Écrire des formules avec le MathBot

Thomas Harbreteau

25 décembre 2018

Table des matières

1 Préambule

Écrire des mathématiques sur ordinateur est compliqué si l'on n'a pas les bons outils. En effet, les logiciels de traitement de texte tels que Microsoft Word ou Libre Office ont beau proposer un mode pour rentrer des formules, quiconque l'a utilisé s'est vite rendu compte que ce n'était pas d'une praticité extrême... Fort heureusement, pour répondre aux besoins de la communauté scientifique, le mathématicien Donald Knuth créa en 1977 le logiciel libre TeX, signifiant *art, science* en grec ancien, destiné à la rédaction de documents scientifiques. En 1983, c'est Leslie Lamport qui apporte sa contribution au projet en mettant au point le désormais célèbre langage LaTeX, abréviation de *Lamport TeX*, qui est un assemblage de diverses macros rendant la composition d'un document TeX nettement plus simple.

Dans ce document, nous n'aborderons pas l'écriture en LaTeX en général, mais nous nous concentrerons sur l'écriture des formules mathématiques reconnues par le bot MathBot sur Discord. Cependant, si vous devez régulièrement rédiger des documents sientifiques, je vous conseille fortement d'apprendre à utiliser ce formidable outil. Parmis ses avantages, on peut citer une mise en forme automatique, propre et épurée, bien sûr des formules facilement insérables dans un document, ainsi que la possibilité de créer des diaporamas.

Je me suis efforcé d'agrémenter le document d'exemples, mais rien ne vaut un peu de pratique, ainsi je vous encourage fortement à manipuler les commandes pour vous familiariser avec la langage. Certains paragraphes, notamment celui sur `operatorname` et `mathrm` n'ont sans doutes pas leur place dans ce document, qui se veut être une introduction au LaTeX, mais j'ai décidé de les laisser pour les lecteurs souhaitant éventuellement rentrer plus dans les détails. Sur ce, bonne lecture !

2 Utiliser le MathBot

2.1 Indiquer qu'une formule doit être interprétée par le bot

Rien de plus simple, il suffit de commencer votre message sur Discord par la séquence

```
=tex
```

puis écrire votre formule. Par exemple, la commande `=tex 40 + 2 = 42` renverra $40 + 2 = 42$.

2.2 Écrire des maths

Et oui, j'imagine que vous ne lisez pas ce document pour apprendre à tricoter (ou autrement vous êtes perdu). Avant de commencer, une petite parenthèse sur l'écriture en LaTeX pour mieux comprendre comment fonctionnent nos commandes avec le bot : le langage distingue les environnements *maths*, *affichage de maths* et *texte*, et ce grâce aux délimiteurs `$$` (affichage de maths) ou `$` (maths). Le `$` est utilisé pour écrire des maths dans un bloc de texte, par exemple la commande

```
$\sum_{n=1}^{+\infty} \frac{(-1)^n}{n} = -\ln 2$
```

écrit avec des simples `$` renverra $\sum_{n=1}^{+\infty} \frac{(-1)^n}{n} = -\ln 2$. C'est comme cela que l'on écrit les objets mathématiques en lettres penchées dans un texte, tels que soit f la fonction, qui s'écrira `soit $f$ la fonction`. En effet, dans un environnement maths, les lettres sont par défaut penchées. Avec les doubles `$$`, le compilateur ira à la ligne, centrera l'expression et affichera les symboles mathématiques en mode *display*, avec les grands symboles, grandes fractions et compagnie.

$$\sum_{n=1}^{+\infty} \frac{(-1)^n}{n} = -\ln 2.$$

Mais revenons à nos moutons, lorsque l'on utilise le MathBot, pas besoin de ces distinctions d'environnements, tout ce qui est écrit après le `=tex` est considéré comme étant dans un environnement affichage de maths. Ainsi, les chiffres et opérations de base, addition, soustraction utilisent les symboles du clavier, `+`, `-` ou `1515`. Le symbole de multiplication s'obtient avec la commande `\times`, qui renvoie $\times$, mais on préfèrera souvent ne rien mettre, surtout dans des expressions littérales, avec des chiffres et des lettres. Pour les fractions, c'est légèrement plus compliqué. Il y a plusieurs commandes, mais on utilisera majoritairement la

commande `\frac{a}{b}` qui renverra

$$\frac{a}{b}.$$

Il existe aussi les variantes `\dfrac` et `\tfrac`, qui imposent la taille des caractères dans la fraction, normaux pour `\dfrac` et petits pour `\tfrac`. Le *d* signifie *display*, c'est la typographie utilisée par LaTeX pour afficher une fraction dans un environnement affichage de maths, tandis que le *t* signifie *text*, c'est la typographie utilisée pour écrire une fraction à l'intérieur d'un bloc de texte. La différence avec `\frac` est que ce dernier gère automatiquement la typographie à utiliser. Pour une illustration, voir ci-dessous.

