

Miscellaneous mathematical macros

The `mismath` package*

Antoine Missier
`antoine.missier@ac-toulouse.fr`

August 16, 2026

Contents

1	Introduction	1
2	Usage	2
2.1	Mathematical constants	2
2.2	Vectors	4
2.3	Standard operator names	5
2.4	A few useful aliases and small macros	8
2.5	Improved spacing in mathematical formulas	10
2.6	Environments for systems of equations and small matrices	12
2.7	Displaymath in double columns	14
2.8	Greek letters for mathematical constants and operators	14
2.9	Summary of package options and comments on single-letter macros	17
3	Implementation	18
4	Change history, references, index	38

1 Introduction

According to the International Standards ISO 31-0:1992 to ISO 31-13:1992 (superseceded by ISO 80000-2:2009), mathematical *constants* e , i , π should be typeset in roman (i.e. upright) and not in italics like variables (see [1] [2] [3]). This package provides some tools to achieve this automatically.

Even though it is recommended to typeset vector names in bold italic style [2] [3], they are often represented with arrows, especially in school documents or in physics. To draw nice arrows above vectors, we use the `esvect` package by Eddie Sautrais [5]. Additionally, we provide a few more macros related to vectors with arrows, particularly to improve the typesetting of the norm: $\|\overrightarrow{AB}\|$ instead of the L^AT_EX version $\|\overline{AB}\|$, which is not vertically adjusted, or worse $\|\overrightarrow{AB}\|$ (when using `\left...\right`).

*This document corresponds to `mismath` v3.3, dated 2026/08/16. Thanks to François Bastouil for initial help in English translation and Romain Noël for his interest and relevant suggestions.

The package also offers other macros for:

- some standard operator names, including Greek letters,
- several commands with useful aliases, including tensors in sans serif bold italic shape (ISO recommendation [1] [2]),
- improved spacings in mathematical formulas,
- systems of equations and small matrices,
- displayed equations in two columns for lengthy calculations involving short expressions.

To avoid compatibility issues, most of our macros will only be defined if there isn't already a command with the same name in the packages loaded before `mismath`. If a macro is already defined, a warning message will be displayed and the `mismath` definition will be ignored. If you wish to keep either the `mismath` definition or the existing one, you can use `\let<command>\relax` either before or after loading `mismath`.

[...] (*opt.*) The `mismath` package loads the `mleftright`¹ package by Heiko Oberdiek [6] and also `mathtools`² by Morten Høgholm and Lars Madsen [7] which in turn loads the `amsmath` package [8]. If you wish to use `amsmath` or `mathtools` with specific options, *you can include these options as options of `mismath`*, or you can load `amsmath` or `mathtools` with the desired options before loading `mismath`. When using `unicode-math` [9], `mismath` should be loaded before `unicode-math`, just like `amsmath`.

An ISO recommendation, although rarely respected, is to typeset uppercase Greek letters in italics, as for other variables [3]. This is automatically achieved, for some particular fonts, with packages such as `fixmath` by Walter Schmidt [11], `isomath` by Günter Milde [12] or `pm-isomath` by Claudio Beccari [13] and optionally with many others (such as `mathpazo` or `mathptmx` with the option `slantedGreek`). When running through `LuaATEX` or `XYLATEX` you can also get this result with the option `math-style=ISO` provided by the `unicode-math` package. We also have the `mathgreek` package [14] which offers a wide range of fonts and different settings with Greek letters. However this feature is not implemented here due to a conflicting rule in France, where all capital letters in mathematics are required to be typeset in upright shape³. The user is free to choose loading one of these packages or not.

2 Usage

2.1 Mathematical constants

`\mathup` As with standard mathematical functions, *predefined* mathematical constants should be typeset in upright shape (typically in roman family), but this practice is not sufficiently respected, probably because it's a bit tedious. A first way is

¹The `mleftright` package defines variants `\mleft` and `\mright` of `\left` and `\right`.

²The `mathtools` package offers numerous helpful macros and improvements of the `amsmath` package.

³The `frenchmath` package [37] ensures to follow the recommended French rules.

to use the `\mathup` macro, which is generally preferable to `\mathrm`⁴, for setting any group of letters in roman (upright font). For instance you can use `\mathup{e}` to get the Euler’s number.

`\e` To avoid cluttering a document that contains many occurrences of Euler’s number or the imaginary unit with `\mathup{e}` or `\mathup{i}`, the package provides `\i` the `\e` command for Euler’s number and `\i` or `\j` for imaginary numbers. Note that `\i` and `\j` already exist in L^AT_EX. In LR (left-to-right) mode, they produce ‘i, j’ without the dot, allowing you to place accents on them. However, in mathematical mode, they produce the warning “LaTeX Warning: Command `\i` invalid in math mode on input line *<line>*”. With the new definition provided by the package, `\i` and `\j` will be redefined specifically for mathematical mode without altering the behavior in text mode.

`\MathUp` Typing a lot of backslashes for constants like *e*, *i*, or *j* in a document with numerous formulas using them can become tiresome and many authors will not do so. That’s why the package proposes another convenient solution with the macro `\MathUp{<letter>}`. For example, when `\MathUp{e}` is called, any subsequent occurrence of *e* will automatically be set in roman, without the need to type `\e` explicitly. The effect of this macro is either global or local, depending on whether it is used outside or inside an environment or a pair of braces. You can also call this macro in the preamble. This powerful command allows you to bring a document into line with these typographic conventions effortlessly and without changing anything in your mathematical formulas. In fact, `\MathUp` can be applied to any valid single Latin letter, making it useful in a variety of situations⁵.

`\MathIt` When there are other occurrences of *e*, *i* or *j* as variables, you can still obtain italicized *e*, *i* or *j* using L^AT_EX commands `\mathit` and `\mathnormal`, which are convenient for occasional use. However, you can also use the inverse switch `\MathIt{<letter>}`, which has a global effect when used outside environments or braces, or a local effect when used inside them. Just as for `\MathUp`, `\MathIt` can be applied to any single Latin letter.

`\MathNumbers` These macros enable you to set roman or italic (the default) typesetting for multiple letters in a single command. For instance, `\MathNumbers{e,i}` is equivalent to `\MathUp{e}\MathUp{i}`. This macro only affects the letters *e*, *i*, or *j*; it has no effect on other characters. On the other hand, `\MathNormal` accepts any comma-separated list of arguments. This means you can apply the normal italic math mode typesetting to various letters at once using `\MathNormal`.

`\enumber` These three commands, used until version 2.2 but only functioning within the `\inumber` preamble, now serve as aliases for the commands `\MathUp{e}`, `\MathUp{i}` or `\jnumber` `\MathUp{j}`, and can therefore be used anywhere in the document or preamble. They also have an inverse switch, `\MathIt`.

The constant π should also be typeset in upright shape (see [1], [2], [3]), which is different from italicized π . However, this recommendation is even less commonly followed compared to the one concerning *e* and *i* [1]. The macros that implement

⁴The `\mathup` macro is based on `\operatorfont`, which comes from the `amsopn` package, automatically loaded by `amsmath`. In `beamer`, the default math font is sans serif, but `\mathrm` produces a font with serifs, which might not match the overall style of the presentation. Hence, using `\mathup` is indeed a better choice in `beamer` presentations to ensure that mathematical constants are typeset in upright shape and consistent with the default sans serif math font.

⁵Another use of it to write probabilities will be presented in section 2.3.

this behavior and allow the automatic replacement of π by π when typing `\pi` will be presented in the section 2.8 dedicated to Greek letters for mathematical constants and operators.

2.2 Vectors

`\vect` By default, the `\vect` command typesets vectors with arrows (thanks to the `esvect` package by Eddie Soudrais⁶) which are more elegant than those produced by the standard L^AT_EX command `\overrightarrow`, particularly with the default Computer Modern or Latin Modern font: $\overrightarrow{AB}$. The `esvect` package has an option (a single letter between `a` and `h`) to define the desired type of arrow (see [5]). In `mismath`, `esvect` is loaded with the option `b`: `\vect{AB}` gives $\overrightarrow{AB}$. If you wish to use a different type of arrow, you must call `esvect` with the appropriate option *before* loading `mismath`. For example, using `\usepackage[d]{esvect}` will provide the same arrows that are used by default in [5].

`\boldvect` The `\vect` macro allows vector names to be typeset using bold italic font, as recommended by ISO [2], instead of using arrows. To achieve this, call the `\boldvect` command, which modifies the behavior of `\vect` locally or globally, depending on its placement in the document (inside or outside a group or an environment):

```
[ \boldvect \vect{v}
= \lambda \vect{e}_x + \mu \vect{e}_y ]
```

$$\mathbf{v} = \lambda \mathbf{e}_x + \mu \mathbf{e}_y.$$

`\boldvectcommand` By default, `\boldvect` uses the `\boldsymbol` command⁷ from the `amsbsy` package, which is automatically loaded by `amsmath`. However, you may prefer other packages that produce bold italic fonts, such as `fixmath` with the `\mathbfbold` command, `isomath` with `\mathbfbf` or `bm` with the `\bm` command; `unicode-math` provides the `\symbfit` command. To use an alternative command instead of `\boldsymbol` in `mismath`, redefine `\boldvectcommand`, for instance if `fixmath` is loaded:

```
\renewcommand\boldvectcommand{\mathbfbold}.
```

According to ISO rules, symbols that represent matrices are also in bold italic. Therefore you can also use `\vect` with `\boldvect` for matrices, or create another alias.

`\arrowvect` At any time, you can revert to the default behavior using the inverse switch `\arrowvect`. These switches may be used anywhere, whether inside mathematical mode or within an environment (with a local effect) or outside (with a global effect).

`\hvect` When vectors with arrows are typeset side by side, the arrows can be set up slightly higher using `\hvect` (which inserts a vertical phantom of ‘*t*’) to prevent inelegant effects. For example, writing

- $\overrightarrow{AB} = \vec{u} + \overrightarrow{AC}$, obtained with `\hvect{u}`, looks better than $\overrightarrow{AB} = \vec{u} + \overrightarrow{AC}$;
- $\vec{F} = m \vec{a}$, obtained with `\hvect{a}`, looks better than $\vec{F} = m \vec{a}$.

⁶`esvect` provides the `\vv` macro used by `\vect`.

⁷`\mathbfbf` produces upright bold font, even when used in combination with `\mathit`.

This adjustment ensures a nicer appearance when vectors with arrows are combined in an equation⁸. The `\boldvect` and `\arrowvect` switches affect `\hvect` in the same way as `\vect`.

`\hvec` In a similar way, `\hvec` raises the little arrow produced by the L^AT_EX command `\vec`, to the height of the letter ‘t’ (but `\boldvect` has no effect on `\vec` nor `\hvec`):

- $\vec{a} \cdot \vec{b} = 0$, obtained with `\hvec{a}`, looks better than $\vec{a} \cdot \vec{b} = 0$.
- $P = \vec{F} \cdot \vec{v}$, obtained with `\hvec{v}`, looks better than $P = \vec{F} \cdot \vec{v}$.

`\norm` The norm of a vector is conventionally represented using the delimiters `\lVert` and `\rVert` (or `\|` unless a plus (+) or minus (-) sign follows the opening delimiter) or `\left\lVert` and `\right\lVert` for adaptive delimiters. Unfortunately, these delimiters are always vertically centered, relative to the mathematical center line, whereas vectors with arrows are asymmetric objects. The code `\norm{\vec{h}}` raises the double bar to produce $\|\vec{h}\|$ instead of $\|\vec{h}\|$ or $\|\vec{h}\|$. Note that the height of the bars doesn’t adjust to content. However, it does adjust to the surrounding math style (main text, subscripts, or superscripts), e.g. $X^{\|\vec{h}\|}$.

Since version 3.3, this macro can also handle symmetric or small sized arguments, e.g. `\norm{a}` gives $\|a\|$. The starred version `\norm*` forces the raising of the bars if necessary.

`\innerprod` The inner product is typeset with the two-argument macro `\innerprod`, which is a shortcut for `\left\langle` and `\right\rangle`, e.g. `\innerprod{u}{v}` yields $\langle u, v \rangle$.

2.3 Standard operator names

`\di` The *differential* operator should be typeset in roman, not in italics, to distinguish it from variables (as mentioned in [1] [2] [3] [38]). To achieve this, we provide the `\di` command. Several authors use `\ud` (meaning upright ‘d’) for this purpose, generally an alias for `\mathrm{d}`. It is not equivalent because the `\di` macro leaves a *thin space before* the ‘d’ letter, like any other operator (which `\mathrm{d}` does not do), but *no space after*. Take a look at the following examples:

$$\begin{aligned} & \int \int xy \, \mathrm{d}x \, \mathrm{d}y, \\ & m \frac{\mathrm{d}^2 x}{\mathrm{d}t^2} + h \frac{\mathrm{d}x}{\mathrm{d}t} + kx = 0 \end{aligned}$$

The command `\di` can also represent the *distance*: $d(x, y)$, hence its name.

`\opDelta` Two other ‘difference’ operators allow for expressing variations (Δ) or small variations (δ). They are obtained using the `\opDelta` and `\opdelta` commands.

$$\Delta h \approx f'(x_0) \Delta x, \quad \frac{\delta T}{T} = \frac{1}{2} \frac{\delta l}{l}, \quad \delta A = x \delta y + y \delta x + \delta x \delta y.$$

Like `\di`, these operators use the same spacing and are typeset using upright Greek letters from the selected font. A thin space is inserted before the operator,

⁸For a fine tuning you can also use `\vstrut` or `\cstrut` from the `spacingtricks` package [28].

but none after it. The commands for choosing and managing Greek letters, in particular how to set the group of special lowercase upright Greek letters, are explained in section 2.8.

`\opGamma` We also provide classic functions that are represented by Greek letters: the
`\opzeta` Gamma function Γ obtained with `\opGamma`, the Riemann zeta function ζ obtained
`\opsigma` with `\opzeta`, the Dirac function δ , or Kronecker delta, which can also be obtained
`\opPhi` with `\opdelta`, the standard deviation σ obtained with `\opsigma`, the cumulative
distribution function of the standard normal distribution Φ , obtained with `\opPhi`.
For any of them, as for the previous difference operators, they have to be typeset
in upright shape and with a thin spacing before and no space after⁹. But they are
operators not just letters. The following examples show that:

$$\Gamma(z) \Gamma\left(z + \frac{1}{m}\right) \Gamma\left(z + \frac{2}{m}\right) \cdots \Gamma\left(z + \frac{m-1}{m}\right) = (2\pi)^{\frac{m-1}{2}} m^{\frac{1}{2}-mz} \Gamma(mz),$$

$$\Gamma(s) \zeta(s) = \int_0^{+\infty} \frac{t^{s-1}}{e^t - 1} dt, \quad \int_{-\infty}^{+\infty} f(t) \delta(t - T) = f(T),$$

$$\sigma(aX + b) = |a| \sigma(X), \quad \Phi^{(n)}(x_0) = -\left(x_0 \Phi^{(n-1)}(x_0) + (n-2) \Phi^{(n-2)}(x_0)\right).$$

However, these macros are not available by default, since an upright Greek font must first be defined, see section 2.8.

`\P` When representing a probability¹⁰ or an expectation, the proper use is to
`\E` typeset the capital letters P, E in roman, just like any standard function identifier.
This can be achieved with the `\P` and `\E` commands.

`\Par` The `\P` command already existed to refer to the ‘end of paragraph’ symbol ($\P$),
in text mode. It has been redefined, but only for the math mode, in such a way
that `\P` can still be used in text mode, or we can use an alias: `\Par`.

