

Miscellaneous mathematical macros

The mismath package*

Antoine Missier
antoine.missier@ac-toulouse.fr

February 26, 2024

Contents

1	Introduction	1
2	Usage	2
2.1	Mathematical constants	2
2.2	Vectors (and tensors)	5
2.3	Standard operator names	7
2.4	A few useful aliases	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	13
2.8	Old commands	14
3	Implementation	14

1 Introduction

According to the International Standards ISO 31-0:1992 to ISO 31-13:1992 (superseded by ISO 80000-2:2009), mathematical *constants* e , i , π should be typeset in roman (up-right shape) and not in italic like variables (see [1] [2] [3] [4]). This package provides some tools to achieve this automatically.

Even though it is recommended to typeset vectors names in bold italic style [2] [4], 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 Soudrais [5]. Additionally we provide a few more macros related to vectors with arrows, particularly to enhance the typesetting of the norm: $\|\overrightarrow{AB}\|$ instead of the \LaTeX version $\|\overline{AB}\|$, which is not vertically adjusted, or worse $\|\overrightarrow{AB}\|$ (when using `\left...\right`).

*This document corresponds to mismath v2.11, dated 2024/02/26. Thanks to François Bastouil for initial help in English translation.

The package also offers other macros for:

- tensors,
- some common operator names,
- a few useful aliases,
- enhancing spacing in mathematical formulas,
- systems of equations and small matrices,
- displaymath in double columns for lengthy calculations.

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 the mismath or the existing command, you can use `\let\<command>\relax`, before loading mismath, or after.

[*<options>*]

The mismath package loads the amsmath package [6] without any options. If you want to use amsmath with specific options (refer to its documentation), you can include these options when calling mismath, or you can load amsmath with the desired options before loading mismath. *When using the package unicode-math [7], mismath must be loaded before unicode-math*, just like amsmath.

In addition mismath loads the mathtools package by Morten Høgholm and Lars Madsen [8]. This package offers numerous helpful macros and improvements of the amsmath package.

A recommendation, although rarely followed, is to typeset uppercase Greek letters in italic shape, similar to other variables [4]. This can be automatically achieved, for some particular fonts, with packages such as fixmath by Walter Schmidt [9], isomath by Günter Milde [10] or pm-isomath by Claudio Beccari [11] and optionally with many others (such as mathpazo or mathptmx with the option `slantedGreek`). 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 shape¹. When running through Lua^AT_EX or X_YL^AT_EX you can also get this result by setting the option `math-style=ISO` in the unicode-math package.

2 Usage

2.1 Mathematical constants

`\mathup`

As for classic functions identifiers, *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 solution is to use the `\mathup`

¹The frenchmath package [25] ensures to follow the recommended French rules.

macro, which is preferable to `\mathrm`², for setting any group of letters in roman. For example 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
`\i` (e) or imaginary numbers (i) with `\mathup{e}` or `\mathup{i}`, the package provides
`\j` the `\e` command for Eulers number and `\i` or `\j` for imaginary numbers. Let us notice that `\i` and `\j` already exist in LaTeX. 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.

`\MathUp` Indeed, typing a lot of backslashes for constants like e, i, or j in a document with numerous formulas can become tiresome. To alleviate this, the package proposes a powerful solution with the macro `\MathUp{<char>}`. For example, when `\MathUp{e}` is called, any subsequent occurrence of e will automatically be set in roman (upright shape), without the need to type `\e` explicitly. The effect of this macro can be either global or local, depending on whether it is used outside or inside an environment or braces. Furthermore, you can call this macro in the preamble to apply the change from the beginning of the document. This powerful feature allows you to bring a document up to the standards effortlessly. In fact, `\MathUp` can be applied to any valid single character, offering flexibility for various use cases (another use of it with probability will be presented in section 2.3).

`\MathIt` When there are other occurrences of *e*, *i* or *j* as variables, you can still obtain italicized *e*, *i* or *j* using \TeX commands `\mathit` or `\mathnormal`, which are useful for a single use. However, you also have the option to use the inverse switch `\MathIt{<char>}`, which has a global effect when used outside environments or braces, or a local effect when used inside them. Similar to `\MathUp`, `\MathIt` can be applied to any single character.

