end{array} \right.$$

s'obtient à partir du source suivant :

```
\[ f(x) = \left\{
   \begin{array}{ccc}
   x & \text{si} & x \geq 0 \\
   -x & \text{si} & x < 0
   \end{array}
   \right.
\]
```

3.4.2 Pour les perfectionnistes

L'usage des parenthèses en mathématiques peut s'avérer un peu plus subtil que simplement choisir entre petit, grand, et très grand.

Par exemple dans cette équation, on s'y retrouve difficilement :

$$((a+b+c(t))(d+e(x+2(y-1))+f(y+2(x+1)))+i(y+2(x+1))-j(k))^2$$

Le source est simple, mais pas beaucoup plus lisible :

```
\[ ((a+b+c(t))(d+e(x+2(y-1))+
  f(y+2(x+1)))+i(y+2(x+1))-j(k))^2 \]
```

L'usage systématique de `\left` et `\right` n'y changera pas grand chose, en effet, tout dans cette ligne est d'une hauteur standard. Cela donnera un source plus dur à lire, et un résultat décevant :

```
\[ \left(\left(a+b+c\left(t\right)\right)
   \left(d+e\left(x+2\left(y-1\right)\right)+
  f\left(y+2\left(x+1\right)\right)\right)+
  i\left(y+2\left(x+1\right)\right)-
  j\left(k\right)\right)^2 \]
```

Avec un résultat assez nul :

$$\left((a+b+c\left(t\right)\right)\left(d+e\left(x+2\left(y-1\right)\right)+f\left(y+2\left(x+1\right)\right)\right)+i\left(y+2\left(x+1\right)\right)-j\left(k\right)\Big)^2$$

Il y a quatre variété de commandes qui vont nous tirer d'affaire, pour permettre de régler un peu plus finement la taille des parenthèses : `\big`, `\Big`, `\bigg` et `\Bigg`. Chacune existe en trois modèles, l'un à gauche (`\bigl`), le second au centre (`\bigm`) et le troisième à droite (`\bigr`).

Ça donne par exemple ça :

$$\Big(\Big(\Big(\big((x)\big)\Big)\Big)\Big)$$

Si je l'applique à notre exemple précédent, en utilisant des parenthèses normales pour les plus internes, et des parenthèses de plus en plus grandes pour les regroupements d'opérateurs, j'arrive à un résultat dont le source reste pénible à lire, mais qui est plus joli :

```
\[ \biggl(\Bigl(a+b+c(t)\Bigr)
   \Bigl(d+e\bigl(x+2(y-1)\bigr)+
  f\bigl(y+2(x+1)\bigr)\Bigr)+
  i\bigl(y+2(x+1)\bigr)-
  j(k)\biggr)^2 \]
```

Pour un résultat qui au final facilite tout de même bien la lecture :

$$\Big(\Big(a+b+c(t)\Big)\Big(d+e\big(x+2(y-1)\big)+f\big(y+2(x+1)\big)\Big)+i\big(y+2(x+1)\big)-j(k)\Big)^2$$

Bien entendu, ces commandes fonctionnent avec tous les délimiteurs.

Un exemple simple d'utilisation de la variété centrée :

$$\langle x|y\rangle = 6\left\langle \frac{x}{2} \,\middle|\, \frac{y}{3} \right\rangle$$

Que l'on obtient à partir du source suivant :

```
\[ \langle x|y\rangle =
 6 \biggl\langle \frac{x}{2}
   \biggm|
   \frac{y}{3} \biggr\rangle \]
```

3.5 Les matrices

Pour faire une matrice c'est très simple. Regarde l'exemple là et tu sauras :

```
\[A=\left(
\begin{array}{ccccc}
1 & 2 & 3 & \cdots & n \\
2 & 3 & 4 & \cdots & n+1 \\
3 & 4 & 5 & \cdots & n+2 \\
\vdots & \vdots & \vdots &
\ddots & \vdots \\
n & n+1 & n+2 & \cdots & 2n-1 \\
\end{array}
\right)^{2}\]
```

produira :

$$A = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ 2 & 3 & 4 & \cdots & n+1 \\ 3 & 4 & 5 & \cdots & n+2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n & n+1 & n+2 & \cdots & 2n-1 \end{pmatrix}^2$$

Fastoche non ? À noter : le `c` dans l'argument du `array` indique que l'on souhaite une colonne centrée. On peut préférer indiquer `l` pour une colonne gauche ou `r` pour une colonne droite.

3.6 Les équations et assimilés

Souvent, dans les ouvrages de mathématiques, on fait référence à certaines équations dans le texte, pas seulement juste avant ou juste après, où un simple choix de mots peut suffir pour y faire référence (par exemple : « l'équation précédente nous a montré... »).

Il est parfois pratique de pouvoir faire explicitement référence à une équation, par exemple par son numéro.

C'est à ça que sert l'environnement `equation`.

Il ressemble en tous points à l'environnement `displaymath`, sauf qu'il numérote.

Ainsi, là où ce source :

```
\[ a+b=x \]
```

produit :

$$a+b=x$$

l'environnement `equation` :

```
\begin{equation}
a+b=x
\end{equation}
```

produit :

$$a+b=x \tag{3.1}$$

3.7 Les accents et les espaces

Si tu voulais taper un texte en mode mathématiques, tu apprendrais très vite que les accents ne sont pas compris et que les espaces sont éjectés. De plus, les ligatures sont sauvagement ignorées, ce qui est normal : fi en mode maths (fi) signifie le produit de f par i et ne doit donc pas être ligaturé puisque ce sont deux entités distinctes.

Toutefois, LaTeX est quand même gentil, il accepte de faire des accents en mode mathématiques, mais il faut alors le demander différemment de ce qui se fait pour le texte. Il devient nécessaire de se conformer au tableau 3.1 page suivante.