`\V` Variance is generally denoted by `var` or `Var` (see the table below), but some
authors prefer to use `V`, which can be produced using `\V`.

`\apply` As for e, i or j, you can use `\MathUp{P}`, `\MathUp{E}` or `\MathUp{V}` to avoid
typing many `\P`, `\E` or `\V`, and you get the inverse toggle with `\MathIt` for any in-
dividual roman letter, or you can use `\apply\MathUp{<mylist>}`¹¹ or `\MathNormal`
on a comma-separated list. Nevertheless you must be aware that spacing is better
managed with the macros `\P`, `\E` or `\V` than with just roman letters that will not
be considered as operators. For instance

$$P(A \cap B) = P_B(A)P(B), \quad E(XY) - E(X)E(Y), \quad V(aX) = a^2V(X),$$

obtained with `\apply\MathUp{P,E,V}`, produces incorrect spacing before the let-
ters P, E and V when they follow another letter or a closing delimiter. This doesn’t
occur with `\P`, `\E` or `\V`, which are operators and not just letters:

$$P(A \cap B) = P_B(A) P(B), \quad E(XY) - E(X) E(Y), \quad V(aX) = a^2 V(X).$$

`\probastyle` Some authors use double-struck font shape to represent probability, expec-

⁹Unlike classic operators composed of multiple letters who can take a thin space after their name, e.g. $\sin x$, but, if needed, σX without space seems better.

¹⁰ $\LaTeX$ provides also `\Pr` which gives Pr.

¹¹Thanks to the `\apply` macro, developed by Petr Olšák.

tation and variance: $\mathbb{P}, \mathbb{E}, \mathbb{V}$. The `\probastyle` macro sets the appearance of `\P`, `\E` and `\V`. For instance `\renewcommand\probastyle{\mathbb}`¹² switches to double-struck letters. It can be even used inside a formula to change dynamically the signification and behavior of `\P`, `\E` and `\V`, e.g.

$$\begin{array}{l} \text{\code{\P_X(A) \renewcommand\probastyle{\mathbb}} \\ \text{\code{\sum_{x \in A} P(X=x) \}}} \end{array} \quad P_X(A) = \sum_{x \in A} \mathbb{P}(X = x).$$

If you have to do many such changes, you can define switch aliases `\probbb` or `\probrm`. The `\mathbb` command is provided by `amsfonts` package (which is *not* loaded by `mismath`), but also by other complete math font packages such as `mathdesign`, `kpfonts`, `fourier`, `unicode-math...`, or you get the convenient `mathalpha` [10] by Michael Sharpe, which lets you choose among many available fonts for the commands `\mathbb`¹³, but also `\mathcal`, `\mathfrak`, `\mathscr...`

The following standard operator names are defined in `mismath`.

<code>\adj</code>	adj	<code>\End</code>	End	<code>\rank</code>	rank
<code>\Aut</code>	Aut	<code>\erf</code>	erf	<code>\Res</code>	Res
<code>\codim</code>	codim	<code>\grad</code>	$\overrightarrow{\text{grad}}$	<code>\rot</code>	$\overrightarrow{\text{rot}}$
<code>\codom</code>	codom	<code>\Hv</code>	H	<code>\sgn</code>	sgn
<code>\coker</code>	coker	<code>\id</code>	id	<code>\sinc</code>	sinc
<code>\Conv</code>	Conv	<code>\Id</code>	Id	<code>\spa</code>	span
<code>\Cov</code>	Cov	<code>\im</code>	im	<code>\supp</code>	supp
<code>\cov</code>	$\overrightarrow{\text{cov}}$	<code>\lb</code>	lb	<code>\tr</code>	tr
<code>\curl</code>	curl	<code>\lcm</code>	lcm	<code>\Var</code>	Var
<code>\divg</code>	div	<code>\ord</code>	ord	<code>\var</code>	var
<code>\dom</code>	dom	<code>\ran</code>	ran	<code>\Zu</code>	Z

By default, operators returning vectors, `\grad` and `\curl` (or its synonym `\rot` more commonly used in Europe), are written with an arrow on the top. When `\boldvect` is activated, they are typeset in bold style: **grad**, **curl**, **rot**. For the variance, the covariance and the identity function, two notations are proposed, with or without a first capital letter, because both are very common. Note that `\div` already exists ($\div$) and `\span` is a $\text{\TeX}$ primitive; they have not been re-defined. Therefore the provided macros are called `\divg` (divergence) and `\spa` (span of a set of vectors). Furthermore `\Z` is used to denote the set of integers and `\H` the set of quaternions (see 2.4), We therefore use `\Zu` to denote the center of a group: $Z(G)$ (from German Zentrum), and `\Hv` for the Heaviside step function.

The `mismath` package also provides some (inverse) trigonometric or hyperbolic functions, that are missing in $\text{\LaTeX}$.

<code>\arccot</code>	arccot	<code>\arsinh</code>	arsinh	<code>\arcoth</code>	arcoth
<code>\sech</code>	sech	<code>\arcosh</code>	arcosh	<code>\arsech</code>	arsech
<code>\csch</code>	csch	<code>\artanh</code>	artanh	<code>\arcsch</code>	arcsch

`\FT` The Fourier and Laplace transforms are typeset with `\FT` and `\LT` respectively,
`\LT` which are defined as operators, but typeset using `\mathcal`.

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \mathcal{F}\{g\}, \quad \mathcal{L}\{af + bg\} = a \mathcal{L}\{f\} + b \mathcal{L}\{g\}.$$

`nofunction (opt.)` Some may find that the definition of all these operators and functions is not

¹²The effect of this redefinition is global or local when inside an environment.

¹³In the present document we have called `mathalpha` with the option `bb=fourier,cal=pmtx`.

relevant to their needs. So, the definitions of standard operators and functions in both previous tables, and also the `\FT` and `\LT` macros, can be disabled with the `nofunction` option.

`\Re` The `\Re` and `\Im` macros refer to the real and imaginary parts of a complex
`\Im` number. They have been redefined to produce ‘Re’ and ‘Im’, in place of outdated
symbols $\Re$ and $\Im$. Nevertheless, it is still possible to obtain the old symbols with
`\oldRe` and `\oldIm`.

`classicReIm (opt.)` The `classicReIm` option deactivates these redefinitions.

`\bigO` Asymptotic comparison operators (in Bachmann-Landau notation) are ob-
`\bigo` tained with `\bigO` or `\bigo` and `\lito` commands. `\bigO` uses `\mathcal` and
`\lito` the two others typeset the letter ‘O’ or ‘o’ in roman¹⁴, as for any operator.

$$n^2 + O(n \log n) \quad \text{or} \quad n^2 + O(n \log n) \quad \text{and} \quad e^x = 1 + x + \frac{x^2}{2} + o(x^2).$$

`lineargroups (opt.)` Macros for typesetting the general linear group and some of its classic sub-
groups are available when activating the package option `lineargroups`. With this
option, suggested by Romain Noël, `mismath` provides the following commands:

`\GL`, `\SL`, `\Sp`, `\O`, `\SO`, `\U`, `\SU`.

These macros typeset the group names as operators in the typographic sense
(roman letters, i.e. upright, with operator spacing).

$$\mathrm{GL}(n, \mathbf{K}), \mathrm{SL}(n, \mathbf{K}), \mathrm{Sp}(2n, \mathbf{F}), \mathrm{O}(n, \mathbf{F}), \mathrm{SO}(n), \mathrm{U}(n), \mathrm{SU}(n).$$

The `\O` command was already defined and yields $\emptyset$, but only in text mode. Thus,
`mismath` redefines it only for math mode, the original macro being saved as `\oldO`.

2.4 A few useful aliases and small macros

In the tradition of Bourbaki [4] and D. Knuth himself, proper usage requires
that standard number sets be typeset in bold roman: **R**, **C**, **Z**, **N**, **Q**, **H**, whereas
double-struck letters ($\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{H}$) are traditionally reserved for blackboard
writing [38]. Similarly, to designate a field we use **F** or **K** (Körper in German).
We obtain these symbols with the following macros:

`\R`, `\C`, `\Z`, `\N`, `\Q`, `\H`, `\F`, `\K`.

The `\H` command was already defined and yields a long Hungarian umlaut in
text mode, e.g. `\H{u}` gives ũ. Thus, `mismath` redefines it only for math mode, the
original macro being saved as `\oldH`.

`\mathset` The `\mathset` command enables you to change the behavior of all these macros
globally. By default, `\mathset` is an alias for `\mathbf`, but if you prefer double-
struck letters, you can simply use `\renewcommand\mathset{\mathbb}` (with local
effect within an environment or a pair of curly braces).

`\ds` The `\displaystyle` command is very common, therefore the `\ds` alias is pro-
vided. It not only makes typing easier, but also makes the source code more
readable.

¹⁴Donald Knuth proposed to use the omicron letter which is similar to ‘O’.

Symbols with limits behave differently for in-line formulas or for displayed equations. In the latter case, ‘limits’ are placed under or above the symbol whereas for in-line math mode, they are placed on the right, as a subscript or exponent. Compare $\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$ with

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}.$$

`\dlim` With in-line math mode, display style can be forced with `\displaystyle` or `\ds`. However, when using these commands, all the rest of the current mathematical environment will be set in display style (as shown in the previous example, where the fraction is expanded). To limit the display style effect to the affected symbol only, like the `amsmath` command `\dfrac`, we can use the following macros: `\dlim`, `\dsum`, `\dprod`, `\dcup`, `\dcap`. So

$$\text{\$}\text{\dlim_}\{x\text{\to }+\infty\}\text{\frac{1}{x}}\text{\$} \text{ yields } \lim_{x \rightarrow +\infty} \frac{1}{x}.$$

`\lbar` Long bars over expressions are produced with `\overline` or its alias `\lbar`, to get for instance $\overline{z_1 z_2}$. Similar to vectors, you can raise the bar (from the height of ‘t’) with the `\hlbar` command, to correct uneven bar heights.

$$\overline{z + z'} = \overline{z} + \overline{z'}, \text{ obtained with } \text{\hlbar}\{z\}, \text{ looks better than } \overline{z + z'} = \overline{z} + \overline{z'}.$$

`\eqdef` The `\eqdef` macro writes the equality symbol topped with ‘def’, or with ‘Δ’
`\eqdef*` for `\eqdef*` (using `\upDelta` if it exists, or `\Delta` if not):

$$\begin{aligned} \text{\[\e~}\{i\theta\} \text{\eqdef} & \qquad e^{i\theta} \stackrel{\text{def}}{=} \cos \theta + i \sin \theta, \\ \text{\[\e~}\{i\theta\} \text{\eqdef*} & \qquad e^{i\theta} \stackrel{\Delta}{=} \cos \theta + i \sin \theta. \end{aligned}$$

`\asympteq` The `\asympteq` macro is used to typeset the asymptotic equivalence. It has an optional argument, set in `\scriptscriptstyle`, whose default value is empty.

$$f(x) \text{\asympteq}[x\text{\to}+\infty] g(x) \text{ yields } f(x) \underset{x \rightarrow +\infty}{\sim} g(x).$$

`\unbr` `\unbr` is an alias for `\underbrace`¹⁵, making the source code more compact:

$$\text{\[(QAP)}^n = \text{\unbr}\{QAP\text{\mul QAP\mul \cdots\mul QAP}\}_{n\text{\text{ times}}}\text{\]} \qquad (QAP)^n = \underbrace{QAP \times QAP \times \cdots \times QAP}_{n \text{ times}}.$$

`\then` This macro produces the symbol $\implies$ surrounded by large spaces, just as the standard `\iff` macro does with $\iff$. It’s simply an alias of the `amsmath` `\implies` macro.

`\compl` Following Bourbaki [4], we can write the complement of a set A with the `\complement` command which is provided by `amssymb`, e.g. $\mathbb{C}(A \cup B) = \mathbb{C}A \cap \mathbb{C}B$. We can also use `\setminus`, which yields a backslash operator: $\mathbb{C}_E A = E \setminus A$. But the letter ‘c’ as exponent is also commonly used. To ensure a typographic distinction from variables, the `\compl` macro typesets ‘c’ in roman:

$$\text{\compl}\{(A \cup B)\} = \text{\compl}\{A\} \cap \text{\compl}\{B\} \qquad (A \cup B)^c = A^c \cap B^c.$$

¹⁵The `mathtools` package [7] provides an improved version of the `\underbrace` command.

`\integerint` The integer interval $\llbracket a, b \rrbracket$, typeset with `\integerint{a,b}`, represents the set of all integers from a to b . One of the `stmaryrd`, `unicode-math`, `fourier` or `mathabx` packages, which provide these double brackets, must be loaded to use this command. Other common notations are $[a..b]$ or just $a..b$.

`\mathbfsfit` For tensor symbols, ISO conventions [1] [2] recommend using sans serif bold italic, but there is no such math alphabet in the default L^AT_EX mathematical style. However, the `mismath` package defines this alphabet (provided that the font encoding and package you use support it) and provides the macro `\mathbfsfit` or its alias `\tensor`. By writing `\tensor{S}\otimes\tensor{T}`, you get $\mathbf{S} \otimes \mathbf{T}$.

`textshortcuts` (*opt.*) The package option `textshortcuts`, provides some text mode macros, suggested by Romain No  l, to reduce typing of some often used sentence elements in mathematics, with a starred version producing an abbreviation¹⁶.

Command	Result	Starred version	
<code>\QED</code>	Q.E.D.	<i>none</i>	
<code>\ale</code>	almost everywhere	<code>\ale*</code>	a.e.
<code>\cc</code>	complex conjugate	<code>\cc*</code>	c.c.
<code>\iff</code>	if and only if	<code>\iff*</code>	iff
<code>\st</code>	such that	<code>\st*</code>	s.t.
<code>\stp</code>	sufficient to prove	<code>\stp*</code>	s.t.p.
<code>\walog</code>	without any loss of generality	<code>\walog*</code>	w.a.l.o.g.
<code>\wma</code>	we may assume	<code>\wma*</code>	w.m.a.
<code>\wrt</code>	with respect to	<code>\wrt*</code>	w.r.t.

- When the abbreviation a.e., c.c. or w.a.l.o.g. ends a sentence, the space after the period should be a little bit longer. To get that correct space, type a period after the command: “`\ale*. Now`” yields “a.e. Now”¹⁷.
- The `\iff` macro already exists in L^AT_EX, but only in math mode. It has therefore been redefined for use in text mode also. The abbreviation “iff” may or may not be followed by a period at the end, depending on national abbreviation conventions, e.g. vs in the UK, but vs. in the USA. Add ‘`.\@`’ iff. needed.
- Three of these expressions can also begin a sentence. For this purpose we provide `\Walog`, `\Wma`, `\Wrt` and the equivalent starred versions.

2.5 Improved spacing in mathematical formulas

`\txt` The `\txt` macro, based on `\text` from the `amstext` package (loaded by `amsmath`), adds `\quad`spacing around the text. See the following example:

$$\begin{array}{l} \backslash[\backslashln x=a \backslashthen x=e^a, \backslashtxt{rather than} \\ \backslashln x=a \backslashLongrightarrow x=e^a \backslash] \\ \ln x = a \implies x = e^a, \quad \text{rather than} \quad \ln x = a \implies x = e^a. \end{array}$$

`\mul` The multiplication symbol obtained with `\times` produces the same spacing

¹⁶The usual abbreviations produced by the macros `\ie` and `\eg`, whose scope extends far beyond mathematics, are provided by other packages like `foreign` [26] or `spacingtricks` [28].