Commande	Résultat
`\tfrac{\tfrac{a}{b}}{c}`	$\tfrac{\tfrac{a}{b}}{c}$
`\dfrac{\dfrac{a}{b}}{c}`	$\dfrac{\dfrac{a}{b}}{c}$
`\dfrac{\frac{a}{b}}{c}`	$\dfrac{\frac{a}{b}}{c}$
`$\frac{3}{4}$ des chats`	$\frac{3}{4}$ des chats
`$\dfrac{3}{4}$ des chats`	$\dfrac{3}{4}$ des chats

Petite remarque concernant les fractions, dans du texte on préferera écrire la division avec le symbole de base `/`, comme dans a/b, plutôt qu'avec une fraction, $\frac{a}{b}$ ou $\dfrac{a}{b}$, ceci étant plus lisible et ne cassant par l'interligne.

Abordons maitenant le cas des exposants et indices. Pour mettre un caractère en exposant, il suffit de lui mettre un chapeau devant, `a^2` renverra a^2. Pour une chaîne de caractères, il faut la délimiter avec des accolades, `a^{42}` renverra a^{42}. De même, pour mettre un caractère (resp. une chaîne de caractères) en indice, il suffit de le (resp. la) faire précéder d'un tiret du huit, ou *tiret du bas* pour les intimes, ainsi `a_2` (resp. `a_{42}`) renverra a_2 (resp. a_{42}). On peut s'amuser à essayer d'écrire une chaîne de caractères en exposant sans accolades, par exemple `a^42`, ce qui renvoie a^42... Ce qui n'est en fin de compte pas très amusant.

Enfin, on peut vouloir mettre des parenthèses autour d'un grand symbole, comme une fraction ou une somme. Les bêtes parenthèses `(\frac{42}{1515})` sont alors trop petites et renvoient

$$(\frac{42}{1515}).$$

Pour régler ce soucis, il faut utiliser les commandes `\left` et `\right` qui ont pour effet d'ajuster automatiquement la taille de ce qui les suit en fonction du contenu à l'intérieur. Par exemple, on utilisera `\left(\frac{42}{1515}\right)`, ce qui est nettement plus joli :

$$\left(\frac{42}{1515}\right).$$

Attention, il ne peut y avoir de `\left` sans `\right`, l'un sans l'autre provoquera une erreur. On peut appliquer ces commandes aux crochets ou parties entières (voir plus loin pour le symbole), et plein d'autres symboles que vous découvrirez en temps et en heures.

2.3 Écrire du texte

Le lecteur attentif s'exclamera : « Minute, puisque l'environnement affichage de maths affiche les caractères en lettres penchées, comment j'écris du texte en lettres droites moi ? » LaTeX étant bien fait, il est possible de modifier la typographie utilisée, par le biais des commandes listées ci-dessous (liste, bien entendu, non-exhaustive) :

Commande	Résultat
`\text{chat}`	chat
`\textit{chat}`	*chat*
`\textbf{chat}`	**chat**
`\underline{chat}`	<u>chat</u>

Dans un environnement maths, il suffit donc d'écrire son texte avec `\text`, comme par exemple `$1+1=2 \text{et} 2+2=4$` qui renverra

$$1+1=2\text{et}2+2=4.$$

Oulà, c'est moche, c'est tout serré ! En effet, LaTeX ne prend pas en compte vos espaces, vous aurez beau insérer avec votre barre d'espace 42 espaces entre chaque mot, après compilation, il n'y en aura plus qu'un. De plus, l'environnement maths supprime tout bonnement les espaces, de sorte que `$1+1=2 \text{et}2+2=4$` et la commande précédente renverront exactement le même résultat. Pour gérer l'espacement, il existe là aussi des commandes, synthétisées ci-dessous.

Commande	Résultat
`a \ b`	$a \ b$
`a \! b`	$a\!b$
`a \, b`	$a \, b$
`a \; b`	$a \; b$
`a \quad b`	$a \quad b$
`a \qquad b`	$a \qquad b$

En pratique, j'utilise personnellement le `\quad` le plus fréquemment, entre des quantificateurs et une expression par exemple :

$$\forall \varepsilon > 0, \exists N \in \mathbf{N}, \forall p, q \geq N, \quad d(u_p, u_q) < \varepsilon.$$

Enfin, mentionnons qu'il est possible de faire un retour à la ligne via la commande `\\`, que l'on se situe dans un environnement maths ou texte.

