

# L'extension frenchmath\*

Antoine Missier  
antoine.missier@ac-toulouse.fr

25 février 2024

## 1 Introduction

Cette extension, inspirée à l'origine de `mafr` de Christian Obrecht [10], permet le respect des règles typographiques des mathématiques françaises, en particulier l'obtention automatique des majuscules en romain (lettres droites) plutôt qu'en italique (voir [1] et [2]), et elle gère correctement les espacements pour les virgules, point-virgules et crochets. L'extension fournit en outre diverses macros francisées<sup>1</sup>. Depuis la version 2.0, `frenchmath` propose en outre deux options permettant de composer les minuscules grecques du mode mathématique en forme droite.

D'autres solutions existent pour composer les majuscules mathématiques en romain, par exemple avec X<sub>Y</sub>LaTeX ou LuaLaTeX et l'extension `unicode-math` [11]. Celle-ci peut être utilisée avec `frenchmath` à condition d'être chargée *avant*. Pour pdfLaTeX nous avons les extensions `fourier` de Michel Bovani [12] (avec la famille des fontes Adobe Utopia) ou encore `mathdesign` de Paul Pichaureau [13] (avec les polices Adobe Utopia, URW Garamond ou Bitstream Charter). Mais `frenchmath` fournit une solution générique s'adaptant à n'importe quelle police de caractères.

Certaines préconisations, telles que composer en lettre droite et non en italique le symbole différentiel, les constantes mathématiques *i* et *e* [2], sont des règles internationales [5] [6] [7]. Elles ne sont donc pas implémentées dans `frenchmath`<sup>2</sup>, tout comme diverses commandes que l'on trouve dans `mafr` et qui ne sont pas spécifiques aux mathématiques françaises : c'est le cas de `\vect`<sup>3</sup>, des ensembles de nombres `\R`, `\N`... (pour `\mathbf{R}`, `\mathbf{N}`...) ainsi que celles relatives à la réalisation de feuilles d'exercices.

Mentionnons par ailleurs l'extension `tdsfrmath` de Yvon Henel [16] qui fournit également beaucoup de commandes francisées ou encore `tblvar` [17] qui permet de réaliser de jolis tableaux de variations, spécificité des mathématiques françaises.

---

\*Ce document correspond à `frenchmath` v2.10, dernière modification le 25/02/2024.

1. Contrairement à `mafr` nous avons choisi de ne pas conserver le même nom de commande pour substituer des symboles français aux symboles anglais.
2. Nous proposons pour cela l'extension `mismath` [14] qui fournit diverses macros pour les mathématiques internationales.
3. Pour de jolis vecteurs on dispose de l'extension `esvect` [15] d'Eddie Saudrais.

## 2 Utilisation

### 2.1 Majuscules mathématiques

Dans les mathématiques françaises, pour l'alphabet latin, « les lettres majuscules sont toujours composées en romain » (A, B, C...) et non en italique (cf. [1] p.107, voir aussi [2]). En utilisant X<sub>Y</sub>LaTeX ou LuaLaTeX, avec l'option `math-style=french` de l'extension `unicode-math`, cette convention est assez comode à mettre en œuvre, y compris avec les polices TrueType ou OpenType ; par contre, avec LaTeX ou pdfLaTeX, elle est peu respectée et les extensions précitées ne fonctionnent qu'avec des polices particulières. Par défaut `frenchmath` compose automatiquement les majuscules mathématiques latines en romain, quelle que soit la police utilisée. Par exemple `\[ P(X)=\sum_{i=0}^n a_i X^i \]` donne avec `frenchmath`

$$P(X) = \sum_{i=0}^n a_i X^i.$$

[capsit] L'option `capsit` de `frenchmath` permet de désactiver la composition des majuscules du mode mathématique en romain pour conserver la composition par défaut (en italique) : `\usepackage[capsit]{frenchmath}`.

Que l'option soit activée ou pas, il est toujours possible de changer ponctuellement l'aspect d'une lettre particulière, avec les macros LaTeX `\mathrm` et `\mathit`. D'autre part l'extension `mismath` [14] fournit deux bascules puissantes `\MathUp` et `\MathIt` qui agissent de manière globale (ou locale dans un environnement) et permettent à tout moment de changer la « famille » d'une lettre particulière ; une commande générique `\apply` permet y compris d'appliquer ces macros sur une liste. Ainsi `\apply\MathIt{F,G,X}` remettra en italique les lettres *F*, *G* et *X*.

### 2.2 Virgule, point-virgule et crochets

virgule Dans le mode mathématique de LaTeX, la virgule est toujours, par défaut, un symbole de ponctuation et sera donc suivie d'une espace. Ceci est légitime dans une liste ou un intervalle : `\$[a,b]\$` donne  $[a, b]$ . Mais, en français, la virgule sert aussi de séparateur décimal pour les nombres et ne doit, dans ce cas, pas être suivie d'espace ; or `\$12,5\$` donne 12,5 au lieu de 12,5. L'extension `babel`, avec l'option `french` [18], fournit deux bascules : `\DecimalMathComma` et `\StandardMathComma`, qui permettent d'adapter le comportement de la virgule du mode mathématique.

Deux autres extensions bien commodes permettent néanmoins de se passer de ces bascules<sup>4</sup>. En mode mathématique :

- avec `icomma` (intelligent comma) de Walter Schmidt [19], la virgule se comporte comme un caractère de ponctuation si elle est suivie d'une espace, sinon c'est un caractère ordinaire ;
- avec `nccomma` de Alexander I. Rozhenko [20], la virgule se comporte comme un caractère ordinaire si elle est suivie d'un chiffre (sans espace), sinon c'est un caractère de ponctuation.