`\MathNumbers` These macros enable you to set upright or normal (italic) typesetting for multiple letters in a single command. For instance, `\MathNumbers{e,i}` is equivalent to `\MathUp{e}\MathUp{i}`. In `\MathNumbers`, the comma separator between letters can be modified or removed as needed. In fact, this macro only affects the letters e, i, or j; it has no effect on other characters. On the other hand, `\MathNormal` can be utilized for probability as well (refer to section 2.3), and it 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`.

`\pinumber[<option>]` The mathematical constant π should also be typeset in upright shape (see [1], [2], [4]), which is different from italicized π . However, this recommendation is even less commonly followed compared to the one concerning e and i [1]. The `\pinumber` com-

²The `\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.

mand replaces the italic π with an upright π each time `\pi` is called. It functions in two different ways.

1. You can install a Greek letters package that provides the glyphs in upright shape. There are many available. Notably, let us mention `upgreek` [12], `mathdesign` [13], `kpfonts` [15], `fourier` [16] (used in the present document), `libertinustlmath`, `pxgreek`s, `txgreek`s³, `libgreek`, etc. A special mention goes to `lgrmath` of Jean-François Burnol [17] which allows the use of any Greek LGR-encoded font in math mode. These packages provide commands like `\uppi` (`upgreek`), `\piup` (`mathdesign`, `kpfonts`, `lgrmath`), `\otherpi` (`fourier`), etc.⁴

In this case, `\pinumber` must be called in the preamble with the name of the command (without the backslash) giving access to the upright pi (`piup`, `uppi`, `otherpi` ...) as the argument for the optional parameter. However, installing such a Greek letters package will modify all the other Greek letter glyphs.

By using the preliminary code `\MathNumbers{ei}\pinumber[otherpi]` (assuming the `fourier` package is loaded) you can achieve the following result:

$$e^{i\pi} = -1 \quad \text{yields} \quad e^{i\pi} = -1.$$

2. Without installing a package, it is possible to change only the glyph of pi without altering the original glyphs for the other Greek letters, which are typically in italics.

In this case, `\pinumber` must be called in the preamble with an optional argument of the `key=value` type. The key name corresponds to a package providing the same glyph. When a key is given without a value, `\pinumber` will choose a default value specified below (depending on the key). The following table summarizes the available options.

Option	Result	Option	Result
<code>lgrmath=lmr</code>	π	<code>mathdesign</code>	π
<code>lgrmath=Alegreya-LF</code>	π	<code>fourier</code>	π
<code>lgrmath=Cochineal-LF</code>	π	<code>kpfonts</code>	π
<code>lgrmath=...</code>	...	<code>pxfonts</code>	π
<code>upgreek=Euler</code>	π	<code>txfonts</code>	π
<code>upgreek=Symbol</code>	π		

- With the `lgrmath` key, we actually have numerous possibilities for values (any Greek letters math fonts in LGR encoding). The documentation of the `lgrmath` package explains how to check and visualize all available fonts on your distribution. We have only presented three of them. The default value is `lmr`.

³When using `pxgreek`s or `txgreek`s, they should be loaded *after* `mismath` to avoid an error due to conflict with the existing macros `\iint`, `\iiint`, `\iiint`, `\idotsint` in `amsmath`.

⁴They have also options to typeset all the Greek lowercase letters in upright shape by default, but this is not our goal here.

- With the `upgreek` key, the default value is `Symbol`. There is a third possible value, `Symbolsmallscale`, which provides the same character as `Symbol` but reduced in size by 10 %.
- With the `mathdesign` key, there are actually 3 possible values: `Utopia`, `Garamond`, or `Charter` (default value), but the glyphs obtained for π look very similar.
- With the `kpfonts` key, we have two possible values: `normal` (default) and `light`. The option `kpfonts=light` provides a slightly less bold character.
- The keys `pxfonts` (based on Palatino) and `txfonts` (based on Times) are booleans whose default value is `true` (when called).
- When `\pinumber` is called without an argument in the preamble, it corresponds to the option `lgrmath=lmr`. This π character is particularly well-suited for use with the default Computer Modern or Latin Modern font family.⁵

If `unicode-math` is used (which provides `\uppi`), the command `\pinumber[uppi]` will not work, and it can be quite complicated to make some Greek letters packages work. In any case, this package must be loaded after `unicode-math` and in `\AtBeginDocument`. However, `unicode-math` supports `\pinumber` very well with the previous `key=value` options, by calling `\pinumber[option]` after `unicode-math`.

`\itpi` When you activate `\pinumber`, the original italic π is still accessible using `\itpi`.