En mode mathématiques, en général, il ne faut pas tenir compte des espaces. L'espacement des différents symboles dans le résultat final sera principalement obtenu d'après le rôle, et le sens usuel, des différents opérateurs. Cependant, il faut tout de même ponctuer certaines phrases mathématiques avec des blancs pour mieux en indiquer le sens. Pour ce faire, on indiquera *explicitement* quelle quantité de blanc il faut laisser grace aux commandes illustrées dans le tableau 3.2 page suivante.

Résultat	Commande
$\hat{a}$	`$\hat{a}$`
$\check{a}$	`$\check{a}$`
$\tilde{a}$	`$\tilde{a}$`
$\acute{a}$	`$\acute{a}$`
$\grave{a}$	`$\grave{a}$`
$\dot{a}$	`$\dot{a}$`
$\ddot{a}$	`$\ddot{a}$`
$\breve{a}$	`$\breve{a}$`
$\bar{a}$	`$\bar{a}$`
$\vec{a}$	`$\vec{a}$`

TAB. 3.1: Accents en mode mathématique

Commande	Description	Exemple
`\;`	grand espace	$xx\;x$
`\:`	espace moyen	$xx\:x$
`\,`	petit espace	$xx\,x$
`\!`	espace négatif	$xx\!x$
`\quad`	espace d'un cadratin	$xx\quad x$
`\qquad`	espace de deux cadratins	$xx\qquad x$

TAB. 3.2: Espaces en mode mathématique

Style	Sens
`plain`	C'est l'équivalent du « `theorem` » de LaTeX de base.
`break`	L'en-tête du théorème est séparé du corps par un changement de ligne.
`marginbreak`	Comme `break` mais le numéro du théorème est dans la marge.
`changebreak`	Comme `break` mais en échangeant le titre et le numéro.
`change`	Comme `plain` mais en échangeant le titre et le numéro.
`margin`	Comme `plain` mais le numéro est mis dans la marge.

TAB. 3.3: Styles de théorèmes acceptés par le package `theorem`

3.8 Le package `undertilde`

C'est un package de Benjamin BAYART[10] [10].

On a souvent demandé dans les news comment produire un tilde *sous* une ou plusieurs lettres en mode maths, et non pas au dessus.

J'ai donc écrit ce petit package qui fournit la commande `\utilde`, qui se comporte comme `\tilde`, mais en plaçant le tilde sous la lettre.

Par exemple :

$$\underset{\sim}{abc} \neq \underset{\sim}{ab} \neq \underset{\sim}{a} \neq \tilde{a}$$

Qui s'obtient à partir de ce source :

```
\[ \utilde{abc} \neq \utilde{ab} \neq
   \utilde{a} \neq \tilde{a} \]
```

3.9 Le package `theorem`

L'environnement standard LaTeX étant limité[11] ou assez complexe à utiliser, Franck MITTELBACH [107] a concocté un petit package d'extension qui, bien que ne permettant pas de réaliser plus simplement ce que j'aime utiliser comme environnement pour les théorèmes, permet toutefois un paramétrage assez sympathique.

L'idée forte de ce package est qu'un théorème (ou tout ce qui s'en rapproche : définition, lemme, proposition...) c'est *un énoncé portant un nom, généralement mis en évidence par des espaces et des changements de fonte.*

Pour permettre de spécifier la façon dont on doit mettre en page un tel type de « théorème », ce package a été écrit, qui contient quelques commandes de réglages simples.

En premier lieu, tu dois régler le style du « théorème » en choisissant dans la liste du tableau 3.3. Pour cela tu utiliseras la commande

```
\theoremstyle{style a utiliser}
```

Cette commande effectue un changement global, c'est-à-dire que tous les types de « théorème » qui seront définis après seront dans ce style là, jusqu'à la prochaine apparition de cette commande.

Une autre commande permet de choisir la fonte du corps du théorème. Attention, il faut indiquer la fonte à l'ancienne manière. Cette commande est

```
\theorembodyfont{\upshape}
```

[10] Comme quoi des fois je fais des choses.

[11] Ce package étant vraiment plus confortable à utiliser, j'ai préféré passer sous silence l'environnement `theorem` de LaTeX.

pour décider que les prochains « théorèmes » seront en caractères droits au lieu d'être en italique par défaut. Attention, le « chemin » de fonte est pris depuis la fonte par défaut, tu peux donc être amené à spécifier des choses un poil plus complexes[12]. Si un argument vide lui est passé, on revient à la fonte par défaut.

Une troisième commande permet de choisir la fonte de l'en-tête du « théorème ». Elle fonctionne comme la précédente :

```
\theoremheaderfont{\scshape}
```

Une telle déclaration ne doit apparaître que dans le préambule et ne saurait en rien être modifiée. Si tu souhaites que tes lemmes et tes définitions n'aient pas la même fonte pour leur en-tête, il te faudra revenir aux anciennes méthodes qui sont toujours valables[13].

Un dernier petit détail : on peut effectuer un réglage supplémentaire sur l'espacement des théorèmes :

```
\theorempreskipamount et \theorempostskipamount
```

sont deux longueurs élastiques qui permettent de positionner convenablement tous les théorèmes.