---

4. Dans ce cas il ne faut pas utiliser les bascules, au risque de rendre ces extensions inopérantes.

Cette deuxième approche paraît meilleure, néanmoins `nccomma` ne fonctionne pas avec `babel-french` utilisé conjointement avec l’extension `numprint` et son option `autolanguage`<sup>5</sup>. En outre `nccomma` ne fonctionne pas non plus avec l’extension `unicode-math` (et bugue à l’appel `\setmathfont`). Dans son article *Intelligent commas* [21], Claudio Beccari propose une autre solution, voisine de `nccomma`, mais qui produit le même type d’incompatibilités. Le code a donc été revu et simplifié afin de régler proprement ces incompatibilités. Comme bien des pays utilisent la virgule comme séparateur décimal, il fait l’objet d’une extension séparée, `decimalcomma` [22], qui est chargée par `frenchmath` (depuis la version 2.7). Mais, si on utilise `unicode-math`, il est alors impératif de charger `frenchmath` *après*.<sup>6</sup>

Lorsque l’on utilise l’extension `pstricks-add` de `PSTricks` pour tracer des axes de coordonnées, l’appel `\psset{comma=true}` permet d’avoir les graduations avec une virgule au lieu du point décimal. Ce réglage est effectué par défaut ici.

**point-virgule** Le symbole « ; » a été redéfini pour le mode mathématique car l’espace précédant le point-virgule est incorrecte en français `$x \in [0,25 ; 3,75]$` donne  $x \in [0,25; 3,75]$  sans `frenchmath` et  $x \in [0,25 ; 3,75]$  avec `frenchmath`; le comportement de « ; » devient identique à celui de « : ».

**crochets** Alors que les Anglais utilisent généralement les parenthèses pour les intervalles ouverts  $(0, +\infty)$ , l’usage en français est d’utiliser les crochets  $]0, +\infty[$ . Mais comme cela n’est pas prévu par `LATEX`, les espaces seront souvent incorrectes. Nous avons redéfini les crochets dans l’extension `ibackets` [23] qui est chargée par `frenchmath`, sauf si l’on active l’option `noibrackets`<sup>7</sup>. Le code `$x \in ]-\pi, 0[ \cup ]2\pi, 3\pi[` produira

$$x \in ]-\pi, 0[ \cup ]2\pi, 3\pi[ \quad \text{avec } \code{ibackets},$$

au lieu de  $x \in ]-\pi, 0[ \cup ]2\pi, 3\pi[$  sans `ibackets`.

Avec `ibackets`, un crochet devient un caractère ordinaire, sauf s’il est immédiatement suivi par un signe + ou - (sans espace), auquel cas c’est un délimiteur ouvrant. Si la borne de gauche possède un signe - (ou +), *il ne faut pas laisser d’espace entre le premier crochet et le signe* : par exemple `$x \in ]-\infty, 0]$` produit  $x \in ]-\infty, 0]$  au lieu de  $x \in ]-\infty, 0]$ . Mais au contraire lorsque l’on veut faire de l’algèbre sur les intervalles, *il faut laisser une espace entre le second crochet et l’opération* + ou -, par exemple, `$$[a,b] + [c,d]$$` produit  $[a, b] + [c, d]$  mais `$$[a,b]+ [c,d]$$` produit  $[a, b]+[c, d]$ .

En cas de comportement problématique, par exemple si une coupure de ligne se produit entre les deux crochets d’un intervalle, il est toujours possible de transformer alors ces crochets en délimiteurs avec `\left` et `\right`.

5. L’option `autolanguage` de `numprint` utilisée conjointement avec l’option `french` de `babel` garantit un espacement correct entre les groupes de trois chiffres dans les grands nombres, qui doit être une espace insécable et non dilatable [1], légèrement plus grande que l’espace que l’on obtient sans cette option.

6. L’extension `icomma` présente la même limitation et doit être chargée après `unicode-math`.

7. D’autres solutions existent, par exemple avec l’extension `interval` ou encore avec la macro `\DeclarePairedDelimiter` de `mathtools` [24], mais utilisée avec des crochets, cette dernière est incompatible avec `ibackets`, d’où la possibilité de désactiver `ibackets`.

## 2.3 Quelques macros et alias utiles

- `\curs` Les lettres cursives ( $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D} \dots$ ), sont composées avec `\mathscr`, ou son alias `\curs`, et sont différentes de celles obtenues avec `\mathcal`<sup>8</sup> ( $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D} \dots$ ). En principe `frenchmath` charge l’extension `mathrsfs` qui fournit ces lettres cursives, sauf si la commande `\mathscr` est définie par ailleurs, en particulier si on utilise l’extension `mathdesign` [13] ou l’option `scr` de `mathalpha` [25]<sup>9</sup>.
- `\infeg` Les relations  $\leq$  et  $\geq$  s’obtiennent avec les commandes `\infeg` et `\supeg` et diffèrent des versions anglaises de `\leq` ( $\leq$ ) et `\geq` ( $\geq$ ). Ce sont des alias de `\leqslant` et `\geqslant` de l’extension `amssymb`, chargée par `frenchmath`.
- `\vide` Le symbole de l’ensemble vide  $\emptyset$  s’obtient avec `\vide` (alias de la commande `\varnothing` de l’extension `amssymb`) ; il diffère de celui obtenu avec `\emptyset`, particulièrement laid dans la fonte classique Latin Modern :  $\emptyset$ .
- `\paral` La commande `\paral` fournit la *relation*<sup>10</sup> du parallélisme :  $\mathcal{D} \parallel \mathcal{D}'$ , plutôt que sa version anglaise `\parallel` :  $\mathcal{D} \parallel \mathcal{D}'$ .
- `\ssi` La commande `\ssi` produit le texte « si, et seulement si, ».
- `\cmod` Le modulo se compose normalement entre parenthèses, avec `\pmod`, mais on rencontre aussi, en français, le modulo entre crochets, ce que permet la commande `\cmod` en respectant le bon espacement propre au modulo :  $53 \equiv 5 \pmod{12}$ .