`\pinormal` In fact, `\pinumber` is a toggle, with its inverse toggle being `\pinormal`. The latter restores the `\pi` command to its default behavior. Thus, `\pinumber` can be used anywhere in the document (like `\pinormal`), but then without arguments and provided it has been initially called in the preamble, according to the procedures outlined above.

2.2 Vectors (and tensors)

`\vect` By default, the `\vect` command⁶, produces vectors with arrows (thanks to the `esvect` package by Eddie Soudrais⁷) which are more elegant than those produced by \LaTeX 's `\overrightarrow` command. The `esvect` package has an optional argument (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 arrows produced by default in [5].

`\boldvect` The `\vect` macro allows vector names to be typeset using bold italic font, as recommended by ISO [2] [3], instead of using arrows. By using the `\boldvect` command, you can modify the behavior of `\vect` locally or globally, depending on its placement

⁵It will look the same as the one provided by Günter Milde's `textalpha` package [14].

⁶The definition of most macros in this package, will only take effect if the macro has not been previously defined by another package. This ensures compatibility and avoids conflicts when using the `mismath` package with other \LaTeX packages.

⁷`esvect` provides the `\vv` macro used by `\vect`.

in the document (inside or outside a group or an environment):

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

`\boldvectcommand` By default `\boldvect` uses the `\boldsymbol` command⁸ from the `amssymb` 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 `\sympbf` command. To use an alternative command instead of `\boldsymbol` in `mismath`, redefine `\boldvectcommand`, for instance after loading `fixmath`:

`\renewcommand\boldvectcommand{\mathbfbold}.`

According to ISO rules, symbols for matrices are also in bold italic. Therefore you can use the same `\boldvect` command or create another alias.

`\arrowvect` At any moment, you can revert to the default behavior using the inverse switch `\arrowvect`. These switches can be placed 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 places a vertical phantom box containing “*t*”) to avoid inelegant effects. For example, writing

- $\overrightarrow{AB} = \overrightarrow{u} + \overrightarrow{AC}$, obtained with `\hvect{u}`, is better than $\overrightarrow{AB} = \vec{u} + \overrightarrow{AC}$;
- $\vec{a} \cdot \vec{b} = 0$, obtained with `\hvect{a}`, is better than $\vec{a} \cdot \vec{b} = 0$.

This adjustment ensures a more visually pleasing appearance when vectors with arrows are combined in an equation. The `\boldvect` and `\arrowvect` switches have the same effect on `\hvect` as they do on `\vect`.

`\hvec` In a similar way, `\hvec` raises the little arrow produced by the `\vec` command, to the height of the letter “*t*” (but `\boldvect` have no effect here):

- $\vec{\mathcal{P}} = \vec{f} \cdot \vec{v}$, obtained with `\hvec{v}`, is better than $\vec{\mathcal{P}} = \vec{f} \cdot \vec{v}$.
- $\vec{f} = m\vec{a}$, obtained with `\hvec{a}`, is better than $\vec{f} = m\vec{a}$.

`\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, relatively to the mathematical center line, whereas vectors with arrows are asymmetric objects. The code `\norm{\vec{h}}` raises a smaller double bar to produce $\|\vec{h}\|$ instead of $\|\vec{h}\|$ or $\|\vec{h}\|$. Let's notice that the height of the bars don't adjust to content, but however to context: main text, subscripts or exponents, e.g. $e^{\|\vec{h}\|}$. This macro is useful only for arguments of special height, such as \vec{h} or \overrightarrow{AB} and may give bad results in other situations.

⁸`\mathbfbf` produces upright bold font, even when used in combination with `\mathit`.

`\mathbfsfit` For tensors symbols, ISO rules recommend using sans serif bold italic, but there is no such math alphabet in the default \LaTeX mathematical style. However, the `mismath` package defines this alphabet (assuming the font encoding and package you use permits it) and provides the macro `\mathbfsfit` or its alias `\tensor`. By using `\tensor{T}` you can produce ***T***.

2.3 Standard operator names

`\di` The *differential* operator should be typeset in upright shape, not in italics, to distinguish it from variables (as mentioned in [1] [2] [4] [27]). To achieve this, we provide the `\di` command. Take a look at the following examples (notice the thin spaces before the d, just like with classic function's names):

$$\begin{aligned} & \text{\texttt{[\iint xy\di x\di y]}} & \iint xy dx dy \\ & \text{\texttt{[m\frac{\di^2x}{\di t^2} + h\frac{\di x}{\di t} + kx = 0]}} & m \frac{d^2x}{dt^2} + h \frac{dx}{dt} + kx = 0 \end{aligned}$$

This command can also represent *distance* (hence its name):

$$\lambda d(A, \mathcal{F}) + \mu d(B, \mathcal{H}).$$

`\P` To refer to probability⁹ and expectation the proper use is to typeset the capital letters P, E in roman just like any standard function identifier. This can be achieved with `\P` and `\E` commands.