¹⁷This feature is obtained by using `\xperiod` from the `xpunctuate` package [27].

as addition or subtraction operators, whereas division obtained with `/` is closer to its operands. This makes the precedence of multiplication over addition and subtraction less visually apparent. That's why we provide the `\mul` macro, to avoid the large space surrounding `\times`:

$\lambda + \alpha \times b - \beta \times c$, obtained with `\mul`, looks better than $\lambda + \alpha \times b - \beta \times c$.

Using `\mul` before a function name works also well (since v3.1) without the need of curly braces to avoid the additional space before the operator name: `$x\mul\sin x$` yields $x \times \sin x$.

\abs The `\abs` command typesets the absolute value while properly handling spacing, unlike `|...|`. Compare $|-x|$, obtained with `\abs`, to $|-x|$ without. Using the `\lvert` and `\rvert` delimiters presents another issue when the absolute value follows a function name; for instance `\ln\lvert x\rvert` yields $\ln|x|$ instead of $\ln|x|$ (with `\ln\abs{x}`). Moreover, with `\abs`, the delimiters automatically adapt to the content¹⁸.

\card The cardinality of a set S is most commonly denoted by $|S|$, though it may also be denoted by $\#S$, $n(S)$, $\bar{S}$ or $\text{card}(S)$, the latter being commonly used in French mathematics¹⁹. We define `\card` as an alias for `\abs` but, to prevent incompatibility, our definition is delayed until the beginning of the document, after checking whether `\card` is already defined. Thus, another definition made in the preamble will be preserved; for instance, you can use `\newcommand{\card}{\#}` in the preamble before or after loading `mismath`.

\floor The `\floor` macro is a substitute for `{\left\lfloor...\right\rfloor}`; compare $q = -\lfloor -\frac{a}{b} \rfloor$ obtained with `\floor` with $q = -\lfloor -\frac{a}{b} \rfloor$.

\pow When typesetting an exponent after a closing *big* parenthesis produced by `\right)` but also `\mright)`, the exponent appears to be a little too far from the parenthesis. To address this issue, the `\pow{<expr>}{<pow>}` command is provided, which places `<expr>` in a pair of parentheses and sets the exponent `<pow>` slightly closer to the right parenthesis²⁰. Compare

$$e^a = \lim_{n \rightarrow +\infty} \left(1 + \frac{a}{n}\right)^n, \text{ obtained with \pow, with } e^a = \lim_{n \rightarrow +\infty} \left(1 + \frac{a}{n}\right)^n.$$

When using a `\left ... \right` structure, TeX sets additional surrounding space in some situations. The `mleftright` [6] package, loaded by `mismath`, offers the variants `\mleft` and `\mright` to address these spacing issues in inner formulas. It also provides the `\mleftright` macro, which redefines `\left` as `\mleft` and `\right` as `\mright`. Compare

$$\sin\left(\frac{\pi}{3}\right) \times 2 \text{ with } \sin\left(\frac{\pi}{3}\right) \times 2 \text{ obtained with } \$\sin\mleft(\frac{\pi}{3}\mright)\mul 2$.$$

\lfrac The `\lfrac` macro behaves like `\frac` but with additional spacing around the arguments, making the corresponding fraction bar slightly longer. This macro has

¹⁸You could also define `\abs` using `\DeclarePairedDelimiter` from the `mathtools` package [7], but you get the same bad result $\ln|x|$.

¹⁹The `frenchmath` package [37] defines `\card` this way, and `mismath` will not overwrite it.

²⁰Since version 3.3, the `\pow` macro works also with normal-sized arguments while keeping the exponent correctly positioned.

an optional parameter `\lfrac[⟨space⟩]{⟨num⟩}{⟨denom⟩}` to adjust the length of the fraction bar. The optional `⟨space⟩` argument must be given with *math units* (`\mu`); the default value is `7\mu` (equivalent to `\:\,`). See the following examples, the last one is obtained with `\lfrac[4\mu]{1}{\sqrt{x}}`:

$$\overline{Z} = \frac{\overline{z_1 - z_2}}{\overline{z_1 + z_2}}, \quad u(x) = \frac{\frac{1-2x}{5}}{x^2 + 1}, \quad y' + xy = \frac{1}{\sqrt{x}}.$$

ibackets (*opt.*) Open intervals are commonly represented with parenthesis, e.g. $(0, +\infty)$, but several authors use square brackets, especially in French mathematics: $]0, +\infty[$. In that case, the space around the square brackets is generally inappropriate, as in the expression $x \in]0, +\infty[$. To address this issue, you can use the **ibackets** package [29]. It can also be loaded by **mismath** using the **ibackets** package option. Thus `\x\in ]-\pi,0[ \cup ]2\pi,3\pi[` yields

yields $x \in]-\pi, 0[\cup]2\pi, 3\pi[$ with **ibackets**,
instead of $x \in -\pi, 0[\cup]2\pi, 3\pi[$ without **ibackets**.

However, when the left bound is followed by an operator sign, *you do not have to leave a space between the first bracket and the sign*, otherwise, the spaces surrounding the operator will be too large. For example if you write `\x\in ]-\infty, 0]`, it yields $x \in]-\infty, 0]$ instead of $x \in]-\infty, 0]$. Conversely, when dealing with algebraic expressions involving intervals, *you must leave a space between the second bracket and the +/- operation*. For instance `\[a,b] + [c,d]` yields $[a, b] + [c, d]$ but `\[a,b]+ [c,d]` yields $[a, b]+[c, d]$.

Note that there are other ways to proceed, for example with `\interval`, from the **interval** package [30], or with `\DeclarePairedDelimiter`²¹ from **mathtools** [7].

decimalcomma (*opt.*) In many countries, except, in particular, in English-speaking countries, the comma is used as a decimal separator for numbers. However, in the math mode of L^AT_EX, the comma is always, by default, treated as a punctuation symbol and therefore is followed by a space. This is appropriate in intervals: `\[a,b]` results in $[a, b]$, but not for numbers where the comma represents the decimal separator. For example, `\$12,5\$` is displayed as 12, 5 instead of 12.5.

Two very convenient packages allow handling the comma in math mode: **icomma** by Walter Schmidt [31] and **nccomma** by Alexander I. Rozhenko [32]. The latter package takes a more generic approach, however it poses several compatibility issues, in particular when running through LuaL^AT_EX, using **unicode-math** and calling `\setmathfont`. Therefore we propose the **decimalcomma** package [33], which is functionally equivalent to **nccomma**, but without the aforementioned incompatibility. It can be loaded by **mismath** using the **decimalcomma** package option.

2.6 Environments for systems of equations and small matrices

system (*env.*) The **system** environment, defined in the **mismath** package, is intended for typeset-

²¹You cannot use `\DeclarePairedDelimiter` with square brackets when **ibackets** is loaded.

ting systems of equations:

$$\begin{array}{l} \backslash[\backslash\begin{system} \\ \quad x=1+2t \ \backslash\backslash \ y=2-t \ \backslash\backslash \ z=-3-t \\ \backslash\end{system} \backslash] \end{array} \quad \left\{ \begin{array}{l} x = 1 + 2t \\ y = 2 - t \\ z = -3 - t \end{array} \right.$$

`\systemsep` This first example could also have been achieved using the `cases` environment from the `amsmath` package, although `cases` places mathematical expressions closer to the opening brace. The `\systemsep` length can be used to adjust the gap between the opening brace and the mathematical expressions. By default, the gap is set to `\medspace`. You can reduce this gap by redefining the command, e.g. `\renewcommand{\systemsep}{\thinspace}`. Alternatively you can increase the gap using `\thickspace`; the same spacing as in the `cases` environment being obtained with `\renewcommand{\systemsep}{}`.

By default, a system is written like an `array` environment with only one column, left aligned. However the `system` environment has an optional argument for creating systems with multiple columns, specifying their alignment using the same syntax as the `array` environment in `LATEX`. For instance, using `\begin{system}[cl]` will produce a two-column system, with the first column centered and the second column left-aligned, as shown in the following example:

$$\begin{array}{l} \backslash[\backslash\begin{system}[cl] \\ \quad y \ \& \ =\dfrac{1}{2}x-2 \ \backslash[1ex] \\ \quad (x,y) \ \& \ \neq \ (0,-2) \\ \backslash\end{system} \backslash] \end{array} \quad \left\{ \begin{array}{l} y = \frac{1}{2}x - 2 \\ (x,y) \neq (0,-2) \end{array} \right.$$

`\systemstretch` The default row spacing in a `system` environment has been slightly enlarged compared to the one used in `array` environments (using a factor of 1.2). This can be adjusted by using `\renewcommand{\systemstretch}{\langle stretch \rangle}`, where `\langle stretch \rangle` is the desired factor for the spacing. You can place this command inside the current mathematical environment for a local change, or outside for a global change. The default value is 1.2. Furthermore you can use the optional argument of the line-break command, as demonstrated above with `\backslash[1ex]`, to control the spacing between specific lines in the system.

Another example with `\begin{system}[r1@{\quad}l]`²²:

$$\left\{ \begin{array}{ll} x + 3y + 5z = 0 & R_1 \\ 2x + 2y - z = 3 & R_2 \\ 3x - y + z = 2 & R_3 \end{array} \right. \iff \left\{ \begin{array}{ll} x + 3y + 5z = 0 & R_1 \\ 4y + 11z = 3 & R_2 \leftarrow 2R_1 - R_2 \\ 5y + 7z = -1 & R_3 \leftarrow \frac{1}{2}(3R_1 - R_3) \end{array} \right.$$

We should also mention the `systeme` package [34], which provides a lighter syntax and automatic alignments for linear systems. Additionally, there is the `spalign` package [35], which offers a convenient and easy syntax for systems and matrices with visually appealing alignments.

`spmatrix (env.)` The `amsmath` package offers several environments to typeset matrices: for example, the `pmatrix` environment surrounds the matrix with parentheses, and the `smallmatrix` environment creates a smaller matrix suitable for insertion within a text line. We provide the `spmatrix` environment that combines these two features: `\vec{u}\backslash\begin{spmatrix}-1\backslash\backslash2\end{spmatrix}` yielding $\vec{u} \begin{pmatrix} -1 \\ 2 \end{pmatrix}$.

²²@{...} sets inter-column space.

The `mathtools` package enhances the `amsmath` matrix environments and also provides a small matrix environment with parentheses: `psmallmatrix`. Moreover, with the starred version `\begin{psmallmatrix*}[\langle col \rangle]`, you can choose the alignment inside the columns (`c`, `l` or `r`). However, the spacing before the opening parenthesis is unfortunately too narrow compared to the spacing inside the parentheses. To illustrate this, consider the following example: $\vec{u}\left(\begin{smallmatrix} -1 \\ 2 \end{smallmatrix}\right)$ (using `mismath`'s `spmatrix`) vs. $\vec{u}\left(\begin{smallmatrix} -1 \\ 2 \end{smallmatrix}\right)$ (using `mathtools`' `psmallmatrix` environment).

For more sophisticated matrix layouts, let us mention the excellent `nicematrix` package by François Pantigny [36].

2.7 Displaymath in double columns

`mathcols` (*env.*) The `mathcols` environment is particularly useful for long calculations whose successive steps are short enough to fit comfortably in two columns separated by a vertical rule, as shown in the following example²³. To use this environment, the `multicol` package must be loaded in the preamble. The `mathcols` environment automatically enters display math mode and uses the `aligned` environment (from `amsmath`).

$$\begin{aligned} & \frac{\gamma e^2}{4\pi\epsilon_0 r^2} - \frac{1}{\gamma} \frac{e^2}{4\pi\epsilon_0 r^2} \\ &= \frac{\gamma e^2}{4\pi\epsilon_0 r^2} \left(1 - \frac{1}{\gamma^2}\right) \\ & \quad \left(\text{with } \frac{1}{\gamma^2} = 1 - \beta^2\right) \end{aligned} \quad \left| \quad \begin{aligned} &= e(\beta c) \left(\frac{\beta}{c} \frac{\gamma e}{4\pi\epsilon_0 r^2}\right) \\ &= ev \frac{\beta}{c} \frac{\gamma e}{4\pi\epsilon_0 r^2} \\ &= ev \frac{\beta}{c} E. \end{aligned}$$

`\changeacol` The `\changeacol` macro is used to switch to the next column, and alignment within each column is achieved using the usual alignment markers `&` and `\`.

```
\def\denom{4\pi\varepsilon_0 r^2}
\begin{mathcols}
  & \frac{\gamma e^2}{\denom}
  - \frac{1}{\gamma} \frac{e^2}{\denom} \\
=& \frac{\gamma e^2}{\denom} \left(1 - \frac{1}{\gamma^2}\right) \\
& \quad (\text{with } \frac{1}{\gamma^2} = 1 - \beta^2) \\
\changeacol
& = e(\beta c) \left(\frac{\beta}{c} \frac{\gamma e}{\denom}\right) \\
& = ev \frac{\beta}{c} \frac{\gamma e}{\denom} \\
& = ev \frac{\beta}{c} E.
\end{mathcols}
```

2.8 Greek letters for mathematical constants and operators

The main macro of this section is `\specialgreeks` which will be presented after `\specialgreeksdef`.

`\specialgreeksdef` The macro `\specialgreeksdef{\font}` defines the following commands:

`\numpi, \numphi, \numgamma,`
`\opDelta, \opdelta, \opGamma, \opzeta, \opsigma, \opPhi.`

²³Unlike mathematical constants, physical constants (here ϵ_0 and c) do not have to be typeset in upright shape, but in italics, roman being reserved for physical units [1] [2] [3].

The first three are classic mathematical constants, `\numphi` represents the golden ratio and `\numgamma` the Euler-Mascheroni constant:

$$\varphi = \frac{1 + \sqrt{5}}{2}, \quad \gamma = \lim_{n \rightarrow +\infty} \left(\sum_{k=1}^n \frac{1}{k} - \ln n \right), \quad \Gamma(z) = \frac{e^{\gamma z}}{z} \prod_{n=1}^{\infty} \frac{ne^{\frac{z}{n}}}{n+z}.$$

They are typeset upright, as are the constants `e`, `i`, and `j`. The six remaining commands, `\opDelta`, `\opdelta`, etc., concerning standard operators, have been presented in section 2.3. The special Greek letters defined as operators are also available as ordinary mathematical symbols through commands prefixed with `spe` instead of `op`: `\speDelta`, `\spdelta`...

By default in L^AT_EX, Greek lowercase letters are in italic type. However many packages provide these letters in upright shape. With `\specialgreekstdef`, you don't need to install such a package. It is possible to pick only the glyphs that correspond to the aforementioned commands without altering the default Greek letters. The `(font)` argument of `\specialgreekstdef` is of the type `key=value`. The key name corresponds to a package providing the desired glyphs. The following table summarizes the available options. When a key is given without a value, a default value is associated, see the list following the table.