Un exemple :

```
Dans le preambule:
\theoremheaderfont{\scshape}
\theoremstyle{break}
\theorembodyfont{\upshape}
\newtheorem{Def}{D\'efinition}[subsection]

Dans le texte:
\begin{Def}[Suite convergente]
$\forall(u_n)\in\mathbb{R}^\mathbb{N}\quad
\forall\ell\in\mathbb{R}$ si
\[ \forall\varepsilon\in\mathbb{R}_+ \quad
\exists n_0\in\mathbb{N} \quad \forall n\in\mathbb{N} \quad
n\geqslant n_0\Rightarrow |u_{n}-\ell|\leqslant\varepsilon \]
On dira que $u_n$ converge vers $\ell$

\noindent On notera:
\[ \lim_{n\rightarrow +\infty}u_n=\ell \]
\end{Def}
```

Cela produit la définition suivante[14] :

Définition 3.9.1 (Suite convergente)
$\forall(u_n)\in\mathbb{R}^{\mathbb{N}} \quad \forall\ell\in\mathbb{R}$ si

$$\forall\varepsilon\in\mathbb{R}_+ \quad \exists n_0\in\mathbb{N} \quad \forall n\in\mathbb{N} \quad n\geqslant n_0\Rightarrow |u_n-\ell|\leqslant\varepsilon$$

On dira que u_n converge vers ℓ
On notera :

$$\lim_{n\to+\infty} u_n=\ell$$

[12] Par exemple, si la fonte par défaut de ton document est le Sans-serif et que tu veux que le nom soit en gras romain, il faudra faire appel à deux commandes, `\rmfamily` pour passer en romain et `\bfseries` pour le gras. Le point de départ est bien la fonte par défaut de *ton* document, pas la fonte par défaut de LaTeX.

[13] Valables, certes, mais pas agréables. C'est pour ça que je n'en parle même pas. Si vraiment ça t'intéresse, je te recommande le livre de Leslie Lamport [85].

[14] La fonte utilisée ici pour `\mathbb` est celle fournie par `amsmath`.

3.10 Le package `subeqnarray`

Ce petit package, que nous devons à Johannes BRAAMS [17] permet bien des jolies choses pour la manipulation des équations en mode mathématique.

Un environnement (`equation`, que j'ai peu l'habitude d'utiliser) permettait déjà de numéroter les équations. Comme par exemple :

$$f(x) = \sin(x^2) \tag{3.2}$$

Il existe même un environnement, `eqnarray`, permettant de mettre plusieurs équations dans un tableau :

$$\begin{aligned} f(x) &= \sin(x^2) && (3.3)\\ &= \frac{e^{ix^2} - e^{-ix^2}}{2i} && (3.4)\\ &= \frac{e^{ie^{2\log(x)}} - e^{-ie^{2\log(x)}}}{2i} && (3.5) \end{aligned}$$

à partir du source suivent :

```
\begin{eqnarray}
f(x) & = & \sin(x^2) \\
     & = & \frac{e^{ix^2}-e^{-ix^2}}{2i} \\
     & = & \frac{e^{ie^{2\log(x)}}-e^{-ie^{2\log(x)}}}{2i}
\end{eqnarray}
```

Mais, si tu regardes avec une attention soutenue ce que tu viens de lire, tu te rendras compte bien vite qu'il ne s'agit que d'une seule équation en trois volets, et qu'il n'y a pas de raison de donner un nouveau numéro à chaque volet de cette unique équation. On peut alors indiquer qu'il s'agit d'une seule équation et la rentrer sous forme d'un tableau, mais alors il n'y a pas de moyen simple de référencer chaque volet individuellement. Bref un problème insoluble.

Enfin presque. Il y a maintenant le package `subeqnarray` qui permet d'entrer une équation en plusieurs volets en les sous-numérotant. Un exemple rapide.

$$\begin{aligned} f'(x) &= (x^2)'\cos(x^2) && (3.6a)\\ &= 2x\cos(x^2) && (3.6b)\\ &= 2x\frac{e^{ix^2} + e^{-ix^2}}{2} && (3.6c) \end{aligned}$$

On peut, très facilement, parler du volet 3.6b de l'équation 3.6. D'ailleurs voici le source de l'équation 3.6 :

```
\begin{subeqnarray}
f^\prime(x) & = & (x^2)^\prime\cos(x^2)
                  \label{eq}\slabel{seq1} \\
            & = & 2x\cos(x^2) \slabel{seq2}\\
            & = & 2x\frac{e^{ix^2}+e^{-ix^2}}{2}
                  \slabel{seq3}
\end{subeqnarray}
```

`\slabel` est une variante de `\label` qui au lieu de sauvegarder bêtement le dernier compteur incrémenté sauvegarde le numéro de la dernière sous-équation (ou du dernier volet). C'est bête comme chèvre, mais il fallait y penser.

Commande	Résultat
`\overrightarrow{...}`	$\overrightarrow{A_{i,j}B_{k,l}}$
`\underrightarrow{...}`	$\underrightarrow{A_{i,j}B_{k,l}}$
`\overleftarrow{...}`	$\overleftarrow{A_{i,j}B_{k,l}}$
`\underleftarrow{...}`	$\underleftarrow{A_{i,j}B_{k,l}}$
`\overleftrightarrow{...}`	$\overleftrightarrow{A_{i,j}B_{k,l}}$
`\underleftrightarrow{...}`	$\underleftrightarrow{A_{i,j}B_{k,l}}$

TAB. 3.4: Flèches étendues — package `amsmath`

3.11 Constructions avancées (amsmath)

3.11.1 Intégrales

En premier lieu voyons les intégrales multiples. Un exemple, c'est très parlant :

```
\[ \iint_V\neq\iint\limits_V \]
\[ \iint\neq\iiint\neq\iiiint\neq\idotsint \]
```

Pour voir les deux équations suivantes :

$$\iint_V \neq \iint\limits_V$$

$$\iint \neq \iiint \neq \iiiint \neq \int\cdots\int$$

Sans commentaire.

3.11.2 Flèches

Quelques petits jeux sur les flèches (pour les vecteurs, par exemple). Juste pour s'amuser. On avait déjà `\over...arrow`, l'$\mathcal{AMS}$ nous offre les même en dessous (`\under...arrow`).

```
\[\overrightarrow{A_{i,j}B_{k,l}}\]
\[\underrightarrow{A_{i,j}B_{k,l}}\]
```

produira

$$\overrightarrow{A_{i,j}B_{k,l}}$$

$$\underrightarrow{A_{i,j}B_{k,l}}$$

Et ça marche même en indice ou en exposant (même la pointe de la flèche !). On pourra également utiliser les mêmes variantes avec `left` à la place de `right` pour obtenir des flèches vers la gauche, ou encore `leftright` pour obtenir des flèches dans les deux sens.