## 2.4 Identifiants de « fonctions » classiques

- `\pgcd` En arithmétique, nous avons les classiques `\pgcd` et `\ppcm`, qui diffèrent de leur version anglaise `\gcd` et `\lcm`<sup>11</sup>.
- `\card` Pour le cardinal d’un ensemble, nous proposons `\card`, cité dans [1] et [3], ou `\Card`, qui est aussi d’usage courant (cf. Wikipedia).
- `\Ker` `\Hom`  $\text{\LaTeX}$  fournit les macros `\ker` et `\hom`, alors que l’usage français est souvent de commencer ces noms par une majuscule pour obtenir `Ker`<sup>12</sup> et `Hom`.
- `\rg` Le rang d’une application linéaire ou d’une matrice (`rg`) s’obtient avec la commande `\rg` et l’espace vectoriel engendré par une famille de vecteurs avec `\Vect`.
- `\ch` En principe, les fonctions hyperboliques s’écrivent en français avec les macros `\sh`  $\text{\LaTeX}$  standard `\cosh`, `\sinh`, `\tanh`. Néanmoins les écritures  $\text{ch } x$ ,  $\text{sh } x$  et  $\text{th } x$ , qui sont la norme avec les langues d’Europe de l’Est (voir [31]), sont aussi utilisées en français [1]. On les obtient avec les commandes `\ch`, `\sh` et `\th`<sup>13</sup>.

8. L’extension `calrsfs` fournit les mêmes cursives mais en redéfinissant la commande `\mathcal`.

9. L’extension `mathalpha` de Michael Sharpe permet d’accéder à différentes variantes élégantes de lettres calligraphiques, par exemple avec les options `scr=boondox`, `scr=rsfso` ou `scr=kp`.

10. Pour noter que deux objets sont perpendiculaires, on utilise `\perp` :  $\mathcal{D} \perp \mathcal{D}'$ , défini comme une *relation* mathématique plutôt que `\bot` défini comme un *symbole* (les espacements diffèrent).

11. Cette dernière n’est pas implémentée en standard dans  $\text{\LaTeX}$  (mais dans `mismath` [14]).

12. La commande `\Im` existe déjà pour la partie imaginaire des nombres complexes et produit  $\Im$  ; elle est redéfinie en `Im` par l’extension `mismath` et peut aussi être utilisée pour l’image.

13. La commande `\th` existe déjà, pour le mode texte uniquement, et produit  $\text{th}$  ; elle a été redéfinie, uniquement pour le mode mathématique.

`\cosec` La fonction cosécante (inverse du sinus) s'obtient avec la macro `\csc`, mais en français, on utilise aussi `\cosec` [1] et `\cosech` pour la cosécante hyperbolique <sup>14</sup>.

## 2.5 Bases et repères

`\Oij` Les repères classiques du plan ou de l'espace seront composés avec des hauteurs de flèches homogénéisées : `\Oijk` compose  $(O, \vec{i}, \vec{j})$ , `\Oijk` compose  $(O, \vec{i}, \vec{j}, \vec{k})$  et `\Ouv` compose  $(O, \vec{u}, \vec{v})$  (utilisé dans le plan complexe). On peut écrire ces commandes en mode texte, sans les délimiteurs du mode mathématique.

`\Oij*` Les versions étoilées utilisent le point-virgule et non la virgule comme séparateur après le point O, comme mentionné dans [1]. On obtient  $(O ; \vec{i}, \vec{j})$ ,  $(O ; \vec{i}, \vec{j}, \vec{k})$ , `\Ouv*`  $(O ; \vec{u}, \vec{v})$ .

`\ij` Enfin les macros `\ij` <sup>15</sup> et `\ijk` composent les bases du plan et de l'espace, `\ijk`  $(\vec{i}, \vec{j})$  et  $(\vec{i}, \vec{j}, \vec{k})$ , en homogénéisant la hauteur des flèches.

Signalons que, pour l'extension `mathptmx` (basée sur la police de texte Times), `\jmath` n'est pas disponible, mais `frenchmath` contourne ce problème en chargeant alors `dotlessj` de David Carlisle [26], ce qui permet aux macros ci-dessus de fonctionner normalement.

## 2.6 Lettres grecques

La norme concernant l'usage des lettres grecques en italique ou en forme droite pour les mathématiques françaises ne semble pas aussi claire que pour les lettres romaines et il y a parfois divergence sur ce point. Beaucoup recommandent l'usage des lettres grecques minuscules en forme droite ([12] [13] [9]), mais certains auteurs préconisent l'italique, comme pour toutes les variables mathématiques ([3]). Le lexique des règles typographiques en usage à l'Imprimerie Nationale [1] les compose en forme droite et relativement grasses (p.108) sans préciser s'il s'agit vraiment d'une règle s'appliquant aux variables, au même titre que celle énoncée pour l'alphabet latin.