2.4 Les symboles mathématiques

Sans doutes ce qui vous intéressera le plus, car que seraient les mathématiques sans symboles bizzares utilisés à tord et à travers ? Ici pas de mystères, les symboles sont des macros composées de l'éternel backslash \ suivi (sans espace) du nom du symbole... En anglais. Et oui, le mot *pour tout* devient *for all*, *somme* devient *sum* et je ne vais pas tous les faire mais vous avez compris. Le tableau ci-dessous récapitule les symboles les plus usités.

Commande	Résultat	Commande	Résultat	Commande	Résultat
`\sum_{a}^{b}`	$\sum_{a}^{b}$	`\leq`	$\leq$	`\lfloor`	$\lfloor$
`\prod_{a}^{b}`	$\prod_{a}^{b}$	`\geq`	$\geq$	`\rfloor`	$\rfloor$
`\forall`	$\forall$	`\mapsto`	$\mapsto$	`\sqrt{a}`	$\sqrt{a}$
`\exists`	$\exists$	`\to`	$\to$	`\int_{a}^{b}`	$\int_a^b$
`\iff`	$\Leftrightarrow$	`\implies`	$\Longrightarrow$	`\iiint_{\tau}`	$\iiint_\tau$
`\infty`	∞	`\cup`	$\cup$	`\cap`	$\cap$

Les produits, sommes et intégrales obéissent aux mêmes règles que les exposants et indices, si il n'y a qu'un seul caractère, pas besoin des accolades, sinon, elles sont nécéssaires.

Toutes les lettres grecques sont également facilement accessibles, `\alpha` renverra α, `\Gamma`, Γ et ainsi de suite. Vous trouverez aisément des tables de caractères spéciaux sur Internet.

Concernant l'écriture des limites, le symbole *limite* est lim et s'obtient avec la commande `\lim`, qui peut recevoir un indice, pour indiquer en quel point la limite est regardée. Comme d'habitude, `\lim_{x \to 0} f(x)` renverra

$$\lim_{x \to 0} f(x).$$

On peut aussi empiler les indices à l'aide de la commande `\atop`, comme ceci : `\lim_{x \to 0 \atop x > 0} f(x)` renverra

$$\lim_{\substack{x \to 0 \\ x > 0}} f(x).$$

Le symbole *tend vers* n'existe pas vraiment, la commande `\to` crée une flèche ridiculement petite et sous laquelle on ne peut rien préciser (`\to_{a \to 0}` renvoie $\to_{a \to 0}$). Un bon compromis est le symbole `\xrightarrow[a]{}`, ce qui mis en contexte donne `f(x) \xrightarrow[x \to +\infty]{} 42` (attention à laisser l'accolade vide mais à l'écrire tout de même), qui renvoie

$$f(x) \xrightarrow[x \to +\infty]{} 42.$$

C'est plus joli, mais laisse une accolade vide, chose que certains peuvent considérer comme un crime. Rajouter un argument entre ces accolades rajoute cet argument au dessus de la flèche, comme ceci :

$$f(x) \xrightarrow[x \to +\infty]{\text{entre accolades}} 42.$$

Comme on peut le constater, la grande force de ce symbole est que sa longueur s'adapte à celle de ses indice et exposants.

Enfin, on va voir comment écrire un système d'équations. L'environnement `cases` ou `dcases` (d étant pour *display*, les mêmes remarques que sur les fractions s'appliquent) est très pratique, cependant sa syntaxe est probablement la plus compliquée rencontrée jusqu'à présent. Et oui, on n'a rien sans rien. L'idée est que l'environnement place une accolade devant notre système (pour que ce soit, justement, un système) puis laisse la liberté de rajouter autant de lignes qu'on le souhaite. On peut aussi séparer le contenu en deux colonnes (uniquement, pas plus de deux), pour bien tout aligner, ce qui revient à construire un tableau. On écrit ligne par ligne de gauche à droite, en commençant par la première. Le

séparateur des colonnes est `&` tandis que celui des lignes est `\\`. Voici un exemple : la commande

```
\begin{dcases}
\dot{\theta}^2 + \frac{g}{l}\cos\theta &= T \\
\ddot{\theta} + \frac{g}{l}\sin\theta &= 0 \\
\end{dcases}
```

renvoie

$$\begin{cases} \dot{\theta}^2 + \frac{g}{l}\cos\theta &= T \\ \ddot{\theta} + \frac{g}{l}\sin\theta &= 0 \end{cases}.$$