Les six commandes en question sont reprises dans le tableau 3.4.

Commande	Ex. 1 Appel	Ex. 2 Appels
\Acute	$\acute{A}$	$\acute{\acute{A}}$
\Bar	$\bar{A}$	$\bar{\bar{A}}$
\Breve	$\breve{A}$	$\breve{\breve{A}}$
\Check	$\check{A}$	$\check{\check{A}}$
\Dot	$\dot{A}$	$\dot{\dot{A}}$
\Ddot	$\ddot{A}$	$\ddot{\ddot{A}}$
\Grave	$\grave{A}$	$\grave{\grave{A}}$
\Hat	$\hat{A}$	$\hat{\hat{A}}$
\Tilde	$\tilde{A}$	$\tilde{\tilde{A}}$
\Vec	$\vec{A}$	$\vec{\vec{A}}$

TAB. 3.5: Accents doublés (ajout `amsmath`)

3.11.3 Accents

Les accents mathématiques usuels de LaTeX fonctionnent bien, mais ne peuvent pas apparaître deux fois sur la même lettre. Certains pervers auraient pourtant bien besoin de telles notations. Par exemple les physiciens qui ont l'habitude (étrange) de noter par un accent « point » la dérivée, et qui du coup notent par un double, voire triple, point la dérivée seconde ou troisième.

Par exemple :

$$\frac{\partial y}{\partial t} = \dot{y} \quad \frac{\partial \dot{y}}{\partial t} = \ddot{y} \quad \frac{\partial \ddot{y}}{\partial t} = \dddot{y} \quad \frac{\partial \dddot{y}}{\partial t} = \ddddot{y}$$

Des accents mathématiques ont été redéfinis pour qu'on puisse en mettre deux sur la même lettre sans trop de difficulté. Pour en savoir plus, voir tableau 3.5.

3.11.4 Cadre

On peut dorénavant encadrer des formules :

```
\[ \boxed{\sum_iu_i}\neq
\boxed{\sum_iv_i} \]
```

produira

$$\boxed{\sum_i u_i} \neq \boxed{\sum_i v_i}$$

Les flèches à rallonge : c'est très pratique pour certaines formules, mais ça parasite pas mal LaTeX dans ses habitudes.

```
\[\xleftarrow{texte}\]
\[\xleftarrow[texte long]{texte}\]
\[\xrightarrow{texte}\]
\[\xrightarrow[texte long]{texte}\]
```

donnera

$$\xleftarrow{texte}$$
$$\xleftarrow[textelong]{texte}$$
$$\xrightarrow{texte}$$
$$\xrightarrow[textelong]{texte}$$

3.12 Options de chargement de $\mathcal{AMS}$-LaTeX

Les extensions de l'$\mathcal{AMS}$ sont nombreuses et font l'objet de plusieurs packages qui tous ensemble portent le joli nom de $\mathcal{AMS}$-LaTeX.

Un package particulier est `amsmath` puisqu'il comprend entre autres les trois packages `amstext`, `amsbsy`, et `amsopn` sans le dire.

3.12.1 Packages d'extensions

`amsmath` Le monstre décrit ci-dessus. Il fait appel à `amstext`, `amsbsy` et `amsopn` et fournit tout un tas de constructions mathématiques que je ne documente pas.

Il prend tout un tas d'options possibles :

intlimits bornes au-dessus et au-dessous des intégrales

nointlimits bornes à côté des intégrales

sumlimits pareil pour les sommes

nosumlimits ben, euh, je te fais un dessin?

namelimits comme pour `amsopn`

nonamelimits comme pour `amsopn`

leqno numéros d'équations à gauche

reqno numéros d'équations à droite

centertags je sais pas

tbtags je sais pas non plus

fleqn option globale prise en compte (les équations hors-texte sont mises à gauche et non pas centrées)

Par défaut les options nointlimits, sumlimits, namelimits, et centertags sont positionnées.

`amstext` Crée le pseudo-alphabet `\text`.

`amsbsy` Pour les alphabets `\boldsymbol` et `\pmb`.

`amsopn` Retouche les `\lim`, `\max` et autres commandes du même accabit. L'option nonamelimits doit interdire de mettre les commandes du type `\lim` ou du moins rendre innactif le positionnement de l'indice en dessous. Par exemple en produisant $\lim_{n\to+\infty} u_n = \ell$ au lieu de $\lim\limits_{n\to+\infty} u_n = \ell$

`amsthm` S'amuse avec les théorèmes. Le package `theorem` de Frank Mittelbach est meilleur, voir à ce sujet la section 3.9 page 77.

`amsintx` Pour les choses rigolotes sur les intégrales.

`amscd` Pour les diagrammes commutatifs, mais j'en cause pas.

`amsxtra` Trop compliqué pour que je te raconte. Pas très utile.

`amsgen` Chargement de quelques macros utiles aux autres packages. Sera chargé automatiquement.

3.12.2 Packages pour les symboles

L'$\mathcal{AMS}$ fournit les packages suivants pour définir de nouveaux symboles et/ou de nouveaux alphabets.

`amsfonts` : Pour charger les fontes, inintéressant. L'option psamsfonts est reconnue.

`amssymb` : Pour charger les symboles, appel tout seul `amsfonts`, reconnaît l'option psamsfonts décrite plus loin.

`euscript` : Pour charger l'alphabet `\EuScript`. Admet trois options :

psamsfonts utilise les versions PostScript des fontes. Tu ne les a probablement pas et ça sert grosso-modo à pas grand chose[15].

mathcal `\mathcal` devient alors équivalent à `\EuScript`.

mathscr `\mathscr` est créé et est équivalent à `\EuScript`.