Pour les physiciens (et chimistes) l'affaire est plus claire puisque les quantités doivent toujours être écrites en italique et les unités ou les constantes en romain (forme droite), conformément à la norme ISO [5] [6] [7]. Ainsi la constante  $\pi \approx 3,14$  ne s'écrit pas de la même manière qu'une variable  $\pi$ .

Dans la section « How to get upright small Greek letters », la documentation de `isomath` de Günter Milde [8] expose différentes méthodes pour obtenir les lettres grecques en forme droite. Par exemple les extensions `mathdesign` [13], `fourier` [12] ou `kpfonts` [27] disposent d'options permettant l'écriture automatique des lettres grecques minuscules en forme droite (ou des majuscules en italique). Citons éga-

<sup>14</sup>. La fonction sécante est définie en standard par L<sup>A</sup>T<sub>E</sub>X avec `\sec` et la sécante hyperbolique `\sech` est définie par `mismath` [14].

<sup>15</sup>. Notons que la macro `\ij` existait déjà (ligature entre i et j pour le hollandais) et a été redéfinie.

lement `newpxmath`, `newtxmath`<sup>16</sup> et `libertinust1math` de Michael Sharpe, `pxgreek`, `txgreek`<sup>17</sup> et `libgreek` de Jean-François Burnol, qui donnent de beaux résultats pour une utilisation avec respectivement les polices Palatino, Times et Libertinus.

Jean-François Burnol a également développé l’extension `lgrmath` [28] qui permet d’utiliser, en mode mathématique, les différentes fontes de lettres grecques accessibles par L<sup>A</sup>T<sub>E</sub>X avec l’encodage LGR. La documentation de l’extension indique comment consulter et utiliser les fontes accessibles sur votre distribution.

Enfin, comme pour les majuscules, l’extension `unicode-math` réalise cette tâche automatiquement avec l’option `math-style=french`, mais nécessite une compilation avec X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X ou LuaL<sup>A</sup>T<sub>E</sub>X.

L’extension `frenchmath` fournit les options décrites ci-dessous qui permettent d’obtenir les lettres grecques en forme droite. Ces options peuvent être utilisées y compris avec `unicode-math`, mais `frenchmath` doit toujours être chargé *après* `unicode-math`.

[`lgrmath`] En activant l’option `lgrmath`, `frenchmath` charge l’extension du même nom avec son option `style=french` et la fonte `fcm` (de l’extension `cm-lgc`)<sup>18</sup>. Celle-ci se marie particulièrement bien avec la police usuelle Latin Modern. Les commandes `\alpha`, `\beta`, etc. produisent alors les lettres en forme droite  $\alpha$ ,  $\beta$ , ...,  $\pi$ , etc., tandis que `\alphait`, `\betait`, etc. produisent des formes italiques  $a$ ,  $\beta$ , ...,  $\pi$ , etc. Ces dernières sont peu à notre goût, mais on conserve les lettres d’origine avec `\italpha`, `\itbeta` :  $\alpha$ ,  $\beta$ , ...,  $\pi$ , etc.

`\SaveGreekItalics` On peut aussi choisir d’autres fontes en chargeant l’extension `lgrmath` indépendamment de `frenchmath` (voir par exemple les options `font=Alegreya-LF` ou `font=Cochineal-LF` de `lgrmath`). Dans ce cas, il ne faut pas activer l’option `lgrmath` de `frenchmath` et, si l’on veut conserver les lettres italiques d’origine, il faut précéder le chargement de `lgrmath` de la commande `\SaveGreekItalics` ; avec `unicode-math`, il faut en outre placer cette commande dans un `\AtBeginDocument` avec `\usepackage[style=french,font=...]{lgrmath}` et *après* le chargement de `unicode-math`.

[`upgreek`] Avec la même philosophie, `frenchmath` fournit aussi l’option `upgreek` basée sur l’extension `upgreek` de Walter Schmidt [29] qui donne accès à d’autres fontes de lettres grecques minuscules en forme droite avec `\upalpha`, `\upbeta`, etc. L’extension `upgreek` sera chargée avec son option `Symbol`<sup>19</sup> utilisant la police Adobe Symbol et qui produit des lettres grecques assez grasses :  $\alpha$ ,  $\beta$ , ...,  $\pi$ , etc.

Si l’on veut utiliser l’extension `upgreek` avec l’une des deux autres options disponibles, `Euler` (qui produit  $\alpha$ ,  $\beta$ , ...,  $\pi$ , etc.) ou `Symbolsmallscale`, il faut char-

16. L’extension `newtxmath` doit être chargée après `frenchmath` qui utilise `amssymb` car la compilation produit sinon un message d’erreur ; ou sinon on peut ajouter `\let\Bbbk\relax` avant de charger `frenchmath`.

17. Si on utilise `amsmath` (ou `mismath`), `pxgreek` ou `txgreek` doit être chargée *après* `amsmath` (ou `mismath`), pour éviter une erreur de compilation due à la redéfinition des commandes `\iint`, `\iiint`, `\idotsint`.

18. Évidemment il faut que `cm-lgc` soit installée sur votre distribution sans quoi la fonte de substitution LGR/`cmr`/`m`/`n` sera utilisée.