Option <code>lgrmath=...</code>	Result	Other options	Result
<code>Alegreya-LF</code>	$\pi\phi\gamma\Delta\delta\Gamma\zeta\sigma\Phi$	<code>fontspec=...</code>	...
<code>Cochineal-LF</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\varsigma\Phi$	<code>upgreek=Symbol</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\sigma\Phi$
<code>LibertinusSerif-LF</code>	$\pi\phi\gamma\Delta\delta\Gamma\zeta\varsigma\Phi$	<code>upgreek=Euler</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\sigma\Phi$
<code>LibertinusSans-LF</code>	$\pi\phi\gamma\Delta\delta\Gamma\zeta\varsigma\Phi$	<code>mathdesign</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\sigma\Phi$
<code>NotoSerif-LF</code>	$\pi\phi\gamma\Delta\delta\Gamma\zeta\varsigma\Phi$	<code>kpfonts</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\sigma\Phi$
<code>gentium</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\varsigma\Phi$	<code>fourier</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\sigma\Phi$
<code>lmr</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\varsigma\Phi$	<code>pxfonts</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\sigma\Phi$
<code>lmss</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\varsigma\Phi$	<code>txfonts</code>	$\pi\varphi\gamma\Delta\delta\Gamma\zeta\sigma\Phi$

- With the `lgrmath` key, we actually have numerous possibilities for values (any Greek letters math font in LGR encoding). The documentation of the `lgrmath` package [22] by Jean-François Burnol explains how to check and visualize all available LGR fonts on your distribution. We have only presented a few of them. The default value is `lmr`. It is well-suited for use with the Latin Modern roman font family²⁴ and has been used in the present document. When using `beamer`, you should invoke a sans-serif font family like `lmss`. Other interesting values are `droidserif`, `fct`, `llcmss`, `Clara-TLF`...
- Some sigma characters appearing in the previous table with the `lgrmath` option do not have the correct typeface. This is due to the commands used in the table but will not occur with normal use. For instance, the sigma produced by the current `lmr` key value is σ and not ς . Conversely, phi has sometimes an alternative shape that is noticeable and correct in the table, e.g. ϕ with `Alegreya-LF` font, but it's a commonly used glyph.

²⁴Glyphs look similar as those provided by `lgrmath=cmr`.

- With the `fontspec` key, there are also many possible values, corresponding to the TrueType or OpenType fonts installed on your system (works with Lua $\TeX$ or Xe $\TeX$). See the `mathgreek`s documentation [14] for examples. The default value is GFS Dido. You can also use the `fontspec` key option, to obtain any font that is supported by the `unicode-math \setmathfont` command, e.g. `\specialgreekssdef{fontspec=STIX Two Math}`.
- With the `upgreek` key, the default value is `Symbol`. There is a third possible value, `Symbolsmallscale`, which provides the same characters as `Symbol` but reduced in size by 10 %.
- With the `mathdesign` key, there are actually 3 possible values: `Utopia`, `Garamond` or `Charter` (the default value), but the glyphs obtained look quite similar.
- With the `kpfonts` key, we have two possible values: `normal` (default) and `light`. The option `kpfonts=light` provides slightly less bold characters.
- The last keys, `fourier` (based on Utopia), `pxfonts` (based on Palatino), `txfonts` (based on Times) are booleans whose default value is `true` (when called). The `txfonts` key yields the same glyph as `lgrmath=txr`.

If your current math font already includes upright Greek letters, you can however call this font with `\specialgreekssdef` to produce the mentioned macros, provided this font is supported in any previous key option. This is interesting in particular to get the operators.

`\mmoperator` The symbol font declared in `\specialgreekssdef` is called `mmupgr`, thus, you can pick another Greek letter in the same font with the following syntax, e.g.

`\DeclareMathSymbol{\spepsi}{\mathalpha}{mmupgr}{121}`.

The argument number depends on the encoding scheme, here 121 is psi in LGR encoding. The previous command yields the psi character in upright shape: ψ . To create the corresponding operator, declare²⁵

`\newcommand\oppsi{\mmoperator{\spepsi}}`.

`\specialgreekss` The effect of this main macro is to replace `\pi` with `\numpi`, among other choices, or likewise with any other special Greek letter presented in this section. The syntax of the macro is

`\specialgreekss[<font>]{<letters>}`.

The optional ** argument is passed to `\specialgreekssdef` and can be used only in the preamble.

The mandatory argument *<letters>* is a list of the type `key=value`. The possible keys are one of the letters

`pi, phi, gamma, Delta, delta, Gamma, zeta, sigma, Phi`.

Each key has a default value which correspond to the *characters* produced by `\specialgreekssdef`: `numpi, numphi, numgamma, speDelta, ... spePhi`, thus you can invoke only the key name. For instance

²⁵The macro `\operatorname` gives no output with LGR encoded fonts nor with `fontspec`.

`\specialgreekslgrmath=gentium}{pi,delta,Gamma}`

will replace `\pi` with `\numpi`, `\delta` with `\opdelta` and `\Gamma` with `\opGamma` in the LGR encoding gentium font.

Then, the italic type π will be replaced with an upright gentium π each time `\pi` is called. Thus `\specialgreeksl` makes your document compliant with standards without changing the source code of your mathematical formulas.

Why are the default values corresponding to operators be named as `spedelta`, `speDelta`...and not `opdelta`, `opDelta`...? If you have loaded a package that already provides upright Greek letters, you can simply specify the name of the desired character to substitute, e.g.

`\usepackage{newtxmath}\specialgreeksl{pi=piup,sigma=sigmaup}`.

In that case the command `\pi` will be defined as an alias for `\piup`, but `\sigma` will be defined as an operator based on `\sigmaup` not simply an alias, and no other special character macro will be created, except that the original `\pi` and `\sigma` will be saved as `\originalpi` and `\originalsigma`.

`\normalgreeksl` Moreover `\specialgreeksl` works as a switch, `\normalgreeksl` being the inverse switch. It takes the same keys argument than `\specialgreeksl` but any associated value is ignored. So you can use `\specialgreeksl` and `\normalgreeksl` in the document body, not only in the preamble.

`\pinumber` For compatibility reasons with versions prior to 3.3, these two macros are
`\pinormal` maintained for now, but `\specialgreeksl` and `\normalgreeksl` are extended alternatives. The behavior of `\pinumber` and `\pinormal` is as follows:

- You must first call `\pinumber` in the preamble with an optional argument which is either a command name (without the backslash) to produce an upright pi, or it is passed to `\specialgreeksldef` to define a Greek font. But unlike `\specialgreeksl`, when no optional argument is given, `lgrmath=lmr` is chosen by default. Then `\pi` is replaced by `\numpi` and saved in `\savedpi`.
- `\pinormal` is the inverse switch which brings back `\pi` to `\originalpi`.
- When used again outside the preamble, `\pinumber` acts as a switch to restore `\pi` as `\savedpi`.

2.9 Summary of package options and comments on single-letter macros

The following table summarizes the possible package options. You can add to them any option you want to pass to `amsmath` or `mathtools`. The hyperlinks (in blue) redirect to the paragraphs in the documentation where these options are described.

Option	Effect
nofunction	don't load the additional function definitions
classicReIm	preserves <code>\Re</code> and <code>\Im</code> as $\Re$ and $\Im$
lineargroups	loads macros for classic linear group names
textshortcuts	loads macros for text shortcuts and abbreviations
ibrackets	loads the <code>ibrackets</code> package
decimalcomma	loads the <code>decimalcomma</code> package
nosingleletter	don't load the single letter macros (see below)

`nosingleletter` (*opt.*)

Defining single-letter L^AT_EX macros may be regarded as a bad practice. Why is that? If every package author defined their own single-letter macros, we would soon end up with a maze of package incompatibilities, since the Latin alphabet contains only 26×2 letters. First, let us note that the L^AT_EX kernel already defines a number of such macros:

`\a, \b, \c, \d, \i, \j, \k, \l, \o, \r, \t, \u, \v, \H, \L, \O, \P,`
together with `\C` for Cyrillic typesetting, and probably a few others.

Indeed, `mismath` does define quite a few single-letter macros:

`\e, \i, \j, \P, \E, \V, \R, \C, \Z, \N, \Q, \H, \F, \K,`
and optionally `\U, \O`.

First, the commands that already exist in LaTeX (`\i, \j, \P, \C, \H` and `\O`) are defined specially for text mode, so defining them exclusively for math mode causes no conflict.

Second, `mismath` always checks whether a macro is already defined before attempting to define it. This does not completely rule out package incompatibilities, but it makes them much less likely. In particular, loading `mismath` after another package with which it would otherwise conflict is perfectly safe. Conversely, if you wish to preserve a `mismath` macro that has already been defined beforehand, simply write `\let\<macro>\relax` before loading `mismath`, as explained in the introduction.

Finally the motivation for these single-letter macros is that they keep the source code closer to standard mathematical notation, where variables, sets and many mathematical objects are traditionally represented by single letters. This makes mathematical expressions easier to write in L^AT_EX and more natural for mathematicians. Moreover, such macros are already very common in users' preambles and personal macro files. Writing `\RR` for the set of real numbers may seem less elegant, although this is ultimately a matter of taste.

Nevertheless, for users who are reluctant to use such single-letter macros, we provide the `nosingleletter` option, which disables all these single-letter macros. The corresponding double-letter commands will be defined instead:

`\ee, \ii, \jj, \PP, \EE, \VV, \RR, \CC, \ZZ, \NN, \QQ, \HH,`
`\FF, \KK,` and optionally `\UU, \OO`.

3 Implementation

We load certain packages conditionally to avoid 'option clash' errors in cases where these packages have been previously loaded with other options. The `amsmath` package is loaded by `mathtools`. The `\NewDocumentCommand` is provided by the `xparse` package, but now integrated in L^AT_EX3 kernel. The `xparse` package is no longer needed, as its functionality is now integrated into the L^AT_EX kernel since the October 2020 L^AT_EX release.

```
\newif\ifmm@ibricks % initialized to false
\DeclareOption{ibricks}{\mm@ibrickstrue}
\newif\ifmm@decimalcomma
\DeclareOption{decimalcomma}{\mm@decimalcommatrue}
\newif\ifmm@nofunction
\DeclareOption{nofunction}{\mm@nofunctiontrue}
\DeclareOption{otherReIm}{\PackageWarningNoLine{mismath}
```

```

    {Option otherReIm is obsolete}}
\newif\ifmm@classicReIm
\DeclareOption{classicReIm}{\mm@classicReImtrue}
\newif\ifmm@lineargroups
\DeclareOption{lineargroups}{\mm@lineargroupstrue}
\newif\ifmm@textshortcuts
\DeclareOption{textshortcuts}{\mm@textshortcutstrue}
\newif\ifmm@singleletter\mm@singlelettertrue
\DeclareOption{nosingleletter}{\mm@singleletterfalse}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{mathtools}}
\ProcessOptions \relax

%\@ifpackageloaded{amsmath}{\RequirePackage{amsmath}}
%\@ifpackageloaded{mathtools}{\RequirePackage{mathtools}}
%\@ifpackageloaded{esvect}{\RequirePackage[b]{esvect}}
\RequirePackage{mleftright}
\RequirePackage{ifthen}
\providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
\IfFormatAtLeastTF{2020-10-01}{\RequirePackage{xparse}}
% xparse provides \NewDocumentCommand, now in LaTeX3
\ifmm@textshortcuts
    \RequirePackage{xspace} % for textshortcuts commands
    \RequirePackage{xpunctuate} % provides \xperiod
\fi
\RequirePackage{etoolbox} % provides \AtEndPreamble and \AfterEndPreamble
\RequirePackage{xkeyval} % for \specialgreekdef options

```

The package `unicode-math` causes some compatibility issues with `ibackets` and `decimalcomma`: theses packages must be loaded *after* `unicode-math`, but `mismath` (like `amsmath`) should be loaded *before* `unicode-math`. And to complicate matters, `unicode-math` defines all its commands by `\AtBeginDocument`. Therefore we used the command `\AtEndPreamble`, from the `etoolbox` package, which makes the job (because both `ibackets` and `decimalcomma` work also in `\AtBeginDocument`).

```

\@ifpackageloaded{unicode-math}{
    \PackageWarningNoLine{mismath}{The package unicode-math\MessageBreak
        should be loaded after mismath}
}{}
\newif\ifmm@multicol
\newif\ifmm@unicodemath
\AtEndPreamble{
    \ifmm@decimalcomma\RequirePackage{decimalcomma}\fi
    \ifmm@ibackets\RequirePackage{ibackets}\fi
    \@ifpackageloaded{multicol}{\mm@multicoltrue}{}
    \@ifpackageloaded{unicode-math}{\mm@unicodemathtrue}{}
    \@ifpackageloaded{lua-unicode-math}{\mm@unicodemathtrue}{}
}

```

`\bslash` The `\bslash` macro originates from Frank Mittelbach's `doc.sty` package. It can be employed as an alternative to `\textbackslash`, especially in situations where `\textbackslash` does not work correctly, such as inside warning messages.

```

{\catcode'\|=\z@ \catcode'\=12 \gdef\bslash{\}} % \bslash command

```

The next three internal macros are generic tools for conditionally defining macros and issuing a warning if the macro already exists..

```
\mm@warning
\newcommand\mm@warning[1]{
  \PackageWarningNoLine{mismath}{Command \bslash #1 already exists
    \MessageBreak and will not be redefined}
}

\mm@macro
\newcommand\mm@macro[2]{
  \@ifundefined{#1}{
    \expandafter\def\csname #1\endcsname{#2}
  }{\mm@warning{#1}}
}

\mm@operator
\NewDocumentCommand\mm@operator{O{#3}mm}{%
  \@ifundefined{#1}{
    \DeclareMathOperator{#2}{#3}
  }{\mm@warning{#1}}
}
```

`\mathup` To produce the correct upright shape font when working with the `beamer` package, you don't have to use `\mathrm` but rather `\mathup` (based on `\operatorfont` from the `amsopn` package). This command also works fine with other sans serif fonts like `cmbright`.

`\e` Moreover for `beamer`, which changes the default font family (to sans serif), `\e`, `\i`, `\j` have no effect without `\AtBeginDocument` and `\AtBeginDocument` is also necessary to redefine `\i` when calling the `hyperref` package which overwrites the `\i` definition.

```
\@ifundefined{mathup}{
  \newcommand*\mathup[1]{\operatorfont #1}}
}{\mm@warning{mathup} } % also in kpfonts and unicode-math

\ifmm@singleletter
  \mm@macro{e}{\mathup{e}}
  \AtBeginDocument{\let\oldi\i \let\oldj\j
    \renewcommand{\i}{\TextOrMath{\oldi}{\mathup{i}}}
    \renewcommand{\j}{\TextOrMath{\oldj}{\mathup{j}}} }
\fi
```

`\MathFamily` The following macros `\MathUp` and `\MathIt` are toggles that transform any chosen letter in math mode to roman or italic style. These switches can be used anywhere in the document or the preamble. They are based on the generic macro `\MathFamily`. To obtain a letter in roman style instead of italic, we need to change the mathcode digit that represents the font family: 1 to 0.

For example, except for `LuaLTEX`, mathcode of the ‘e’ letter is: `e="7165` (decimal 29029), with the second digit ‘1’ indicating “italic” style. To get a roman ‘e’, we need to change its mathcode to “7065.