`eufrak` : Pour charger l'alphabet `\EuFrak`. Admet l'option psamsfonts.

`eucal` : Comme `euscript` avec l'option mathcal par défaut.

3.13 Le package `vector`

Ce package, que nous devons au britanique Nick EFFORD [47], étend considérablement le traditionnel « accent » mathématique `\vec` (cf. tableau 3.1 page 76).

Je te rappelle, avant d'aller plus loin, que `\vec{a}` produit $\vec{a}$. C'est la notation la plus usuelle en mathématiques, jusqu'à la terminale. D'autres sont prévues, pour les physiciens par exemple :

`\uvec{a}`	$\underline{a}$
`\bvec{a}`	$\mathbf{a}$
`\svec{a}`	a

De plus on note souvent les vecteurs de norme 1 avec un accent circonflexe. C'est en effet ce qui est prévu ici pour les « `uvec` » (unary vector) :

`\uuveca`	$\underline{\hat{a}}$
`\buveca`	$\hat{\mathbf{a}}$
`\suveca`	$\hat{\mathsf{a}}$

D'autres commandes sont prévues, pour taper directement les vecteurs et non plus simplement leurs noms :

```
\[ (\rvec{x}{1}{5}) \]
\[ \left( \cvec{x}{1}{5} \right) \]
```

pour obtenir :

$$(x_1, x_2, x_3, x_4, x_5)$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

On manipulera souvent des vecteurs de taille n :

[15] Il parait que dans certains cas c'est utile, sur MacIntosh entre autres, mais je suis pas sûr. De nos jours, pour une utilisation avec pdfTEX, ça permettra de produire du PDF plus facile à lire avec Acrobat Reader.

```
\[ (\irvec{x}) \]
\[ \left(\icvec{x}) \]
```

pour obtenir :

$$(x_1, \ldots, x_n)$$

$$\left(\begin{array}{c} x_1 \\ \vdots \\ x_n \end{array} \right)$$

On pourra vouloir spécifier l'indice du dernier terme :

```
\[ (\irvec[k]{x}) \]
\[ \left( \icvec[12]{x} \right) \]
```

pour obtenir :

$$(x_1, \ldots, x_k)$$

$$\left(\begin{array}{c} x_1 \\ \vdots \\ x_{12} \end{array} \right)$$

Ou encore changer l'indice du premier élément :

```
\[ \firstelement{0}
\left( \icvec[12]{x} \right) =
(\irvec[12]{x})^t \]
```

nous donnera :

$$\left(\begin{array}{c} x_0 \\ \vdots \\ x_{12} \end{array} \right) = (x_0, \ldots, x_{12})^t$$

Voilà, tu devrais tout savoir faire sur les vecteurs maintenant.

3.14 Tableaux d'Young — package `young`

Le petit package utilitaire `young` que nous devons à Jörg KNAPPEN [74] permet de taper facilement des tableaux d'YOUNG.

Par exemple, ce tableau :

1	2	3
4	5	
6	7	
8		

a été produit par :

```
\begin{Young}
1 & 2 & 3 \cr
4 & 5     \cr
6 & 7     \cr
8         \cr
\end{Young}
```

Voilà, ça se passe de commentaires.

Ah, si, tout de même, un détail. J'ai eu du mal à centrer le tableau d'YOUNG dans la page. Pour de sombres raisons, l'environnement `Young` produit seulement un objet de type « vertical ». Quand on le croise dans un paragraphe, il est traité comme si c'était une grosse lettre.

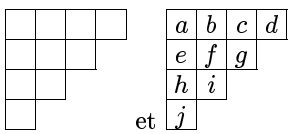
Si on veut le centrer, il faudra le faire précéder ou suivre de quelque chose qui nous fasse sortir du mode vertical. Ça peut sembler un peu technique, mais, dans la pratique, ça donne quelque chose comme ça :

```
\begin{center}
\leavevmode
\begin{Young}
...
\end{Young}
\end{center}
```

3.15 Tableaux d'Young — package `youngtab`

Ce package permet lui aussi de mettre en forme des tableaux d'Young, mais avec une syntaxe très différente. Bien entendu, l'auteur la trouve plus jolie. Question de goût, sans doute. Ce package est écrit et maintenu par Volker BÖRCHERS et Stefan GIESEKE [12].

Le principe est relativement simple, pour un usage de base. Pour obtenir



il suffit de taper :

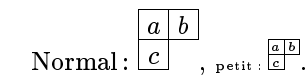
```
\yng(4,3,2,1) et \young(abcd,efg,hi,j)
```

3.15.1 Gestion de la taille du tableau : option autoscale

Par défaut, `youngtab` fait changer la taille du tableau en fonction du corps de la fonte en cours, par exemple :

```
Normal: \young(ab,c), {\tiny petit: \young(ab,c)}.
```

produira :



On peut désactiver cette fonctionnalité en passant l'option noautoscale au package. Curieusement, l'option autoscale *n'exsite pas*.

On peut également activer ou désactiver cette fonctionnalité à tout moment. Pour l'activer :

```
\Yautoscale1
```

et pour la désactiver :

```
\Yautoscale0
```

Si on désactive l'ajustement de la taille du tableau, on peut fixer cette taille à autre chose que la taille par défaut. Ça se fait en appelant[16] :

```
\Yboxdim{4mm}
```

Un appel à `\Yboxdim` implique la désactivation de l'ajustement de la taille.

3.15.2 En mode math ou en mode texte

En mode texte, les tableaux sont toujours alignés comme l'ont montré les exemples précédents. Point. Ce n'est pas négociable : il n'y a pas d'option prévue permettant un alignement vertical autre (par exemple avec la première ligne du tableau d'Young alignée sur le texte, ou avec le tableau centré sur le texte autour).