19. L’option `Symbol` de `upgreek` se marie bien avec une police comme Times par exemple.

ger l’extension `upgreek` avec l’option souhaitée indépendamment de `frenchmath`. Par contre, pour que les commandes `\alpha`, `\beta`... correspondent aux formes droites (comme alias de `\upalpha`, `\upbeta`...), il faut ici conserver l’option `upgreek` de `frenchmath`. Comme pour `lgrmath`, si l’on veut que les formes italiques d’origine restent toujours disponibles avec les commandes `\italpha`, `\itbeta`, etc., il faut précéder le chargement de l’extension `upgreek` de la commande `\SaveGreekItalics`; et si l’on utilise `unicode-math`, il faut en outre ajouter la commande `\upgreekUndefined` avant le `\usepackage[...]{upgreek}` et les trois dans un `\AtBeginDocument` placé *entre* le chargement de `unicode-math` et celui de `frenchmath`<sup>20</sup>.

`\upgreekUndefined`

[`Upgreek`]

Avec  $\LaTeX$ , les lettres grecques majuscules sont automatiquement composées en forme droite et l’option `upgreek` ne concerne que les minuscules. Cependant l’extension `upgreek` propose aussi `\Upgamma`, ..., `\Upomega` :  $\Gamma$ , ...,  $\Omega$ . Afin de conserver majuscules et minuscules dans le même style, `frenchmath` fournit l’option `Upgreek` qui redéfinit les majuscules `\Gamma`, ..., `\Omega` pour correspondre à ces variantes. Par contre l’on n’a alors plus accès aux caractères d’origine :  $\Gamma$ , ...,  $\Omega$ . L’option `Upgreek` reprend aussi les minuscules grecques de l’option `upgreek`, qu’il est donc inutile d’invoquer simultanément.

Signalons aussi l’extension `textalpha` de Günter Milde [30] qui donne accès aux lettres en forme droite  $\alpha$ ,  $\beta$ , ...,  $\pi$ , etc., mais en mode texte avec `\textalpha`, `\textbeta`, etc. Ces glyphes, semblables à celles de la fonte `lmr` accessible avec `lgrmath`, se marient également bien avec la police Latin Modern, par contre le thêta produit,  $\vartheta$ , n’est pas vraiment celui qui est d’usage en mathématiques.

Mentionnons pour finir ce commentaire de Walter Schmidt [29] que le  $\mu$  utilisé pour le préfixe des unités physiques,  $\mu$ , doit se composer avec `\textmu`<sup>21</sup>, disponible en mode texte dans beaucoup de fontes (ou avec `textcomp`); il diffère du  $\mu$  droit mathématique.

### 3 Le code

```

1 \newif\ifcapsit
2 \DeclareOption{capsit}{\capsittrue}
3 \newif\iflgrmath
4 \DeclareOption{lgrmath}{\lgrmathtrue}
5 \newif\ifupgreek
6 \DeclareOption{upgreek}{\upgreektrue}
7 \newif\ifUpgreek
8 \DeclareOption{Upgreek}{\Upgreektrue\upgreektrue}
9 \newif\ifnoibrackets
10 \DeclareOption{noibrackets}{\noibracketstrue}
11 \ProcessOptions \relax

```

<sup>20</sup>. Tout cela est bien contraignant, mais c’est lié au fonctionnement capricieux de `unicode-math` qui génère de nombreuses incompatibilités. Ceci dit, on n’a en principe besoin de nos options de lettres grecques droites que si l’on n’est pas satisfait de celles fournies par `unicode-math`.

<sup>21</sup>. L’extension `textalpha` fournit à la place `\textmicro` (depuis 2020) car elle redéfinit `\textmu`.

```

12
13 \@ifpackageloaded{unicode-math}{
14   \@ifpackageloaded{mathdesign}{
15     \PackageInfo{frenchmath}{Package mathdesign
16       is loaded, \MessageBreak
17       I don't load mathrsfs and amssymb}
18   }{
19     %\let\circledS\relax % utilisé uniquement par mathdesign
20     \let\eth\relax % 3 incompatibilités unicode-math / amssymb
21     \let\digamma\relax
22     \let\backepsilon\relax
23     \RequirePackage{amssymb}
24     \PackageInfo{frenchmath}{Package unicode-math
25       is loaded, \MessageBreak
26       I don't load mathrsfs}
27   }
28   \let\vide\varnothing % \varnothing sera écrasé par unicode-math
29 }{
30 \AtBeginDocument{
31   \@ifpackageloaded{mathdesign}{
32     \PackageInfo{frenchmath}{Package mathdesign
33       is loaded, \MessageBreak
34       I don't load mathrsfs and amssymb}
35   }{
36     \RequirePackage{amssymb} % \leqslant, \geqslant, \varnothing
37     \@ifundefined{mathscr}{\RequirePackage{mathrsfs}}{
38       \PackageInfo{frenchmath}{Command \string\mathscr\space
39         already defined, \MessageBreak
40         I don't load mathrsfs}
41     }
42   }
43 }
44 }
45 \RequirePackage{amsopn} % fournit \DeclareMathOperator
46 \@ifpackageloaded{mathptmx}{\RequirePackage{dotlessj}}{
47 \RequirePackage{xspace} % utile pour les commandes \ssi, \Oij
48 \ifnoibackets\else\RequirePackage{ibackets}\fi % intelligent brackets
49 \RequirePackage{decimalcomma} % depuis frenchmath 2.7
50