On peut également se servir de cela pour définir des fonctions :

```
R_{\omega,\theta} : \begin{dcases}
\textbf{C} &\to \textbf{C} \\
z &\mapsto \omega + e^{i \theta} (z - \omega)
\end{dcases}
```

renvoie

$$R_{\omega,\theta} : \begin{cases} \mathbf{C} &\to \mathbf{C} \\ z &\mapsto \omega + e^{i\theta}(z-\omega) \end{cases}.$$

2.5 Les fonctions usuelles et opérateurs

La convention en mathématiques est d'écrire les fonctions usuelles et opérateurs en lettres droites, on écrira donc ln pour désigner la fonction logarithme népérien, et non pas ln. Pour cela, il faut rajouter un \ (backslash) devant le nom de l'opérateur ou de la fonction. Le tableau ci-dessous compare les résultats des différentes commandes.

Commande	Résultat
`\ln`	ln
`ln`	ln
`\pgcd`	pgcd
`pgcd`	$pgcd$

Attention, le backslash n'a pas pour effet de mettre votre texte en lettres droites, en effet la commande `\chat` vous renverra une erreur. Plus généralement en LaTeX le backslash indique au compilateur que ce qui suit est une macro, une commande connue dont le résultat visuel est connu. Vous trouverez facilement sur Internet des listes non-exhaustives de toutes les fonctions et opérateurs pris en charge de base par LaTeX.

Et si LaTeX ne connaît pas mon opérateur ? Pas de panique, on peut déclarer localement un opérateur à l'aide de la commande `\mathrm{texte}`. texte

apparaîtra alors en lettres droites. On peut également utiliser la commande `\operatorname{texte}` qui fait presque la même chose à ceci près que la ponctuation est traitée par `\mathrm` comme des caractères mathématiques, tandis que l'autre la traite comme des caractères textes. Comme un exemple vaut mieux qu'un long discours, voici la différence en images :

Commande	Résultat
`\mathrm{semi-norme}`	$\mathrm{semi-norme}$
`\operatorname{semi-norme}`	$\operatorname{semi-norme}$
`\mathrm{Ker} f`	$\mathrm{Ker} f$
`\operatorname{Ker} f`	$\operatorname{Ker} f$

Pour `\mathrm`, dans le premier exemple , le tiret est en police mathématique, et dans le second, l'expace entre Ker et f est plus petit.

Un autre avantage de `\operatorname` est qu'il est possible de spécifier si l'on peut écrire quelque chose dessous et dessous en rajoutant une petite étoile, `\operatorname*`. Et c'est repartit pour un exemple.

Commande	Résultat
`\operatorname{\sim}_{n \to +\infty}`	$\operatorname{\sim}_{n \to +\infty}$
`\operatorname*{\sim}_{n \to +\infty}`	$\operatorname*{\sim}_{n \to +\infty}$

2.6 Écrire des matrices

Une matrice, c'est un tableau de nombres entre parenthèses. De manière analogue à l'environnement `cases`, on a l'environnement `pmatrix` qui au lieu de placer une accolade à gauche de l'environnement l'entoure de parenthèses. La syntaxe est la même, à ceci près que l'on peut évidemment créer plus de deux colonnes.

```
\begin{pmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta \\
\end{pmatrix}
```

renvoie

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

On peut bien sûr faire beaucoup plus compliqué,

```
\begin{pmatrix}
0 & \hdots & \hdots & 0 & a_0 \\
1 & 0 & & \vdots & a_1 \\
```

```
0 & 1 & \ddots & \vdots & \vdots \\
\vdots & \ddots & \ddots & 0 & a_{p-2} \\
0 \hdots & 0 & 1 & a_{p-1} \\
\end{pmatrix}
```

renvoie

$$\begin{pmatrix} 0 & \dots & \dots & 0 & a_0 \\ 1 & 0 & & \vdots & a_1 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & a_{p-2} \\ 0 & \dots & 0 & 1 & a_{p-1} \end{pmatrix}.$$

Il est possible d'écrire aussi des déterminants avec l'environnement `vmatrix`.

3 Conclusion

Avec tout ceci, vous pouvez désormais écrire à peu près tout ce que vous voulez avec le MathBot. Il est important en science de pouvoir bien se faire comprendre, et ce bot est l'outil idéal pour ça, n'hésitez pas à l'utiliser pour bien clarifier les choses. Vous possédez aussi maintenant les bases du langage LaTeX, n'hésitez pas à poursuivre votre apprentissage, vous n'en serez pas déçu et l'investissement en temps en vaut la chandelle si vous comptez continuer à travailler dans les sciences. N'hésitez pas à pratiquer régulièrement, et écrire en LaTeX deviendra rapidement très naturel !