Par contre, en mode maths, c'est négociable. Par défaut, ça fonctionne comme en mode texte, mais tu peux changer ça. Soit de manière globale en passant l'option vcentermath au package :

```
\usepackage[vcentermath]{youngtab}
```

soit, à tout moment dans ton document en appelant :

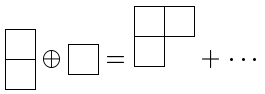
```
\Yvcentermath1
```

pour activer l'option, ou

```
\Yvcentermath0
```

pour la désactiver.

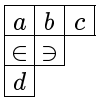
Par exemple, en jouant un peu avec cette option, on obtient :



à partir du source suivant :

```
\[ \Yvcentermath1
   \yng(1,1) \oplus \yng(1) =
   \Yvcentermath0
   \yng(2,1) + \cdots
\]
```

Par défaut, les symboles à l'intérieur du tableau d'Young sont en mode maths, ce qui permet des droleries, par exemple :



à partir du source suivant :

[16] Attention, la documentation du package est fausse sur ce point, elle prétend qu'on peut se passer des accolades, et elle se trompe. Ça ne marchera probablement pas dans tous les cas.

```
\young(abc,\in\ni,d)
```

Tu peux changer cet état de fait globalement, en passant l'option stdtext au package lors de son chargement ; ou à tout moment dans ton document en appelant

```
\Ystdtext1
```

pour passer en mode texte, ou

```
\Ystdtext0
```

pour passer en mode maths.

Ainsi, on peut obtenir

$$\begin{array}{|c|c|c|}\hline a&b&c\\\hline d&e\\\cline{1-2}f\\\cline{1-1}\end{array}\neq\begin{array}{|c|c|c|}\hline a&b&c\\\hline d&e\\\cline{1-2}f\\\cline{1-1}\end{array}$$

à partir de :

```
\[ \Ystdtext1
   \young(abc,de,f) \neq
   \Ystdtext0
   \young(abc,de,f)
   \]
```

3.15.3 Constructions plus complexes

Bien entendu, en mode texte, hors de question d'appeler `\in`, ou tout autre symbole math directement. Il va falloir ruser. Par exemple, pour obtenir :

$$\begin{array}{|c|c|c|}\hline a&b&c\\\hline \in&\ni\\\cline{1-2}d\\\cline{1-1}\end{array}\neq\begin{array}{|c|c|c|}\hline a&b&c\\\hline \in&\ni\\\cline{1-2}d\\\cline{1-1}\end{array}$$

on utilisera le source suivant[17] :

```
\newcommand\textin{$\in$}
\newcommand\textni{$\ni$}
\[ \Ystdtext1
   \young(abc,\textin\textni,d) \neq
   \Ystdtext0
   \young(abc,\in\ni,d)
   \]
```

En fait, c'est vrai pour n'importe quoi : si tu veux mettre un objet un peu compliqué dans un tableau d'Young, il faut passer par une macro. Mais il faut absolument que le texte produit tienne dans la case, sans quoi il dépassera sur la case à gauche et le résultat sera très laid.

[17] Pour une explication sur `\newcommand`, reportes-toi au chapitre neuvième page 283 qui explique comment on écrit des macros.

Par exemple pour jouer un peu avec le gras en mode mathématique (attention, les nuances sont subtiles et ne survivent pas bien à une mauvaise imprimante, ou à une photocopie)[18] :

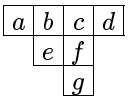
à partir du source suivant :

```
\[ \newcommand\ba{\bm{a}}
   \newcommand\bCa{\bm{\hat}{a}}
   \newcommand\bCA{\bm{\hat{a}}}
   \newcommand\ca{\hat{a}}
   \young(\bCA\bCa,\ca,\ba)
   \]
```

3.15.4 Les tableaux à trous

Par défaut, les tableaux d'Young sont alignés à gauche, et ce n'est pas négociable. Si on souhaite faire des tableaux dont les premières colonnes puissent être vides (même pas de cadre, rien, vide), il faut passer l'option `enableskew` au package lors de son chargement. Curieusement, cette option ne peut pas être activée ou désactivée après le chargement.

Le principe est simple, pour obtenir :



il faut faire appel à :

```
\[ \noextrasfrancais
   \young(abcd,:ef,::g) \]
```

Évidemment, comme tout ce qui utilise le deux-points, ça crée des petits problèmes avec `babel`, il faut donc respecter les règles suivantes pour pouvoir combiner les deux :

- charger `babel` *avant* `youngtab`,
- désactiver les extensions de `babel` pour le français *juste avant* l'appel à `\young`,
- utiliser `\young` comme on veut,
- réactiver les extension du français juste après.

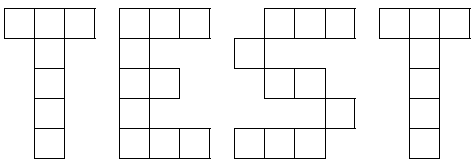
Ainsi, par exemple, si on veut utiliser `\young` dans ce mode là en plein milieu d'un paragraphe, et qu'on ne veut pas fausser la typographie de celui-ci, on pourra procéder comme ça : , à partir de ce source là :

```
... proc\'eder comme \c{c}a: {\noextrasfrancais\young(ab,:c)},
```

Les deux accolades englobantes sont très importantes. `\noextrafrancais` désactive les extensions du français *dans le groupe où il se trouve*, du coup, lors de la sortie du groupe (juste après l'accolade fermante), ces extensions sont réactivées. Beaucoup de constructions forment un groupe sans le dire, en particulier les commande permettant de passer en mode maths. Dans le doute, au pire, rajouter un niveau de groupe (*i.e.* un niveau d'accolades) ne coûte rien, et comme ça on est sur de ce qu'on fait.