```

`\DeclareMathUp`

Sauf si l'option `capsit` est activée, on redéfinit toutes les lettres majuscules du mode mathématique grâce à la commande `\DeclareMathUp`. Contrairement aux bascules `\MathUp` et `\MathIt` de l'extension `mismath`, `\DeclareMathUp` ne fonctionne que dans le préambule, mais son code est bien plus simple et suffit à nos besoins ici. Pour balayer toutes les lettres majuscules de l'alphabet, les boucles usuelles `\@for` ou `\foreach` ne fonctionnent pas et produisent un message d'erreur « ! Improper alphabetic constant ». Ceci est dû au fait qu'une *séquence de contrôle* comme par exemple `\@letter` ne peut être utilisée comme argument de la macro `\DeclareMathUp` et ne sera pas gérée de manière identique à un simple ca-

`\apply` ractère. Mais la macro `\apply` ci-dessous va faire le travail<sup>22</sup>. `\AtBeginDocument` est nécessaire pour que ces définitions soient prises en compte avec la classe `beamer` par exemple, ou avec `unicode-math`.

```

51 \newcommand*\DeclareMathUp[1]{
52   \DeclareMathSymbol{#1}{\mathalpha}{operators}{' #1}}
53
54 \def\apply#1#2{\apply@#1#2, \apply@,}
55 \def\apply@#1#2,{\ifx\apply@#2\empty
56   \else #1{#2}\afterfi@{\apply@#1}\fi}
57 \def\afterfi@#1#2\fi{\fi#1}
58
59 \ifcapsit\else\AtBeginDocument{
60   \apply\DeclareMathUp{A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}
61 }
62 \fi
63
64 \AtBeginDocument{\ifpackageloaded{pstricks-add}{\psset{comma=true}}{}}
65 \DeclareMathSymbol{;}{\mathbin}{operators}'73} % \mathpunct à l'origine
66

```

Passons aux alias et identifiants de fonctions classiques.

```

67 \newcommand\curs{\mathscr}
68 \newcommand\infeg{\leqslant}
69 \newcommand\supeg{\geqslant}
70 \ifundefined{vide}{\newcommand\vide{\varnothing}}{}}

```

La définition de `\paral` remplace, depuis la version 2.2, l'ancienne définition plus simple `\mathrel{/\!\!/}`, mais qui donnait des barres trop serrées avec `mathas-text + times` ou avec `libertinustlmath`. Merci à Jean-François Burnol de me l'avoir fait remarquer et pour ses suggestions dans la mise au point d'une macro plus efficace.

```

71 \newcommand\paral{\mathrel{\ooalign{$\mkern-1.75mu/\mkern1.75mu$\cr%
72   $\mkern1.75mu/\mkern-1.75mu$}}}
73 \newcommand\ssi{si, et seulement si, \xspace}
74 \newcommand*\cmod[1]{\quad[#1]}
75
76 \DeclareMathOperator{\pgcd}{pgcd}
77 \DeclareMathOperator{\ppcm}{ppcm}
78 \DeclareMathOperator{\card}{card}
79 \DeclareMathOperator{\Card}{Card}
80 \DeclareMathOperator{\Ker}{Ker}
81 \DeclareMathOperator{\Hom}{Hom}
82 \DeclareMathOperator{\rg}{rg}
83 \DeclareMathOperator{\Vect}{Vect}
84 \DeclareMathOperator{\ch}{ch}
85 \DeclareMathOperator{\sh}{sh}

```

---

22. Cette puissante macro, postée le 26/02/2021 sous le pseudonyme de *wipet*, se trouve sur le forum de discussion [iterate190.rssing.com](https://iterate190.rssing.com), en réponse à « TeX How to iterate over a comma separated list? ». Que son auteur, Petr Olšák, soit remercié. Cette macro a été définie de manière identique dans `mismath`, mais cela ne génère pas d'incompatibilité.

```

86 \AtBeginDocument{\let\oldth\th %\th existe déjà (mode texte)
87   \renewcommand\th{\TextOrMath{\oldth}{\operatorname{th}}}}
88 \DeclareMathOperator{\cosec}{cosec}
89 \DeclareMathOperator{\cosech}{cosech}
90
Présentons les commandes pour les bases et repères, qui peuvent être utilisées en
mode texte grâce à \ensuremath (et \xspace qui garantit le bon espacement).
91 \newcommand\@Oij{%
92   \ensuremath{\left(0, \vec{\imath}, \vec{\jmath}\right)\xspace}
93 \newcommand\@Oijj{%
94   \ensuremath{\left(0 ; \vec{\imath}, \vec{\jmath}\right)\xspace}
95 \newcommand\Oij{\ifstar{\@Oij}{\@Oijj}}
96
97 \newcommand\@Oijk{%
98   \ensuremath{%
99     \left(0, \vec{\vphantom{t}\imath}, \vec{\vphantom{t}\jmath},
100    \vec{\vphantom{t}\smash{k}}\right)%
101   \xspace}
102 \newcommand\@Oijkj{%
103   \ensuremath{%
104     \left(0 ; \vec{\vphantom{t}\imath}, \vec{\vphantom{t}\jmath},
105     \vec{\vphantom{t}\smash{k}}\right)%
106   \xspace}
107 \newcommand\Oijk{\ifstar{\@Oijk}{\@Oijkj}}
108
109 \newcommand\@Ouv{%
110   \ensuremath{\left(0, \vec{u}, \vec{v}\right)\xspace}
111 \newcommand\@Ouvj{%
112   \ensuremath{\left(0 ; \vec{u}, \vec{v}\right)\xspace}
113 \newcommand\Ouv{\ifstar{\@Ouv}{\@Ouvj}}
114
115 \AtBeginDocument{
116   \renewcommand\ij{%
117     \ensuremath{\left(\vec{\imath}, \vec{\jmath}\right)\xspace}}
118 \newcommand\ijk{%
119   \ensuremath{%
120     \left(\vec{\vphantom{t}\imath}, \vec{\vphantom{t}\jmath},
121     \vec{\vphantom{t}\smash{k}}\right)%
122   \xspace}
123