When used in the preamble, we call `\MathFamily` by `\AtBeginDocument` for working with the beamer package. Let's notice that `\MathFamily` has an erratic behavior when `unicode-math` is loaded, but fortunately, in that case, the `\DeclareMathSymbol` can be used instead, even outside the preamble.

```

\newcount\mm@charcode
\newcount\mm@charclass
\newcount\mm@charfam
\newcount\mm@charslot

\newcommand*\MathFamily[2]{%
  \mm@charfam=#2
  \ifluatex
    \mm@charclass=\Umathcharclass'#1
    %\mm@charfam=\Umathcharfam'#1
    \mm@charslot=\Umathcharslot'#1
    \Umathcode'#1=\mm@charclass \mm@charfam \mm@charslot
  \else
    \mm@charcode=\mathcode'#1
    % extract charclass
    \@tempcnta=\mm@charcode
    \divide\@tempcnta by "1000
    \multiply\@tempcnta by "1000 % charclass
    \mm@charclass=\@tempcnta
    % extract charslot
    \@tempcnta=\mm@charcode
    \@tempcntb=\mm@charcode
    \divide\@tempcnta by "100
    \multiply\@tempcnta by "100 % charclass + charfam
    \advance\@tempcntb by -\@tempcnta % charslot
    \mm@charslot=\@tempcntb
    % construct charcode
    \mm@charcode=\mm@charclass
    \multiply\mm@charfam by "100
    \advance\mm@charcode by \mm@charfam
    \advance\mm@charcode by \mm@charslot
    \mathcode'#1=\mm@charcode
  \fi
}

\MathUp

\newcommand*\@MathUp[1]{
  \ifpackageloaded{unicode-math}{
    \DeclareMathSymbol{#1}{\mathalpha}{operators}{'#1}
  }{
    \MathFamily{#1}{0}
  }
}

\newcommand*\MathUp[1]{%
  \ifx\@onlypreamble\@notprerr % not in preamble
    \@MathUp{#1}
  \else % in preamble

```

```

        \AtBeginDocument{\@MathUp{#1}}
    \fi
}

\MathIt
\newcommand*\@MathIt[1]{
    \ifpackageloaded{unicode-math}{
        \DeclareMathSymbol{#1}{\mathalpha}{letters}{‘#1}
    }{
        \MathFamily{#1}{1}
    }
}

\newcommand*\MathIt[1]{%
    \ifx\@onlypreamble\@notprerr % not in preamble
        \@MathIt{#1}
    \else % in preamble
        \AtBeginDocument{\@MathIt{#1}}
    \fi
}

```

With a similar approach we could also create additional macros to set any letter in bold or sans serif. However, there is no default family number associated with these typefaces. The family number depends on the font package being loaded and may vary depending on specific `\DeclareSymbolFont` used. Therefore, setting letters in bold or sans serif requires additional consideration and may not have a straightforward solution.

\MathNumbers In addition to `\MathUp` and `\MathIt`, we also offer the following command to set a group of letters, among ‘e, i, j’, in roman family.

```

\newcommand*\MathNumbers[1]{%
    \in@{e}{#1} \ifin@ \MathUp{e} \fi
    \in@{i}{#1} \ifin@ \MathUp{i} \fi
    \in@{j}{#1} \ifin@ \MathUp{j} \fi
}

```

\apply With the inverse switch `\MathNormal`, you can apply the normal (italic) style on any comma-separated list of characters. This is achieved using the powerful macro `\apply`, e.g. `\apply\macro{arg1,arg2}` expands to `\macro{arg1}\macro{arg2}`. So `\apply\MathUp{e,i,j}` is equivalent to `\MathUp{e}\MathUp{i}\MathUp{j}`. I discovered this powerful macro on iterate190.rssing.com by searching for “TeX How to iterate over a comma separated list”. The answer was posted under the pseudonym ‘wipet’ on 2021/02/26. Let its author, Petr Olšák, be thanked. This macro allows to accomplish tasks that usual loop instructions like `\@for` or `\foreach` cannot achieve due to errors like “! Improper alphabetic constant”. For example, if you try `\def\letter{A} \MathUp{\letter}` it will fail.

```

\def\apply#1#2{\apply@#1#2,\apply@,}
\def\apply@#1#2,{\ifx\apply@#2\empty

```

```

\else #1{#2}\afterfi@{\apply@#1}\fi}
\def\afterfi@#1#2\fi{\fi#1}

\MathNormal Apply \MathIt on a list argument.
\newcommand*\MathNormal[1]{\apply\MathIt{#1} }

\enumber The following commands were used originally (until version 2.2) to set the
\inumber math letters e, i or j in upright shape, but only worked in the preamble. This is
\jnumber now handled by the more powerful \MathUp command, but the old commands are
maintained as alias for \MathUp.

\mm@macro{enumber}{\MathUp{e}}
\mm@macro{inumber}{\MathUp{i}}
\mm@macro{jnumber}{\MathUp{j}}

Commands for vectors and tensors follow.

\arrowvect
\boldvect \newboolean{arrowvect}
\boldvectcommand \setboolean{arrowvect}{true}
\newcommand{\arrowvect}{\setboolean{arrowvect}{true}}
\newcommand{\boldvect}{\setboolean{arrowvect}{false}}
\newcommand{\boldvectcommand}{\boldsymbol} % from amsbsy package

\vect
\hvect \mm@macro{vect}{\ifthenelse{\boolean{arrowvect}}{
\hvec \vv}{\boldvectcommand}} % \if...\fi doesn't work reliably here
\newcommand*\{hvect}[1]{\vect{\vphantom{t}#1}}
\newcommand*\{hvec}[1]{\vec{\vphantom{t}#1}}

\norm We first define the macro corresponding to \norm* with alternatives for the four
math styles: displaystyle, textstyle, scriptstyle and scriptscriptstyle. Then the
macro \norm chooses between the starred version and the classic \left\Vert
...\right\Vert structure, depending on the difference between the height and
depth of the argument.

\newcommand*\{norm}[1]{
\mbox{\raisebox{1.75pt}{\small$\bigl\Vert$}} #1
\mbox{\raisebox{1.75pt}{\small$\bigr\Vert$}} }
% Absolute lengths work better here than relative lengths
\newcommand*\{norm}[1]{
\mbox{\footnotesize\raisebox{1pt}{\scriptstyle$\Vert$}} #1
\mbox{\footnotesize\raisebox{1pt}{\scriptstyle$\Vert$}} }
\newcommand*\{norm}[1]{
\mbox{\tiny\raisebox{1pt}{\scriptscriptstyle$\Vert$}} #1
\mbox{\tiny\raisebox{1pt}{\scriptscriptstyle$\Vert$}} }
\newcommand*\{norm@star}[1]{
\mathchoice{\norm{#1}}{\norm{#1}}{\norm{#1}}{\norm{#1}} }
\newcommand*\{mm@norm}[1]{
\sbox\@tempboxa{#1$}
\@tempdima=\ht\@tempboxa
\@tempdimb=\dp\@tempboxa
\addtolength{\@tempdima}{-\@tempdimb}

```

```

\ifdim \@tempdima < 1.85ex
  \left\lVert #1 \right\rVert
\else
  \norm@star{#1}
\fi
}
\mm@macro{norm}{\@ifstar{\norm@star}{\mm@norm}}

```

`\innerprod`

```

\@ifundefined{innerprod}{
  \newcommand*{\innerprod}[2]{\left\langle #1, #2\right\rangle}{
  \mm@warning{innerprod} }

```

`\di` Operators, as for example common function names, are generally defined using the `\DeclareMathOperator` command or `\operatorname` for occasional use. Operators are then typeset in roman, thanks to `\operatorfont`, and with appropriate thin space before and after the operator name. However, for the ‘differential’ operator `d`, no space should be left after it to obtain, for example, ‘ dx ’ instead of ‘ $d x$ ’.

Two other operators, suggested by ‘quark67’, are provided to represent differences or variations: `\opDelta` and `\opdelta`. They use the Greek letters Δ and δ with appropriate spacing, similar to `\di`. Romain Noël suggested to enhance `mismath` with other special functions usually represented by Greek letters: Γ, ζ . Since this involves more sophisticated code and usage, the handling of Greek letters is deferred to the end.

```

\mm@macro{di}{\operatorname{d}\mathopen{}}

```

`\P` For the domain of probability, we provide the macros `\P`, `\E`, and `\V`, which
`\E` are defined as operators. They are typeset in roman, as for any operator, but this
`\V` can be changed to double-struck style (or another style) if desired.

```

\newcommand\probastyle{}
\ifmm@singleletter
  \let\Par\P % end of paragraph symbol
  \renewcommand{\P}{\TextOrMath{\Par}{\operatorname{\probastyle{P}}}}
  \mm@macro{E}{\operatorname{\probastyle{E}}}
  \mm@macro{V}{\operatorname{\probastyle{V}}}
\fi

\newcommand*\MathProba[1]{%
  \PackageWarning{mismath}{The macro \string\MathProba\space
    is obsolete \MessageBreak and no longer supported}
}

```

`nofunction` (*opt.*) Standard operators and function identifiers are presented below. They are defined only if the option `nofunction` has not been enabled.

```

\ifmm@nofunction\else
  \mm@operator{\adj}{adj}
  \mm@operator{\Aut}{Aut}

```

```

\mm@operator{\codim}{codim}
\mm@operator{\codom}{codom}
\mm@operator{\coker}{coker}
\mm@operator{\Conv}{Conv}
\mm@operator{\cov}{cov}
\mm@operator{\Cov}{Cov}
\mm@macro{curl}{\operatorname{\vect{\mathup{curl}}}}
\mm@operator[divg]{\divg}{div}
\mm@operator{\dom}{dom}

\mm@operator{\End}{End}
\mm@operator{\erf}{erf}
\mm@macro{grad}{\operatorname{\vect{\mathup{grad}}}}
\mm@operator[Hv]{\Hv}{H}
\mm@operator{id}{id} % mathop or mathord?
\mm@operator{Id}{Id}
\mm@operator{im}{im}
\mm@operator{lb}{lb}
\mm@operator{lcm}{lcm}
\mm@operator{ord}{ord}
\mm@operator{ran}{ran}

\mm@operator{rank}{rank}
\mm@operator{Res}{Res}
\mm@macro{rot}{\operatorname{\vect{\mathup{rot}}}}
\mm@operator{sgn}{sgn}
\mm@operator{sinc}{sinc}
\mm@operator[spa]{\spa}{span}
\mm@operator{\supp}{supp}
\mm@operator{tr}{tr}
\mm@operator{var}{var}
\mm@operator{Var}{Var}
\mm@operator[Zu]{\Zu}{Z}

\mm@operator{\arccot}{arccot}
\mm@operator{\sech}{sech}
\mm@operator{\csch}{csch}
\mm@operator{\arsinh}{arsinh}
\mm@operator{\arcosh}{arcosh}
\mm@operator{\artanh}{artanh}
\mm@operator{\arcoth}{arcoth}
\mm@operator{\arsech}{arsech}
\mm@operator{\arcsch}{arcsch}

\mm@operator[FT]{\FT}{\mathcal{F}}
\mm@operator[LT]{\LT}{\mathcal{L}}
\fi

```

`\Re` If `unicode-math` is loaded, it will redefine the commands `\Re` and `\Im` in `\AtBeginDocument`. Hence `\AfterEndPreamble...` is used to ensure that the mis-math redefinitions occur after those made by `unicode-math`.

```

\AtEndPreamble{\AtBeginDocument{% cannot be replaced by \AfterEndPreamble
\ifmm@classicReIm\else

```

```

\let\oldRe\Re \let\oldIm\Im
\let\Re\relax \let\Im\relax
\DeclareMathOperator{\Re}{Re} % only in preamble
\DeclareMathOperator{\Im}{Im}
\fi
}}
```

`\bigO` The following operators are easily defined thanks to `\mm@operator`.

```

\bigO \mm@operator[bigO]{\bigO}{\mathcal{O}}
\lito \mm@operator[bigO]{\bigO}{O}
\lito \mm@operator[lito]{\lito}{o}
```

`lineargroups` (*opt.*) The redefinition of `\O` is specific to math mode. It is not overwritten by `beamer` or `unicode-math`.

```

\ifmm@lineargroups
\mm@operator{\GL}{GL}
\mm@operator{\SL}{SL}
\mm@operator{\Sp}{Sp}
\ifmm@singleletter
\let\oldO\O
\renewcommand{\O}{\TextOrMath{\oldO}{\operatorname{O}}}
\mm@operator{\U}{U}
\fi
\mm@operator{\SO}{SO}
\mm@operator{\SU}{SU}
\fi
```

`\C` When using a Cyrillic language, the command `\C` may already be defined, but only for use in text mode. In that case, `mismath` redefines `\C` to be used in math mode without interfering with its existing text-mode definition. The test is delayed with `\AfterEndPreamble`. The redefinition of `\H` concerns only math mode. It is not overwritten by `beamer` or `unicode-math`.

```

\mm@macro{mathset}{\mathbf}
\ifmm@singleletter
\mm@macro{R}{\mathset{R}}
\AfterEndPreamble{%
\@ifundefined{C}{\newcommand{\C}{\mathset{C}}}{
\let\oldC\C
\renewcommand{\C}{\TextOrMath{\oldC}{\mathset{C}}}}
}
\providecommand\onlymathC{\PackageWarning{mismath}{The macro
\string\onlymathC\space is obsolete and no longer useful}
\MessageBreak} }
\mm@macro{N}{\mathset{N}}
\mm@macro{Z}{\mathset{Z}}
\mm@macro{Q}{\mathset{Q}}
\let\oldH\H
\renewcommand{\H}{\TextOrMath{\oldH}{\mathset{H}}}
\mm@macro{F}{\mathset{F}}
\mm@macro{K}{\mathset{K}}
```

```

\fi

\ds
\dlim \mm@macro{ds}{\displaystyle}
\dsum \mm@macro{dlim}{\lim\limits}
\dprod \mm@macro{dsum}{\sum\limits}
\dcup \mm@macro{dprod}{\prod\limits}
\dcap \mm@macro{dcup}{\bigcup\limits}
\mcap \mm@macro{dcap}{\bigcap\limits}

\lbar
\hlbar \mm@macro{lbar}{\overline}
\ifundefined{hlbar}{
  \newcommand*{\hlbar}[1]{\overline{\vphantom{t}#1}}{
  \mm@warning{hlbar} }

\eqdef The \eqdef macro also exists in the package libertinustlmath, but with def written
very small. If you want to preserve the mismath macro, load libertinustlmath first
and call \let\eqdef\relax after it.

  \newcommand\eqdef{\stackrel{\mathup{def}}{=}}
  \newcommand\@eqdef{\@ifundefined{upDelta}{\PackageInfo{mismath}{%
    Command \string\upDelta\space is undefined in \string\eqdef*,
    \MessageBreak
    I use \string\Delta\space instead}\stackrel{\Delta}{=}}
  }{\stackrel{\upDelta}{=}}
  \mm@macro{eqdef}{\@ifstar{\@eqdef}{\eqdef}}

\asympreq The command \mathclap, used in \asympreq, puts its argument in a zero
width box and centers it, to avoid a lot of white space around the  $\sim$  symbol.
\unbr
\then \@ifundefined{asympreq}{
\compl \newcommand*{\asympreq}[1][\%
  \underset{\scriptscriptstyle\mathclap{#1}}{\sim}}{
  \mm@warning{asympreq} }
  \mm@macro{unbr}{\underbrace}
  \mm@macro{then}{\implies}
  \@ifundefined{compl}{
    \newcommand*{\compl}[1]{#1^{\mathup{c}}}} }{
    \mm@warning{compl} }

\integerint The definition of \integerint is delayed with \AtBeginDocument to check
whether \llbracket26 exist, or whether unicode-math or mathabx package is
loaded after mismath. The \left..\right structure may be useful to get the
correct size with big arguments like  $\llbracket 2^n, 2^{2^n} \rrbracket$ . Why did this macro take only one
argument and not two? This permits conveniently choosing another separator,
e.g.  $\llbracket a; b \rrbracket$ , and the typing of \{a,b\} instead of \{a\}{b} is easier (I know it's a bad
argument :).

  \AtBeginDocument{\@ifundefined{integerint}{
    \@ifundefined{llbracket}{
      \ifmm@unicodemath
        \newcommand*{\integerint}[1]{\left\lBrack #1\right\rBrack}

```

```

\else\@ifpackageloaded{mathabx}{
  \newcommand*\integerint[1]{\left\lbrack #1\right\rbrack}
}{
  \newcommand\integerint{\PackageError{mismath}{The
    \string\integerint\space macro requires either stmaryrd
    or fourier or unicode-math or mathabx package}{Load stmaryrd
    or fourier or unicode-math or mathabx in your preamble.} }
}
\fi
}{\newcommand*\integerint[1]{\left\llbracket #1\right\rrbracket}}
}{\mm@warning{integerint} }
}