Pour finir, un exemple idiot :

[18] Cette utilisation du package `bm` est incompatible avec l'utilisation de `amsmath`, mais je n'ai pas très bien compris pourquoi, ne sois donc pas surpris si tu utilises ces deux packages et que ça ne marche pas.



qui s'obtient à partir de :

```
\begin{center}
\noextrasfrancais
\young(~~~,:~,:~,:~,:~)\hspace{2mm}
\young(~~~,~,~~,~,~~~)\hspace{2mm}
\young(:~~~,~,:~~,:::~,~~~)\hspace{2mm}
\young(~~~,:~,:~,:~,:~)
\end{center}
```

3.16 Divisions non-commutatives de Maxwell

Un package, `holtplot`, que nous devons à Jörg KNAPPEN [79], permet de mettre en page de telles divisions.

Un exemple y suffira :

$$\left\lfloor \begin{array}{c} a \\ b \end{array} \right\rfloor \qquad \left\lceil \begin{array}{c} c \\ d \end{array} \right\rceil$$

que l'on obtient à partir de

```
\[ \holter{a}{b} \qquad \polter{c}{d} \]
```

3.17 Le package `dotseqn`

Ce petit package très pratique, nous le devons à Donald ARSENEAU [7]. Il permet d'obtenir une mise en page des équations assez particulière[19] : les équations sont mises au fer à gauche, comme lorsque l'on utilise l'option `fleqn` de `\documentclass`, le numéro de l'équation se retrouve à droite, comme d'habitude, et des pointillés joignent les deux.

Ce package affecte les environnements `equation` et `eqnarray`, par contre, et c'est logique, leurs versions ne numérotant pas les équations (`displaymath` et `eqnarray*`) ne sont pas affectés. Dans sa version actuelle, à savoir 1.1, il semble que ce package ne soit pas capable d'une interaction intéressante avec `subeqnarray`, ce qui est bien dommage.

Ce package ajoute tout seul l'option `fleqn` à `\documentclass`, et interdit `leqno`. Il accepte deux options : `leftjust` pour que la colonne de gauche des `eqnarray` soit alignée à gauche et pas à droite comme c'est le cas par défaut ; et `nocolsep` qui supprime l'espace disgracieux que `eqnarray` ajoute tout seul entre les colonnes, et qui n'a pas lieu d'être dans une équation. La seconde option n'est pas prise par défaut, pour ne pas changer inutilement le comportement standard de LaTeX, cependant Donald la recommande fortement.

Sans ce package, une équation obtenue par :

[19] Dixit Donald, cette mise en page est assez classique dans certains ouvrages de mathématiques anglo saxons anciens, cependant, je ne la recommande pas particulièrement, elle entraine une plus grande lourdeur des pages.

```
\begin{equation}
a+b=x
\end{equation}
```

ressemble à ça :

$$a+b=x \tag{3.7}$$

Avec le package `dotseqn`, ça donne ça :

$$a+b=x \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \tag{3.8}$$

Chapitre deuxième
Mathématiques

4 Alphabets mathématiques

4.1 Définition

Un « alphabet mathématique » est, à peu près, aux mathématiques sous LaTeX ce que les fontes sont aux textes. Il ne s'agit pas vraiment de fontes, puisque dans une fonte donnée on a plusieurs alphabets. Par exemple, dans la fonte usuelle sous LaTeX qui s'appelle Computer Modern, on trouve des alphabets différents tels que `\mathbb` pour obtenir les lettres à double barre, ou `\mathcal` pour obtenir des lettres rondes.

Je vais te montrer dans cette section tout ce qui ressemble de près ou de loin à des alphabets pour LaTeX. Ne te fie pas trop à ce que je raconte, il m'arrive de te mentir : certaines des choses présentées ici *ne sont pas* des alphabets, mais simplement des commandes permettant de modifier l'aspect visuel des caractères produits.

4.2 Gras (`amsmath`)

Si tu souhaites obtenir des symboles en gras dans une formule mathématique, sans pour autant que toute la formule soit en gras, deux choix s'offrent à toi : soit le symbole existe en gras, et alors tu utiliseras :

```
\boldsymbol{...}
```

soit le symbole n'existe pas en gras, et alors tu utiliseras

```
\pmb{...}
\mathop{\pmb{...}}
```

la deuxième forme servant à produire des opérateurs du type `\sum`. Un exemple rapide et c'est bouclé :

```
\[ \sum_{i=1}^nu_i
\neq
 \boldsymbol{\sum}_{i=1}^nu_i
\neq
 \mathop{\pmb{\sum}}_{i=1}^nu_i
\]
```

Ça donne ça :

$$\sum_{i=1}^{n} u_i \neq \sum\nolimits_{i=1}^{n} u_i \neq \sum_{i=1}^{n} u_i$$

4.3 Gras (LaTeX 2_ε)

Pour faire du gras dans toute une équation voire dans toutes tes équations, il y a des méthodes moins brutales que celle vue ci-dessus, et en plus elles ne requièrent pas de package d'extension. LaTeX 2_ε prévoit une commande

```
\mathversion{...}
```

qui permet de commuter entre les versions `bold` et `normal` en standard.

Un exemple rapide :

```
\mathversion{bold}
\[ \sum_{i=1}^nu_i \]
\mathversion{normal}
\[ \sum_{i=1}^nu_i \]
```

Pour produire :

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{u_i}$$

$$\sum_{i=1}^{n} u_i$$

4.4 Le package `bm`

Le package `bm` (comme « bold math »), de David CARLISLE et Frank MITTELBACH [38], est livré normalement en standard avec LaTeX. Il permet d'accéder simplement aux symboles mathématiques en gras, par exemple :

```
\[ \alpha \neq \bm{\alpha} \]
```

produira :

$$\alpha \neq \boldsymbol{\alpha}$$

Ce package permet également de définir ses propres symboles en gras, par exemple, si tu t'es défini un symbole `\monsymbole`, que tu t'es définit le même en gras (`\bmonsymbole`), en utilisant `\bmdefine` tu indiqueras qu'un appel à `\bm` sur `\monsymbole` doit utiliser `\bmonsymbole` :

```
\bmdefine\bmonsymbole{\monsymbole}
```

Un petit exemple[20] :

```
\newcommand\xx{\mathrm{xx}}
\newcommand\bxx{\mathbf{xx}}
\bmdefine\bxx{\xx}

\[ \bm{\sqrt{\xx}} \neq \sqrt{\xx} \]
```

$$\sqrt{\mathbf{xx}} \neq \sqrt{\mathrm{xx}}$$

[20] Pour une explication sur le fonctionnement de `\newcommand`, reporte toi au chapitre neuvième page 283.