```

Ce n'est qu'à la fin du préambule, donc avec `\AtBeginDocument`, que l'on examine les options `lgrmath` ou `upgreek`, pour que cela fonctionne avec `unicode-math` et pour laisser à l'utilisateur la possibilité de charger `upgreek` après `frenchmath` et éviter un conflit d'option. On sauvegarde les lettres grecques d'origine dans les macros `\italpha`, ..., `\itomega`, grâce à la commande `\SavedGreekItalics`. La macro `\upgreekUndefined` ne sert que si l'on veut utiliser `upgreek` avec `unicode-math` : elle permet de « vider » la définition de `\upalpha`, `\upbeta`..., car sinon leur redéfinition par `upgreek` est impossible.

```

124 \providecommand\SaveGreekItalics{% pourrait être définie ailleurs
125     \let\italpha\alpha
126     \let\itbeta\beta
127     \let\itgamma\gamma
128     \let\itdelta\delta
129     \let\itepsilon\epsilon
130     \let\itzeta\zeta
131     \let\iteta\eta
132     \let\ittheta\theta
133     \let\itiota\iota
134     \let\itkappa\kappa
135     \let\itlambda\lambda
136     \let\itmu\mu
137     \let\itnu\nu
138     \let\itxi\xi
139     \let\itpi\pi
140     \let\itrho\rho
141     \let\itsigma\sigma
142     \let\ittau\tau
143     \let\itupsilon\upsilon
144     \let\itphi\phi
145     \let\itchi\chi
146     \let\itpsi\psi
147     \let\itomega\omega
148     \let\itvarepsilon\varepsilon
149     \let\itvartheta\vartheta
150     \let\itvarpi\varpi
151     \let\itvarsigma\varsigma
152     \let\itvarphi\varphi
153 }
154
155 \providecommand\upgreekUndefined{
156     \let\upalpha\@undefined
157     \let\upbeta\@undefined
158     \let\upgamma\@undefined
159     \let\updelta\@undefined
160     \let\upepsilon\@undefined
161     \let\upzeta\@undefined
162     \let\upeta\@undefined
163     \let\uptheta\@undefined
164     \let\upiota\@undefined
165     \let\upkappa\@undefined
166     \let\uplambda\@undefined
167     \let\upmu\@undefined
168     \let\upnu\@undefined
169     \let\upxi\@undefined
170     \let\uppi\@undefined
171     \let\uprho\@undefined
172     \let\upsigma\@undefined
173     \let\uptau\@undefined

```

```

174 \let\upupsilon\@undefined
175 \let\upphi\@undefined
176 \let\upchi\@undefined
177 \let\uppsi\@undefined
178 \let\upomega\@undefined
179 \let\upvarepsilon\@undefined
180 \let\upvartheta\@undefined
181 \let\upvarpi\@undefined
182 \let\upvarrho\@undefined
183 \let\upvarsigma\@undefined
184 \let\upvarphi\@undefined
185 }
186
187 \AtBeginDocument{
188   \iflgrmath
189     \SaveGreekItalics
190     \RequirePackage[font=fcm,style=french]{lgrmath}
191   \fi
192   \ifupgreek
193     \@ifpackageloaded{upgreek}{}{
194       \@ifpackageloaded{unicode-math}{\upgreekUndefined}{}
195       \SaveGreekItalics
196       \RequirePackage[Symbol]{upgreek}
197     }
198     \renewcommand\alpha{\upalpha}
199     \renewcommand\beta{\upbeta}
200     \renewcommand\gamma{\upgamma}
201     \renewcommand\delta{\updelta}
202     \renewcommand\epsilon{\upepsilon}
203     \renewcommand\zeta{\upzeta}
204     \renewcommand\eta{\upeta}
205     \renewcommand\theta{\uptheta}
206     \renewcommand\iota{\upiota}
207     \renewcommand\kappa{\upkappa}
208     \renewcommand\lambda{\uplambda}
209     \renewcommand\mu{\upmu}
210     \renewcommand\nu{\upnu}
211     \renewcommand\xi{\upxi}
212     \renewcommand\pi{\uppi}
213     \renewcommand\rho{\uprho}
214     \renewcommand\sigma{\upsigma}
215     \renewcommand\tau{\uptau}
216     \renewcommand\upsilon{\upupsilon}
217     \renewcommand\phi{\upphi}
218     \renewcommand\chi{\upchi}
219     \renewcommand\psi{\uppsi}
220     \renewcommand\omega{\upomega}
221     \renewcommand\varepsilon{\upvarepsilon}
222     \renewcommand\vartheta{\upvartheta}
223     \renewcommand\varpi{\upvarpi}