```

`\tensor` The command `\mathbfsfit` (used for tensors) is already defined in `unicode-math` or `lua-unicode-math` and will not be redefined if these packages are loaded even after `mismath`.

```

\AtBeginDocument{
  \@ifundefined{mathbfsfit}{% used by unicode-math and lua-unicode-math
    \DeclareMathAlphabet{\mathbfsfit}{\encodingdefault}%
      {\sfdefault}{bx}{it}
  }{\mm@warning{mathbfsfit}} % isomath uses \mathsfbfit
}
\mm@macro{tensor}{\mathbfsfit}

```

`textshortcuts` (*opt.*)

The following text shortcuts have a starred version (except `\QED`), which generally produces an abbreviation with periods. When the abbreviation is not followed by a character, the last period must be followed by a normal space by using `\@`. But when there is a punctuation symbol immediately after, for example a comma, `\@` causes an issue. This is why we used `\xspace` from the `eponym` package. For abbreviations that may end a sentence, the spacing may be larger. Then put `\@` after the macro. The “iff” abbreviation does not correspond to the initial letters of the expression, thus we didn’t put any period in it. It is possible to place a period at the end as is customary in the US. The macro `\iff` already existed and had to be redefined, to be used in math or in text mode.

```

\ifmm@textshortcuts
  \mm@macro{QED}{Q.E.D.}
  \@ifundefined{ale}{
    \NewDocumentCommand{\ale}{s}{\IfBooleanTF#1{a.e\xperiod}%
      {almost everywhere\xspace}}
  }{\mm@warning{ale}}
  \@ifundefined{cc}{
    \NewDocumentCommand{\cc}{s}{\IfBooleanTF#1{c.c\xperiod}%
      {complex conjugate\xspace}}
  }{\mm@warning{cc}}
  \let\iffmath\iff
  \RenewDocumentCommand{\iff}{s}{\IfBooleanTF#1{iff\xspace}%
    {\TextOrMath{if and only if\xspace}{\iffmath}}}
  \@ifundefined{st}{
    \NewDocumentCommand{\st}{s}{\IfBooleanTF#1{s.t.\@\xspace}%
      {such that\xspace}}
  }{\mm@warning{st}}
  \@ifundefined{stp}{

```

```

\NewDocumentCommand{\stp}{s}{\IfBooleanTF#1{s.t.p.\@xspace}%
  {sufficient to prove\xspace}}
}{\mm@warning{stp}}
\@ifundefined{walog}{
  \NewDocumentCommand{\walog}{s}{\IfBooleanTF#1{w.a.l.o.g.\xperiod}%
    {without any loss of generality\xspace}}
}{\mm@warning{walog}}
\@ifundefined{wma}{
  \NewDocumentCommand{\wma}{s}{\IfBooleanTF#1{w.m.a.\@xspace}%
    {we may assume\xspace}}
}{\mm@warning{wma}}
\@ifundefined{wrt}{
  \NewDocumentCommand{\wrt}{s}{\IfBooleanTF#1{w.r.t.\@xspace}%
    {with respect to\xspace}}
}{\mm@warning{wrt}}
\@ifundefined{Walog}{
  \NewDocumentCommand{\Walog}{s}{\IfBooleanTF#1{W.a.l.o.g.%
    \@xspace}{Without any loss of generality\xspace}}
}{\mm@warning{Walog}}
\@ifundefined{Wma}{
  \NewDocumentCommand{\Wma}{s}{\IfBooleanTF#1{W.m.a.\@xspace}%
    {We may assume\xspace}}
}{\mm@warning{Wma}}
\@ifundefined{Wrt}{
  \NewDocumentCommand{\Wrt}{s}{\IfBooleanTF#1{W.r.t.\@xspace}%
    {With respect to\xspace}}
}{\mm@warning{Wrt}}
\fi

```

\txt The \mul macro has better spacing when \mathclose{} comes first and
 \mul \mathopen{} comes last. Compare $\sin(\frac{\pi}{3}) \times 2$ with $\sin(\frac{\pi}{3}) \times 2$. For \abs and
 \abs \floor, the \left...\right structure is inserted within a pair of curly braces to
 \card prevent incorrect spacing before the first delimiter when it follows a function name,
 \floor in cases where the package mleftright is used with the command \mleftright ac-
 \pow tivated. Compare $\ln|x|$ with $\ln|x|$. This issue was pointed out by ‘quark67’. The
 \lfrac \card definition is delayed with \AtBeginDocument in case another package or
 the user has provided its own definition in the preamble.

```

\@ifundefined{txt}{
  \newcommand*{\txt}[1]{\quad\text{#1}\quad }{
  \mm@warning{txt} }
\mm@macro{mul}{\mathclose{}\mathord{\times}\mathopen{}}
\@ifundefined{abs}{
  \newcommand*{\abs}[1]{\left\vert\right.#1\right\vert }{
  \mm@warning{abs} }
\AtBeginDocument{\mm@macro{card}{\abs}}
\@ifundefined{floor}{
  \newcommand*{\floor}[1]{\left\lfloor\right.#1\right\rfloor }{
  \mm@warning{floor} }
\@ifundefined{pow}{\newcommand*{\pow}[2]{%
  \sbox\@tempboxa{${#1}$}
  \@tempdima=\ht\@tempboxa
  \addtolength{\@tempdima}{-0.5ex}
  \@tempdimb=\dp\@tempboxa

```

```

\addtolength{\@tempdimb}{0.5ex}
\ifdim \@tempdima > \@tempdimb
  \@tempdimc=\@tempdima
\else
  \@tempdimc=\@tempdimb
\fi
\ifdim \@tempdimc < 1.27ex
  \left(#1 \right)^{#2}
\else
  \left(#1 \right)^{\!#2}
\fi
}{\mm@warning{pow} }
\@ifundefined{lfrac}{
  \newcommand*{\lfrac}[3][7mu]{%
    \frac{\mkern#1#2\mkern#1}{\mkern#1#3\mkern#1}} }{
  \mm@warning{lfrac} }

```

`system (env.)`

```

\newcommand{\systemstretch}{1.2}
\newcommand{\systemsep}{\medspace}
\newenvironment{system}[1][1]{
  \renewcommand{\arraystretch}{\systemstretch}
  \setlength{\arraycolsep}{0.15em}
  \left\{\begin{array}{@{\systemsep}#1@{}} %
}{\end{array}\right.}

```

`spmatrix (env.)`

```

\newenvironment{spmatrix}{
  \left(\begin{smallmatrix}
}{\end{smallmatrix}\right)}

```

`mathcols (env.)`

```

\newenvironment{mathcols}{% requires multicol to be loaded
  \ifmm@multicol
    \renewcommand{\columnseprule}{0.1pt}
    \begin{multicols}{2}
      \par\noindent\hfill
      \begin{math}\begin{aligned}\displaystyle
    \else
      \PackageError{mismath}{The mathcols environment
        requires the multicol package}{Load the multicol package
        in your preamble.}
    \fi
  }{
    \end{aligned}\end{math} \hfill\mbox{}
    \end{multicols}
}

```

`\changeacol`

```

\newcommand{\changeacol}{%
  \end{aligned}\end{math} \hfill\mbox{}
  \par\noindent\hfill

```

`\begin{math}\begin{aligned}\displaystyle`

`\specialgreekstdef` `\specialgreekstdef` takes key-value options to choose a Greek letter font without loading an entire package, thus without altering the other (italic) Greek letters. We achieve this with `\DeclareSymbolFont` and `\DeclareMathSymbol`. We simply have to know the name of the desired symbol font and the codes of the desired letters.

```

\newif\ifmm@lgrmath
\define@cmdkey{specialgreekstdef}[mm@]{lgrmath}[lmr]{\mm@lgrmathtrue}
\newif\ifmm@upgreek \newif\ifmm@upgreekSymbol
\define@choicekey{specialgreekstdef}{upgreek}[\mm@upgreek@option]%
    {Euler,Symbol,Symbolsmallscale}[Symbol]{\mm@upgreektrue}
\newif\ifmm@mathdesign
\define@choicekey{specialgreekstdef}{mathdesign}[\mm@mathdesign@option]%
    {Utopia,Garamond,Charter}[Charter]{\mm@mathdesigntrue}
\newif\ifmm@kpfnts
\define@choicekey{specialgreekstdef}{kpfnts}[\mm@kp@option]%
    {normal,light}[normal]{\mm@kpfntstrue}
\define@boolkeys{specialgreekstdef}[mm@]{fourier,pxfnts,txfnts}[true]
\newif\ifmm@fontspec
\define@cmdkey{specialgreekstdef}[mm@]{fontspec}[GFS Didot]%
    {\mm@fontspectrue}

\newcommand*\specialgreekstdef[1]{%
    \setkeys{specialgreekstdef}{#1}

    \ifmm@lgrmath
        \DeclareFontEncoding{LGR}{-}{-}
        \DeclareSymbolFont{mmupgr}{LGR}{\mm@lgrmath}{m}{n}
        \DeclareMathSymbol{\numpi}{\mathalpha}{mmupgr}{112}
        \DeclareMathSymbol{\numgamma}{\mathalpha}{mmupgr}{103}
        \DeclareMathSymbol{\numphi}{\mathalpha}{mmupgr}{102}
        \DeclareMathSymbol{\spdelta}{\mathalpha}{mmupgr}{100}
        \DeclareMathSymbol{\speDelta}{\mathalpha}{mmupgr}{68}
        \DeclareMathSymbol{\speGamma}{\mathalpha}{mmupgr}{71}
        \DeclareMathSymbol{\speZeta}{\mathalpha}{mmupgr}{122}
        \DeclareMathSymbol{\speSigma}{\mathalpha}{mmupgr}{115}
        \DeclareMathSymbol{\spePhi}{\mathalpha}{mmupgr}{70}

    \else\ifmm@fontspec
        \@ifpackageloaded{fontspec}{-}{-% unicode-math loads fontspec
            \PackageError{mismath}{\string\specialgreekstdef\space with
                the ‘fontspec’ option\MessageBreak
                needs the fontspec or unicode-math package,\MessageBreak
                which must be run with XeLaTeX or LuaLaTeX}{-}
        }
        \newfontfamily\mismathgreekfont{\mm@fontspec}[NFSSFamily=mgr]
        \DeclareSymbolFont{mmupgr}{TU}{mgr}{m}{n}
        \Umathchardef\numpi="7 \symmmupgr "03C0
        \Umathchardef\numgamma="7 \symmmupgr "03B3
        \Umathchardef\numphi="7 \symmmupgr "03C6
        \Umathchardef\spdelta="7 \symmmupgr "03B4
    \fi

```

```

\Umathchardef\speDelta="7 \symmmupgr "0394
\Umathchardef\speGamma="7 \symmmupgr "0393
\Umathchardef\spezeta="7 \symmmupgr "03B6
\Umathchardef\spesigma="7 \symmmupgr "03C3
\Umathchardef\spePhi="7 \symmmupgr "03A6

\else\ifmm@upgreek
\ifdefstring{\mm@upgreek@option}{Euler}{
\DeclareFontFamily{U}{eur}{\skewchar\font'177}
\DeclareFontShape{U}{eur}{m}{n}{%
<-6> eurm5 <6-8> eurm7 <8-> eurm10}{}
\DeclareSymbolFont{mmupgr}{U}{eur}{m}{n}
}{
\ifdefstring{\mm@upgreek@option}{Symbol}{
\DeclareSymbolFont{mmupgr}{U}{psy}{m}{n}
\mm@upgreekSymboltrue
}{
\ifdefstring{\mm@upgreek@option}{Symbolsmallscale}{
\DeclareFontFamily{U}{fsy}{}
\DeclareFontShape{U}{fsy}{m}{n}{<->s* [.9]psyr}{}
\DeclareSymbolFont{mmupgr}{U}{fsy}{m}{n}
\mm@upgreekSymboltrue
}{}}
\fi
\ifmm@upgreekSymbol
\DeclareMathSymbol{\numpi}{\mathalpha}{mmupgr}{'p}
\DeclareMathSymbol{\numgamma}{\mathalpha}{mmupgr}{'g}
\DeclareMathSymbol{\numphi}{\mathalpha}{mmupgr}{'j} % varphi
\DeclareMathSymbol{\spedelta}{\mathalpha}{mmupgr}{'d}
\DeclareMathSymbol{\speDelta}{\mathalpha}{mmupgr}{'D}
\DeclareMathSymbol{\speGamma}{\mathalpha}{mmupgr}{'G}
\DeclareMathSymbol{\spezeta}{\mathalpha}{mmupgr}{'z}
\DeclareMathSymbol{\spesigma}{\mathalpha}{mmupgr}{'s}
\DeclareMathSymbol{\spePhi}{\mathalpha}{mmupgr}{'F}

\else\ifmm@mathdesign
\ifdefstring{\mm@mathdesign@option}{Utopia}{
\DeclareSymbolFont{mmupgr}{OML}{mdput}{m}{n}
}{
\ifdefstring{\mm@mathdesign@option}{Garamond}{
\DeclareSymbolFont{mmupgr}{OML}{mdugm}{m}{n}
}{
\ifdefstring{\mm@mathdesign@option}{Charter}{
\DeclareSymbolFont{mmupgr}{OML}{mdbch}{m}{n}
}{}}
\fi

\else\ifmm@fourier
\DeclareFontEncoding{FML}{}{}
\DeclareFontSubstitution{FML}{futm}{m}{it}
\DeclareSymbolFont{mmupgr}{FML}{futm}{m}{it}

\else\ifmm@kpfonts
\ifdefstring{\mm@kp@option}{normal}{
\DeclareSymbolFont{mmupgr}{U}{jkpmia}{m}{it}

```

```

    }{
    \ifdefstring{\mm@kp@option}{light}{
        \DeclareSymbolFont{mmupgr}{U}{jkplmia}{m}{it}
    }{}}

\else\ifmm@pxfonts
    \DeclareSymbolFont{mmupgr}{U}{pxmia}{m}{it}

\else\ifmm@txfonts
    \DeclareSymbolFont{mmupgr}{U}{txmia}{m}{it}

\fi\fi\fi\fi\fi
% The following codes are common to the OML-based symbol fonts
\DeclareMathSymbol{\numpi}{\mathalpha}{mmupgr}{"19}
\DeclareMathSymbol{\numgamma}{\mathalpha}{mmupgr}{"0D}
\DeclareMathSymbol{\numphi}{\mathalpha}{mmupgr}{"27} % varphi
\DeclareMathSymbol{\spedelta}{\mathalpha}{mmupgr}{"0E}
\DeclareMathSymbol{\speDelta}{\mathalpha}{mmupgr}{"01}
\DeclareMathSymbol{\speGamma}{\mathalpha}{mmupgr}{"00}
\DeclareMathSymbol{\spezeta}{\mathalpha}{mmupgr}{"10}
\DeclareMathSymbol{\spesigma}{\mathalpha}{mmupgr}{"1B}
\DeclareMathSymbol{\spePhi}{\mathalpha}{mmupgr}{"08}
\fi\fi\fi

\opDelta Operators \opDelta, \opdelta... are defined at the end using \mmoperator.
\opdelta % \operatorname (and \DeclareMathOperator) do not work correctly.
\opGamma \mm@macro{opDelta}{\mmoperator{\speDelta}}
\opzeta \mm@macro{opdelta}{\mmoperator{\spedelta}}
\opsigma \mm@macro{opGamma}{\mmoperator{\speGamma}}
\opPhi \mm@macro{opzeta}{\mmoperator{\spezeta}}
\mm@macro{opsigma}{\mmoperator{\spesigma}}
\mm@macro{opPhi}{\mmoperator{\spePhi}}
}

\mmoperator This macro is inspired by amsmath's \operatorname, but without \operatorfont,
which is not supported by some Greek-letter fonts.
\newcommand*{\mmoperator}[1]{\mathop{\newmcodes@kern\z@#1}\nolimits%
\mathclose{}}

\setsspecialgreek First we define booleans, keys and values for \setsspecialgreek. Compatibility
with unicode-math is a bit tricky!
\newif\ifmm@setpi
\define@cmdkey{specialgreek}[mm@]{pi}[numpi]{\mm@setpittrue}
\newif\ifmm@setphi
\define@cmdkey{specialgreek}[mm@]{phi}[numphi]{\mm@setphittrue}
\newif\ifmm@setgamma
\define@cmdkey{specialgreek}[mm@]{gamma}[numgamma]{\mm@setgammatrue}
\newif\ifmm@setDelta
\define@cmdkey{specialgreek}[mm@]{Delta}[speDelta]{\mm@setDeltatrue}
\newif\ifmm@setdelta
\define@cmdkey{specialgreek}[mm@]{delta}[spedelta]{\mm@setdeltatrue}
\newif\ifmm@setGamma