Nom de l'alphabet	Package à inclure	Résultat produit
\mathcal	aucun	$\mathcal{ABC}$
\mathbf	aucun	$\mathbf{ABC}$
\mathsf	aucun	ABC
\mathtt	aucun	$\mathtt{ABC}$
\mathrm	aucun	ABC
\mathbb	amsfonts	$\mathbb{ABC}$
\mathbb	mathbbol	$\mathbb{ABC}$
\mathos	oldstyle	123
\boldsymbol	amsbsy $\subset$ amsmath	$\boldsymbol{ABC}$
\mathfrak	amsfonts ou eufrak	$\mathfrak{ABC}$
\EuFrak	eufrak	$\mathfrak{ABC}$
\EuScript	euscript	$\mathcal{ABC}$
\mathscr	mathrsfs	$\mathscr{ABC}$

Attention, \boldsymbol n'est pas à proprement parler un alphabet, c'est plus un « modifieur », comme \textbf en est un. On peut le combiner avec \EuScript, \mathfrak ou \mathcal. De même il agit sur les symboles et pas que sur les lettres latines.

TAB. 4.1: Exemples des différents alphabets mathématiques

4.5 Alphabets

Par défaut, LaTeX 2_ε ne connait qu'un alphabet mathématique rigolo, c'est \mathcal. On peut lui en apprendre d'autres. Deux très bons exemples sont les alphabets Euler Script et Euler Frakture (page 96).

Le package amsmath permet d'en créer 5 autres. Étudions les rapidement. \mathbb sers pour les lettres à double barre, \boldsymbol est expliqué plus haut avec \pmb. Ne reste plus que \mathfrak (ou \frak) qui redéfinit Euler Frakture[21] et \text[22] qui permet de placer du texte dans une équation. Le texte sera dans la même fonte que celle valide en dehors de l'équation :

```
\[ \sum_{i=1}^nu_i
\text{ un petit mot }
\sum_{i=1}^nu_i \]
```

Produira :

$$\sum_{i=1}^{n} u_i \text{ un petit mot } \sum_{i=1}^{n} u_i$$

Un exemple de l'alphabet le plus utile, à savoir \mathbb qui permet d'obtenir les lettres à double barre pour noter les ensembles :

```
\[ \forall n\in \mathbb{N}
   \quad u_n=v_{n+1} \]
```

[21] Très exactement comme le fait le package eufrak qui est exposé section 1.2 page 96, sauf que la commande définie ici s'appelle \mathfrak au lieu de \EuFrak.

[22] Attention cependant. Si tout cela se comporte comme des alphabets, tout n'en est pas ! En particulier, ce qui concerne le gras est chargé par amsbsy, \text est chargé par amstext, et le reste par amsfonts.

Produira :

$$\forall n \in \mathbb{N} \quad u_n = v_{n+1}$$

Un autre alphabet rigolo est offert par le package `oldstyle` (voir section 1.7 page 98 à ce sujet). C'est l'alphabet `\mathos` qui permet de produire ça : 123.

Le tableau 4.1 page précédente donne un exemple sur les lettres A, B et C de chaque alphabet en rappelant leur lieu de définition.

4.6 Le package `mathbbol`

Cet intéressant package, que nous devons à Jörg KNAPPEN [76], offre pas mal de possibilités autour d'une idée centrale : obtenir des lettres et symboles à double barre (comme $\mathbb{R}$) en mode mathématiques. L'alphabet utilisé est `\mathbb`. Bien entendu, c'est en conflit avec celui de l'$\mathcal{AMS}$.

Regardons rapidement les possibilités offertes par ce package :

`\mathbb` : l'alphabet mathématique `\mathbb`, ce n'est jamais que le quatre ou cinquième package à le proposer, il varie cependant un peu par rapport aux autres : $\mathbb{ABC}$;

`\Langle` et son petit frère `\Rangle` permettent d'obtenir des délimiteurs comme $\ll 2A, 3B \gg$;

`\Lbrack` et `\Rbrack` pour obtenir des crochets doubles[23] (par exemple pour dénoter les intervales entiers) : $\left[\!\left[\frac{12}{3}, \frac{18}{2} \right]\!\right]$;

`\Lparen` et `\Rparen` pour obtenir des parenthèses doubles, comme par exemple dans $(\!(\frac{1}{2}, \frac{5}{4})\!)$, mais je ne vois pas trop à quelle notation cela pourrait correspondre ;

`\Eins` : permet de produire le symbole $\mathbb{1}$ pour désigner les fonctions indicatrices.

`\bbalpha` : tout un alphabet de lettres grecques à double barre est prévu, lorsque l'on ajoute l'option `bbgreekl`. Cela donne par exemple : $\mathbb{\alpha}$, $\mathbb{\beta}$, $\mathbb{\partial}$, $\mathbb{\delta}$.

4.7 Le package `mathrsfs`

Ce package, dû à Jörg KNAPPEN [77], apporte un nouvel alphabet mathématique : `\mathscr` que certains préfèrent à `\mathcal`.

Un petit exemple : `\mathscr{A}` produira $\mathscr{A}$.

Sans commentaire.

[23] Si tu charges le package avec l'option `cspex`, alors ces crochets doubles deviendront à taille variable, sous réserve de disponibilité de la fonte correspondante (`stmaryrd`) sur ton système.