```

```

224 \renewcommand\varrho{\upvarrho}
225 \renewcommand\varsigma{\upvarsigma}
226 \renewcommand\varphi{\upvarphi}
227 \fi
228 \ifUpgreek
229 % unicode-math utilise \upGamma, \upDelta...
230 \renewcommand\Gamma{\Uppgamma}
231 \renewcommand\Delta{\Uppdelta}
232 \renewcommand\Theta{\Upptheta}
233 \renewcommand\Lambda{\Upplambda}
234 \renewcommand\Xi{\Uppxi}
235 \renewcommand\Pi{\Uppi}
236 \renewcommand\Sigma{\Uppsigma}
237 \renewcommand\Upsilon{\Uppupsilon}
238 \renewcommand\Phi{\Uppphi}
239 \renewcommand\Psi{\Uppsi}
240 \renewcommand\Omega{\Uppomega}
241 \fi
242 }

```

## Références

- [1] *Lexique des règles typographiques en usage à l'Imprimerie Nationale*, édition du 26/08/2002.
- [2] *Composition des textes scientifiques*, Inspection Générale de mathématiques (IGEN-DESCO), 06/12/2001.  
[http://mslp.ac-dijon.fr/IMG/pdf/typo\\_txt\\_sci.pdf](http://mslp.ac-dijon.fr/IMG/pdf/typo_txt_sci.pdf)  
<https://euler.ac-versailles.fr/IMG/pdf/typo2.pdf>
- [3] *Règles françaises de typographie mathématique*, Alexandre André, 02/09/2015. [http://sgalex.free.fr/typo-maths\\_fr.pdf](http://sgalex.free.fr/typo-maths_fr.pdf)
- [4] *Le petit typographe rationnel*, Eddie Saudrais, 20/03/2000.  
<https://www.gutenberg-asso.fr/IMG/pdf/saudrais-typo.pdf>
- [5] *Typesetting mathematics for science and technology according to ISO 31/XI*, Claudio Beccari, TUGboat Volume 18 (1997), N° 1.  
<http://www.tug.org/TUGboat/tb18-1/tb54becc.pdf>
- [6] *Typefaces for Symbols in Scientific Manuscripts*.  
<https://www.physics.nist.gov/cuu/pdf/typefaces.pdf>
- [7] *On the Use of Italic and up Fonts for Symbols in Scientific Text*, I.M. Mills and W.V. Metanomski, ICTNS (Interdivisional Committee on Terminology, Nomenclature and Symbols), dec 1999.  
[https://old.iupac.org/standing/idcns/italic-roman\\_dec99.pdf](https://old.iupac.org/standing/idcns/italic-roman_dec99.pdf)
- [8] *isomath – Mathematical style for science and technology*, Günter Milde, CTAN, v0.6.1 2012/09/04.
- [9] *PM-ISOMath – The Poor Man ISO math bundle*, Claudio Beccari, CTAN, v1.2.00 2021/08/04.

- [10] *La distribution mafr*, Christian Obrecht, CTAN, v1.0 17/09/2006.
- [11] *Experimental Unicode mathematical typesetting : The unicode-math package*, Will Robertson, Philipp Stephani, Joseph Wright, Khaled Hosny, and others, CTAN, v0.8r 13/08/2023.
- [12] *Fourier-GUTenberg*, Michel Bovani, CTAN, v1.3 30/01/2005.
- [13] *The mathdesign package*, Paul Pichaureau, CTAN, v2.31 29/08/2013.
- [14] *mismath – Miscellaneous mathematical macros*, Antoine Missier, CTAN, v2.10 20/02/2024.
- [15] *esvect – Typesetting vectors with beautiful arrow with L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>*, Eddie Sautrais, CTAN, v1.3 11/07/2013.
- [16] *L’extension tdsfrmath*, Yvon Henel, CTAN, v1.3 22/06/2009.
- [17] *L’extension tablvar*, Antoine Missier, CTAN, v2.0 23/12/2023.
- [18] *A Babel language definition file for French*, extension L<sup>A</sup>T<sub>E</sub>X babel-french de Daniel Flipo, CTAN, v3.5c 14/09/2018.
- [19] *The icomma package for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>*. Walter Schmidt, CTAN, v2.0 10/03/2002.
- [20] *The nccomma package*. Alexander I. Rozhenko, CTAN, v1.0 10/02/2005.
- [21] *Intelligent commas*. Claudio Beccari, The PracT<sub>E</sub>X Journal, 2011, No.1. <https://tug.org/pracjourn/2011-1/beccari/Intcomma.pdf>
- [22] *The decimalcomma package*. Antoine Missier, CTAN, v1.4 30/12/2023.
- [23] *Intelligent brackets – The ibrackets package*, Antoine Missier, CTAN, v1.2 26/07/2023.
- [24] *The mathtools package*, Morten Høgholm, Lars Madsen and the L<sup>A</sup>T<sub>E</sub>X3 project, CTAN, v1.29 29/06/2022.
- [25] *The mathalpha, AKA mathalfa package*, Michael Sharpe, CTAN, v1.143 18/11/2021.
- [26] *dotlessj*, David Carlisle, CTAN, v0.03 09/12/1998.
- [27] *Kp-Fonts – The Johannes Kepler project*, Christophe Caignaert, CTAN, v3.34 20/09/2022.
- [28] *The lgrmath package*, Jean-François B., CTAN, v1.0 16/11/2022.
- [29] *The upgreek package for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>*, Walter Schmidt, CTAN, v2.0 12/03/2003.
- [30] *The textalpha package* (partie de l’extension greek-fontenc), Günter Milde, CTAN, v2.1 14/06/2022.
- [31] *L<sup>A</sup>T<sub>E</sub>X Companion*, Frank Mittelbach, Michel Goossens, 2<sup>e</sup> édition, Pearson Education France, 2005.