```

```

\define@cmdkey{specialgreek}[mm@]{Gamma}[speGamma]{\mm@setGammatrue}
\newif\ifmm@setzeta
\define@cmdkey{specialgreek}[mm@]{zeta}[spezeta]{\mm@setzetatrue}
\newif\ifmm@setsigma
\define@cmdkey{specialgreek}[mm@]{sigma}[spesigma]{\mm@setsigmatrue}
\newif\ifmm@setPhi
\define@cmdkey{specialgreek}[mm@]{Phi}[spePhi]{\mm@setPhitrue}

\newcommand*\mm@specialgreek@err[1]{\PackageError{mismath}{Command
  \bslash #1 is unknown, \MessageBreak
  you can't use #1 as key value \MessageBreak
  in \string\specialgreek\space argument}{See documentation}
}

\newcommand*\mm@setgreeknumber[3]{%
  \ifundefined{#2}{
    \mm@specialgreek@err{#2}
  }{
    \ifundefined{original#1}{%
      \expandafter\let\csname original#1\expandafter\endcsname
        \csname#1\endcsname
    }{
      \ifthenelse{\boolean{mm@unicodemath}\AND\equal{#2}{up#1}}{%
        \expandafter\renewcommand\csname#1\endcsname{%
          \symup{\symbol{#3}}}
      }{%
        \expandafter\renewcommand\csname#1\endcsname{%
          \csname#2\endcsname}
      }
    }
  }
}

\newcommand*\mm@setgreekoperator[3]{%
  \ifundefined{#2}{
    \mm@specialgreek@err{#2}
  }{
    \ifundefined{original#1}{%
      \expandafter\let\csname original#1\expandafter\endcsname
        \csname#1\endcsname
    }{
      \ifthenelse{\boolean{mm@unicodemath}\AND\equal{#2}{up#1}}{%
        \expandafter\def\csname mm@#1\endcsname{%
          \symup{\symbol{#3}}}%
        \expandafter\renewcommand\csname#1\endcsname{%
          \expandafter\mmoperator\csname mm@#1\endcsname}
      }{%
        \expandafter\renewcommand\csname#1\endcsname{%
          \expandafter\mmoperator\csname#2\endcsname}
      }
    }
  }
}

\newcommand*\setspecialgreek[1]{
  \setkeys{specialgreek}{#1}
}

```

```

\@ifpackageloaded{unicode-math}{\mm@unicodemathtrue}{%

\ifmm@setpi
  \mm@setgreeknumber{pi}{\mm@pi}{\mm@pi}{03C0}
  \mm@setpifalse
\fi
\ifmm@setphi
  \mm@setgreeknumber{phi}{\mm@phi}{\mm@phi}{03C6}
  \mm@setphifalse
\fi
\ifmm@setgamma
  \mm@setgreeknumber{gamma}{\mm@gamma}{\mm@gamma}{03B3}
  \mm@setgammafalse
\fi
\ifmm@setDelta
  \mm@setgreekoperator{Delta}{\mm@Delta}{\mm@Delta}{0394}
  \mm@setDeltafalse
\fi
\ifmm@setdelta
  \mm@setgreekoperator{delta}{\mm@delta}{\mm@delta}{03B4}
  \mm@setdeltafalse
\fi
\ifmm@setGamma
  \mm@setgreekoperator{Gamma}{\mm@Gamma}{\mm@Gamma}{0393}
  \mm@setGammafalse
\fi
\ifmm@setzeta
  \mm@setgreekoperator{zeta}{\mm@zeta}{\mm@zeta}{03B6}
  \mm@setzetafalse
\fi
\ifmm@setsigma
  \mm@setgreekoperator{sigma}{\mm@sigma}{\mm@sigma}{03C3}
  \mm@setsigmafalse
\fi
\ifmm@setPhi
  \mm@setgreekoperator{Phi}{\mm@Phi}{\mm@Phi}{03A6}
  \mm@setPhifalse
\fi
}

```

specialgreek Command substitutions are delayed within `\AfterEndPreamble` when `unicode-math` or `mathgreek` is loaded, to avoid being overwritten by these packages, which also make delayed redefinitions.

```

\newif\ifspecialgreek@delayed

\newcommand*\specialgreek[2][]{%
  \ifx\@onlypreamble\@notprerr % not in preamble
    \ifthenelse{\equal{#1}{}}{\PackageWarning{mismath}{The
      optional argument of \string\specialgreek\s is ignored
      \MessageBreak out of the preamble}}
  }
  \setspecialgreek{#2}
\else % in the preamble

```

```

\ifthenelse{\equal{#1}{}}{\specialgreekdef{#1}}
\@ifpackageloaded{unicode-math}{\specialgreek@delayedtrue}{}
\@ifpackageloaded{mathgreek}{\specialgreek@delayedtrue}{}
\ifspecialgreek@delayed
\AfterEndPreamble{\setspecialgreek{#2}}
\else
\setspecialgreek{#2}
\fi
\fi
}

\normalgreek

\newcommand*\mm@normalgreek[1]{%
\@ifundefined{original#1}{\PackageWarning{mismath}{Command
\slash original#1 is undefined, I'll do nothing\MessageBreak}
}{\expandafter\let\csname #1\expandafter\endcsname
\csname original#1\endcsname}
}

\newcommand*\normalgreek[1]{%
\setkeys{specialgreek}{#1}
\ifmm@setpi \mm@normalgreek{pi}\mm@setpifalse \fi
\ifmm@setphi \mm@normalgreek{phi}\mm@setphifalse \fi
\ifmm@setgamma \mm@normalgreek{gamma}\mm@setgammafalse \fi
\ifmm@setDelta \mm@normalgreek{Delta}\mm@setDeltafalse \fi
\ifmm@setdelta \mm@normalgreek{delta}\mm@setdeltafalse \fi
\ifmm@setGamma \mm@normalgreek{Gamma}\mm@setGammafalse \fi
\ifmm@setzeta \mm@normalgreek{zeta}\mm@setzetafalse \fi
\ifmm@setsigma \mm@normalgreek{sigma}\mm@setsigmafalse \fi
\ifmm@setPhi \mm@normalgreek{Phi}\mm@setPhifalse \fi
}

\pinumber

\newcommand*\pinumber[1][]{% kept for compatibility
\ifthenelse{\equal{#1}{}}{% no optional argument
\ifx\@onlypreamble\@notprerr % not in preamble
\@ifundefined{savedpi}{
\PackageWarning{mismath}{%
\string\pinumber\space
must be used in the preamble first\MessageBreak}
}{\let\pi\savedpi}
\else % in the preamble, \AfterEndPreamble doesn't work
\AtEndPreamble{\AtBeginDocument{
\let\originalpi\pi
\specialgreekdef{lgrmath}
\let\savedpi\pi
}}
\fi
}{% command name or keyval option, necessarily in the preamble
\AtEndPreamble{\AtBeginDocument{% could be \AfterEndPreamble
\let\originalpi\pi
\@ifundefined{#1}{\specialgreekdef{#1} % keyval option

```

```

        \renewcommand{\pi}{\numpi}
    }{% known command name
    \ifpackageloaded{unicode-math}{\mm@unicodemathtrue}{}
    \ifthenelse{\boolean{mm@unicodemath}\AND\equal{#1}{uppi}}{%
        \renewcommand{\pi}{\symup{\symbol{"03C0}}}
    }{\renewcommand{\pi}{\csname #1\endcsname}}
    }
    \let\savedpi\pi
}}
}

```

`\pinormal`

```
\newcommand{\pinormal}{\normalgreek{\pi}}
```

`nosingleletter` (*opt.*) If the option `nosingleletter` is passed to the package, the single-letter macros will not be defined. Instead, the corresponding double-letter commands will be defined.

```

\ifmm@singleletter\else
    \mm@macro{ee}{\mathup{e}}
    \mm@macro{ii}{\mathup{i}}
    \mm@macro{jj}{\mathup{j}}
    \mm@macro{PP}{\operatorname{\probastyle{P}}}
    \mm@macro{EE}{\operatorname{\probastyle{E}}}
    \mm@macro{VV}{\operatorname{\probastyle{V}}}
    \mm@macro{RR}{\mathset{R}}
    \mm@macro{CC}{\mathset{C}}
    \mm@macro{NN}{\mathset{N}}
    \mm@macro{ZZ}{\mathset{Z}}
    \mm@macro{QQ}{\mathset{Q}}
    \mm@macro{HH}{\mathset{H}}
    \mm@macro{FF}{\mathset{F}}
    \mm@macro{KK}{\mathset{K}}
    \ifmm@lineargroups
        \mm@operator[OO]{\OO}{O}
        \mm@operator[UU]{\UU}{U}
    \fi
\fi

```

4 Change history

- | | |
|---|--|
| <p>v0.1 (2011/12/27)
First personal version.</p> <p>v1.0 (2019/04/11)
Initial published version.</p> <p>v1.1 (2019/04/20)
Changing the default font for <code>\pinumber</code> from Euler to Symbol.</p> <p>v1.2 (2019/04/27)</p> <ul style="list-style-type: none"> • placing commands <code>\enumber</code>, <code>\inumber</code>, <code>\jnumber</code> in <code>\AtBeginDocument</code> works fine with beamer now, • new general <code>\mm@operator</code> macro, • using <code>\mathup</code> instead of <code>\mathrm</code>, • including <code>mathtools</code>, • changing ‘Roman’ to ‘up’ in <code>\DeclareSymbolFont</code>, • changes in the documentation, • replacing <code>\PEroman</code> by <code>\PEupright</code>. <p>v1.3 (2019/05/08)</p> <ul style="list-style-type: none"> • using <code>\bslash</code> in the internal <code>\mm@warning</code> macro to type out a control sequence whose name is given as a parameter, • including the <code>mathfixs</code> package, • many corrections in the documentation. <p>v1.4 (2019/05/22)
Changing ‘up’ to ‘UpSh’ in <code>\DeclareSymbolFont</code> to prevent incompatibility with <code>unicode-math</code>.</p> <p>v1.5 (2019/05/30)</p> <ul style="list-style-type: none"> • a solution for using <code>\mul</code> with <code>\frac</code> use braces, • adding the <code>\paren</code> macro. <p>v1.6 (2019/09/06)
Removing the <code>mathfixs</code> package because of problems with fractions.</p> <p>v1.7 (2019/12/27)
Adding a table of contents to the documentation.</p> | <p>v1.8 (2020/11/15)
Small changes in the documentation, in particular mentioning an incompatibility when using ‘i’ with accent in beamer titles: (use <code>\^i</code> instead of <code>î</code>).</p> <p>v1.9 (2022/10/17)</p> <ul style="list-style-type: none"> • replacing ‘UpSh’ with ‘operators’ in <code>\DeclareSymbolFont</code>, • <code>\PackageWarning</code> replaced by <code>\PackageWarningNoLine</code> for existing macros, • replacing <code>\medspace</code> with <code>\thickspace</code> in <code>\lfrac</code>, • changing the documentation font: from <code>lmodern</code> to <code>Palatino</code> (<code>mathpazo</code>). <p>v1.10 (2022/10/25)
Updating the <code>\pinumber</code> code to prevent incompatibility with the new version of the <code>frenchmath</code> package (in which the default ‘upgreek’ option has been changed from Symbol to Euler).</p> <p>v2.0 (2022/11/11)</p> <ul style="list-style-type: none"> • enhancing <code>\pinumber</code> to use other Greek-letter packages (it is no longer compatible with the previous version), • removing <code>\paren</code> (useless), • slightly modifying <code>\hvect</code> and <code>\hlbar</code> (<code></code> instead of <code></code>), • several changes in the documentation (now the Charter font is used, with the <code>mathdesign</code> package). <p>v2.1 (2022/12/26)</p> <ul style="list-style-type: none"> • including the <code>ibackets</code> package to improve the management of square brackets, • new macros <code>\codim</code>, <code>\sinc</code>, <code>\var</code>, <code>\eqdef*</code>, • removing the warning for the obsolete <code>\paren</code> command, |
|---|--|

- a small change in the `\norm` command (using `\small` for the bars),
 - several changes in documentation.
- v2.2 (2023/01/06)
New option `ibridges` to optionally load the `ibridges` package because of errors when using `\DeclarePairedDelimiter` with square brackets.
- v2.3 (2023/02/09)
Introducing keyval options as alternatives to `\enumber`, `\inumber`, `\jnumber`, `\PEupright`, and for `ibridges`, `\boldvect` and `\arrowvect`.
- v2.4 (2023/02/18)
- new powerful macros `\MathUp`, `\MathIt` and also `\MathNumbers`, `\MathProba`, `\MathNormal`,
 - keyval options are no longer useful and have been removed,
 - forgotten loading the package `ifthen` in v2.3 (causing possible issues),
 - no more incompatibility when using ‘i’ with an accent in beamer titles.
- v2.5 (2023/02/23)
- unification of the code of `\MathUp` and `\MathIt`,
 - using the new powerful macro `\apply` in `\MathNormal` to act on a list,
 - new `\tensor` command.
- v2.6 (2023/03/01)
- bug fix in `\mm@macro`,
 - solving the incompatibility of the `\C` macro when using `babel` with Russian (thanks to Murray Eisenberg for this bug report on TeX StackExchange),
 - `\mathrm` added in the macro `\eqdef*`.
- v2.7 (2023/03/05)
- macros for sets of numbers (`\R`, `\C...`) now available only in math mode (following remarks by David Carlisle and Enrico Gregorio),
 - special warning when loading `babel` with Russian (`\C` will not be defined in that case).
- v2.8 (2023/07/26)
New macro `\onlymathC` designed for using `\C` in math mode when Russian language is loaded.
- v2.9 (2023/12/19)
New option `decimalcomma`.
- v2.10 (2024/02/20)
- better compatibility with `unicode-math` for the options `ibridges`, `decimalcomma` and the commands `\MathUp`, `\MathIt`,
 - explicit error message when using `mathcols` without loading the `multicol` package.
- v2.11 (2024/02/26)
- enhancements of the `\pinumber` macro with keyval options:
 - no necessity to load a Greek letters package,
 - improvements of compatibility with `unicode-math`;
 - changing the font to Adobe Utopia with the package `fourier`.
- v2.12 (2024/02/29)
- the `xparse` package has been removed by mistake in v2.11, causing some compatibility issues, it is loaded again by `mismath`,
 - improvements to make `\pinumber` work better with `unicode-math`.
- v3.0 (2024/03/15)
- rewriting the `\pinumber` command with a new `\pifonts` macro,
 - presenting other `lgrmath` values for `\pinumber` in the doc,

- The `\C` macro is now inside `\AtBeginDocument`,
 - `amsmath` isn't loaded explicitly because `mathtools` loads it,
 - bug fix with options `decimalcomma` and `ibackets`,
 - new option `nofunction` to lighten the package loading,
 - adding macros `\coker` and `\Res` as standard operator names,
 - new option `classicReIm` to deactivate `\Im` and `\Re` redefinition,
 - new option `otherReIm` to provide an alternative writing with `cmsy` font,
 - removing the `\PEupright` command,
 - default space in the `\lfrac` macro increased from `\: (5mu)` to `7mu`,
 - new optional parameter for adjusting the space in `\lfrac`,
 - changing the `\vphantom` argument in `\hvect`, `\hvec` and `\lbar` from 't' to 'A'.
- v3.1 (2024/06/16)
- adding a change history,
 - bug fix with the `\C` macro when using `hyperref` with `LuaTeX` or `XYTeX` engines,
 - the Δ produced in `\eqdef*` is now obtained with `\upDelta` (`\mathrm`, used before, doesn't work generally),
 - several relevant suggestions and remarks by 'quark67':
 - new macros for variations: `\opDelta` and `\opdelta`,
 - redefinition of `\then` as an alias for `\implies`,
 - improvement of the `\abs` macro,
 - including the `mleftright` package;
 - new implementation of `\di`,
 - improvement of the `\mul` macro.
- v3.2 (2025/10/08)
- conditional loading of the `xparse` package since it is obsolete with the October 2020 `LATEX` release,
 - when used in the preamble, `\pinumber` is now called within `\AtEndPreamble` to preserve its effect when used in combination with the `mathgreek` package.
- v3.3 (2026/08/16)
- handling Greek letters other than pi to represent special constants or operators,
 - improving the `\norm` and `\pow` macros to work with normal-sized arguments,
 - `\hvect`, `\hvec` and `\hlbar` use a phantom 't' again, instead of phantom 'A',
 - alternatives for single-letter macros (with `nosingletter`),
 - removing `\Mathproba` and `\onlymathC`  changing in `\C`,
 - new example of use of `\probastyle` inside a math formula,
 - removing the option `otherReIm`,
 - new options `lineargroups`, `textshortcuts`, following suggestions by Romain Noël,
 - `\iif` now obsolete, replaced by `\iff` in text mode only,
 - new commands `\innerprod`, `\FT`, `\LT`, `\asympteq`, `\compl`, `\integerint`, `\H`, `\Hv`, `\dom`, `\codom`, `\ran`, `\ord`, `\supp`, `\card`, `\floor`,
 - changing the definition of `\eqdef*` to use `\Delta` when `\upDelta` is not defined,
 - a few new references: Bourbaki, `mathalpha`, `xpunctuate`, `foreign`,
 - other changes in the doc:
 - explanations about `ibackets` have been lightened,
 - new example with `mathcols`,
 - turning back to the `lmodern` font to get a better contrast,
 - providing an index,
 - considerations on single-letter macros,
 - English revision.

References

- [1] *Typesetting mathematics for science and technology according to ISO 31/XI*, Claudio Beccari, TUGboat Volume 18 (1997), No. 1.
<http://www.tug.org/TUGboat/tb18-1/tb54becc.pdf>.
- [2] *Guide for the Use of the International System of Units (SI)*, NIST (National Institute of Standards and Technology), updated March 4, 2020
<https://www.nist.gov/pml/special-publication-811>.
- [3] *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.
- [4] *ÉLÉMENTS DE MATHÉMATIQUES, LIVRE III, TOPOLOGIE GÉNÉRALE*, N. Bourbaki, Hermann 1960.
- [5] *esvect – Typesetting vectors with beautiful arrow with L^AT_EX 2_ε*, Eddie Sautrais, CTAN, v1.3 2013/07/11.
- [6] *The mleftright package*, Heiko Oberdiek, CTAN, v1.2 2019/12/03.
- [7] *The mathtools package*, Morten Høgholm, Lars Madsen, CTAN, v1.29 2022/06/29.
- [8] *amsmath – A_MS mathematical facilities for L^AT_EX*, Frank Mittelbach, Rainer Schöpf, Michael Downes, Davis M. Jones, David Carlisle, CTAN, v2.17n 2022/04/08.
- [9] *Experimental Unicode mathematical typesetting: The unicode-math package*, Will Robertson, Philipp Stephani, Joseph Wright, Khaled Hosny, and others, CTAN, v0.8r 2023/08/13.
- [10] *The mathalpha, AKA mathalfa package*, Michael Sharpe, CTAN, v1.145 2025/01/17.
- [11] *The fixmath package for L^AT_EX 2_ε*, Walter Schmidt, CTAN, v0.9 2000/04/11.
- [12] *isomath – Mathematical style for science and technology*, Günter Milde, CTAN, v0.6.1 2012/09/04.
- [13] *PM-ISOMath, The Poor Man ISO math bundle*, the pm-isomath package by Claudio Beccari, CTAN, v1.2.00 2021/08/04.
- [14] *The mathgreek package*, Antoine Missier, CTAN, v1.2 2024/05/07.
- [15] *The upgreek package for L^AT_EX 2_ε*, Walter Schmidt, CTAN, v2.0 2003/02/12.
- [16] *The mathdesign package*, Paul Pichaureau, CTAN, v2.31 2013/08/29.
- [17] *Kp-Fonts – The Johannes Kepler project*, Christophe Caignaert, CTAN, v3.34 20/09/2022.

- [18] *Fourier-GUTenberg*, Michel Bovani, CTAN, v1.3 2005/01/30.
- [19] *PX Fonts – Palatino-like fonts in support of mathematics*, Young Ryu, CTAN, 2000/12/14.
- [20] *TX Fonts – Times-like fonts in support of mathematics*, Young Ryu, CTAN, 2000/12/15.
- [21] *The LibertinusT1 Math Package*, Michael Sharpe, CTAN, v2.0.4 2024/01/14.
- [22] *The lgrmath package*, Jean-François B., CTAN, v1.0 2022/11/16.
- [23] *New TX font package*, Michael Sharpe, CTAN, v1.735 2024/03/01.
- [24] *The St Mary’s Road symbol font*, Jeremy Gibbons, Alan Jeffrey, CTAN, v2.02a march 2004.
- [25] *Mathabx – Three series of mathematical symbols*, Anthony Phan, CTAN, 2005/05/18.
- [26] *The foreign package for L^AT_EX 2_ε*, Philip G. Ratcliffe, CTAN, v2.7 2012/09/25.
- [27] *The xpunctuate package for L^AT_EX 2_ε*, Philip G. Ratcliffe, CTAN, v2.0 2023/08/13.
- [28] *The spacingtricks package*, Antoine Missier, CTAN, v1.9 2026/08/02.
- [29] *Intelligent brackets – The ibrackets package*, Antoine Missier, CTAN, v1.2, 2023/07/26.
- [30] *The interval package*, Lars Madsen, CTAN, v0.4 2019/03/06.
- [31] *The icomma package for L^AT_EX 2_ε*, Walter Schmidt, CTAN, v2.0 2002/03/10.
- [32] *The ncccomma package*, Alexander I. Rozhenko, CTAN, v1.0 2005/02/10.
- [33] *The decimalcomma package*, Antoine Missier, CTAN, v1.4 2023/12/30.
- [34] *L’extension pour T_EX et L^AT_EX systeme*, Christian Tellechea, CTAN, v0.32 2019/01/13.
- [35] *The spalign package*, Joseph Rabinoff, CTAN, 2016/10/05.
- [36] *The package nicematrix*, François Pantigny, CTAN, v6.14 2023/02/18.
- [37] *L’extension frenchmath*, Antoine Missier, CTAN, v3.1 2024/05/07.
- [38] *The Not So Short Introduction to L^AT_EX 2_ε*, the lshort package by Tobias Oetiker, Hubert Partl, Irene Hyna and Elisabeth Schlegl, CTAN, v6.4 2021/04/09. <http://tug.ctan.org/info/lshort/english/lshort.pdf>.
- [39] *The L^AT_EX Companion*, Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley, 2nd edition, Pearson Education, 2004.

Index

Numbers written in *italics* refer to the page where the corresponding entry is described; numbers underlined refer to the code definition.

A		E		K	
<code>\abs</code>	11 , 29	<code>\E</code>	6 , 24	<code>\K</code>	8 , 26
<code>\adj</code>	7 , 24	<code>\e</code>	3 , 20	<code>\KK</code>	18 , 37
<code>\ale</code>	10 , 28	<code>\EE</code>	18 , 37		
<code>\apply</code>	6 , 22	<code>\ee</code>	18 , 37	L	
<code>\arccot</code>	7 , 24	<code>\End</code>	7 , 24	<code>\lb</code>	7 , 24
<code>\arcosh</code>	7 , 24	<code>\enumber</code>	3 , 23	<code>\lbar</code>	9 , 27
<code>\arcoth</code>	7 , 24	environments:		<code>\lcm</code>	7 , 24
<code>\arcsch</code>	7 , 24	<code>mathcols</code>	14 , 30	<code>\lfrac</code>	11 , 29
<code>\arrowvect</code>	4 , 23	<code>spmatrix</code>	13 , 30	<code>\lito</code>	8 , 26
<code>\arsech</code>	7 , 24	<code>system</code>	12 , 30	<code>\LT</code>	7 , 24
<code>\arsinh</code>	7 , 24	<code>\eqdef</code>	9 , 27	M	
<code>\artanh</code>	7 , 24	<code>\erf</code>	7 , 24	<code>\mathbbsfit</code>	10 , 28
<code>\asympteq</code>	9 , 27	F		<code>mathcols (env.)</code> . .	14 , 30
<code>\Aut</code>	7 , 24	<code>\F</code>	8 , 26	<code>\MathFamily</code>	20
B		<code>\FF</code>	18 , 37	<code>\MathIt</code>	3 , 22
<code>\big0</code>	8 , 26	<code>\floor</code>	11 , 29	<code>\MathNormal</code>	3 , 23
<code>\bigo</code>	8 , 26	<code>\FT</code>	7 , 24	<code>\MathNumbers</code>	3 , 22
<code>\boldvect</code>	4 , 23	G		<code>\mathset</code>	8 , 26
<code>\boldvectcommand</code> . .	4 , 23	<code>\GL</code>	8 , 26	<code>\MathUp</code>	3 , 21
<code>\bslash</code>	19	<code>\grad</code>	7 , 24	<code>\mathup</code>	2 , 20
C		H		<code>\mm@macro</code>	20
<code>\C</code>	8 , 26	<code>\H</code>	8 , 26	<code>\mm@operator</code>	20
<code>\card</code>	11 , 29	<code>\HH</code>	18 , 37	<code>\mm@warning</code>	20
<code>\CC</code>	18 , 37	<code>\hlbar</code>	9 , 27	<code>\mmoperator</code>	16 , 33
<code>\cc</code>	10 , 28	<code>\Hv</code>	7 , 24	<code>\mul</code>	10 , 29
<code>\changecol</code>	14 , 30	<code>\hvec</code>	5 , 23	N	
<code>\codim</code>	7 , 24	<code>\hvect</code>	4 , 23	<code>\N</code>	8 , 26
<code>\codom</code>	7 , 24	I		<code>\NN</code>	18 , 37
<code>\coker</code>	7 , 24	<code>\i</code>	3 , 20	<code>\norm</code>	5 , 23
<code>\compl</code>	9 , 27	<code>\Id</code>	7 , 24	<code>\normalgreek</code> . . .	17 , 36
<code>\Conv</code>	7 , 24	<code>\id</code>	7 , 24	<code>\numgamma</code>	14 , 31
<code>\Cov</code>	7 , 24	<code>\iff</code>	10 , 28	<code>\numphi</code>	14 , 31
<code>\cov</code>	7 , 24	<code>\ii</code>	18 , 37	<code>\numpi</code>	14 , 31
<code>\csch</code>	7 , 24	<code>\Im</code>	8 , 25	O	
<code>\curl</code>	7 , 24	<code>\im</code>	7 , 24	<code>\O</code>	8 , 26
D		<code>\innerprod</code>	5 , 24	<code>\oldIm</code>	8 , 25
<code>\dcap</code>	9 , 27	<code>\integerint</code>	10 , 27	<code>\oldRe</code>	8 , 25
<code>\dcup</code>	9 , 27	<code>\inumber</code>	3 , 23	<code>\OO</code>	18 , 37
<code>\di</code>	5 , 24	J		<code>\opDelta</code>	5 , 14 , 33
<code>\divg</code>	7 , 24	<code>\j</code>	3 , 20	<code>\opdelta</code>	5 , 14 , 33
<code>\dlim</code>	9 , 27	<code>\jj</code>	18 , 37	<code>\opGamma</code>	6 , 14 , 33
<code>\dom</code>	7 , 24	<code>\jnumber</code>	3 , 23	<code>\opPhi</code>	6 , 14 , 33
<code>\dprod</code>	9 , 27	Options (package):		<code>\opsigma</code>	6 , 14 , 33
<code>\ds</code>	8 , 27	<code>classicReIm</code> . . .			
<code>\dsum</code>	9 , 27				

decimalcomma .	12 , 18	\rot	7 , 24	\then	9 , 27
ibricks	12 , 18	\RR	18 , 37	\tr	7 , 24
lineargroups . .	8 , 26	S			
nofunction	7 , 24	\sech	7 , 24	U	
nosingletter . .	18 , 37	\setspecialgreek .	33	\U	8 , 26
textshortcuts	10 , 28	\sgn	7 , 24	\unbr	9 , 27
\opzeta	6 , 14 , 33	\sinc	7 , 24	\upDelta	9 , 27
\ord	7 , 24	\SL	8 , 26	\UU	18 , 37
P					
\P	6 , 24	\SO	8 , 26	V	
\Par	6 , 24	\Sp	8 , 26	\V	6 , 24
\pinormal	17 , 37	\spa	7 , 24	\Var	7 , 24
\pinumber	17 , 36	\specialgreek . .	16 , 35	\var	7 , 24
\pow	11 , 29	\specialgreekdef	14 , 31	\vect	4 , 23
\PP	18 , 37	\speDelta	15 , 31	\VV	18 , 37
\probastyle	6 , 24	\spedelta	15 , 31	W	
Q					
\Q	8 , 26	\speGamma	15 , 31	\Walog	10 , 28
\QED	10 , 28	\spePhi	15 , 31	\walog	10 , 28
\QQ	18 , 37	\spesigma	15 , 31	\Wma	10 , 28
R					
\R	8 , 26	\spezeta	15 , 31	\wma	10 , 28
\ran	7 , 24	\spmatrix (env.) . .	13 , 30	\Wrt	10 , 28
\rank	7 , 24	\st	10 , 28	\wrt	10 , 28
\Re	8 , 25	\stp	10 , 28	Z	
\Res	7 , 24	\SU	8 , 26	\Z	8 , 26
T					
\R	8 , 26	\supp	7 , 24	\Zu	7 , 24
\ran	7 , 24	system (env.)	12 , 30	\ZZ	18 , 37
\rank	7 , 24	\systemsep	13 , 30		
\Re	8 , 25	\systemstretch . .	13 , 30		
\Res	7 , 24	T			
\R	8 , 26	\tensor	10 , 28		