`\Par` The `\P` command already existed to refer to the end of paragraph symbol ¶ and has been redefined, but this symbol can still be obtained with `\Par`.

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

`\MathProba` As for e, i or j, you can use `\MathUp{P}`, `\MathUp{E}` or `\MathUp{V}` to avoid typing many `\P`, `\E` or `\V`. However you can also achieve this in a single command with `\MathNormal` `\MathProba`, for example `\MathProba{P,E}`. We get the inverse toggle with `\MathIt` for any individual letter or `\MathNormal` for a list.

`\probastyle` Some authors use “blackboard bold” font to represent probability, expectation and variance: $\mathbb{P}, \mathbb{E}, \mathbb{V}$. The `\probastyle` macro sets the appearance of `\P`, `\E` and `\V`. For instance `\renewcommand\probastyle{\mathbb}`¹⁰ brings the previous “double-struck” letters. The `\mathbb` command comes from `amssymb` package (loaded by `amssymb` but also available standalone) which needs to be called in the preamble.

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

⁹ \LaTeX provides also `Pr` which gives \Pr .

¹⁰The effect of this redefinition is global or local to the container environment in which it is used.

<code>\adj</code>	<code>adj</code>	<code>\erf</code>	<code>erf</code>	<code>\Re</code>	<code>Re</code>
<code>\Aut</code>	<code>Aut</code>	<code>\grad</code>	$\overrightarrow{\text{grad}}$	<code>\rot</code>	$\overrightarrow{\text{rot}}$
<code>\codim</code>	<code>codim</code>	<code>\id</code>	<code>id</code>	<code>\sgn</code>	<code>sgn</code>
<code>\Conv</code>	<code>Conv</code>	<code>\Id</code>	<code>Id</code>	<code>\sinc</code>	<code>sinc</code>
<code>\cov</code>	<code>cov</code>	<code>\im</code>	<code>im</code>	<code>\spa</code>	<code>span</code>
<code>\Cov</code>	<code>Cov</code>	<code>\Im</code>	<code>Im</code>	<code>\tr</code>	<code>tr</code>
<code>\curl</code>	$\overrightarrow{\text{curl}}$	<code>\lb</code>	<code>lb</code>	<code>\var</code>	<code>var</code>
<code>\divg</code>	<code>div</code>	<code>\lcm</code>	<code>lcm</code>	<code>\Var</code>	<code>Var</code>
<code>\End</code>	<code>End</code>	<code>\rank</code>	<code>rank</code>	<code>\Zu</code>	<code>Z</code>

By default, operators returning vectors, `\grad` and `\curl` (or its synonym `\rot` rather 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. On the other hand, ‘im’ stands for the image of a linear transformation (like ‘ker’ for the kernel) whereas ‘Im’ is the imaginary part of a complex number. Please note that `\div` already exists (\div) and `\span` is a \TeX primitive; they haven’t been redefined. 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 (see 2.4), which is why we used `\Zu`, to designate the center of a group: $Z(G)$ (from German Zentrum).

`\oldRe` The `\Re` and `\Im` macros already existed to refer to real and imaginary part of a
`\oldIm` complex number, producing outdated symbols \Re and \Im . However, they have been redefined according to current usage, as mentioned in the above table. Nevertheless, it is still possible to obtain the old symbols with `\oldRe` and `\oldIm`.

The package `mismath` also provides some (inverse) circular or hyperbolic functions, that are missing in \LaTeX :

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

`\bigO` Asymptotic comparison operators (in Landau notation) are obtained with `\bigO`
`\bigo` or `\bigo` and `\lito` commands:
`\lito`

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

2.4 A few useful aliases

In the tradition of Bourbaki and D. Knuth, proper use requires that classic sets of numbers are typeset in bold roman: **R, C, Z, N, Q**, whereas “double-barred” or “openwork” letters ($\mathbb{R}, \mathbb{C}, \mathbb{Z}, \dots$) are reserved for writing at the blackboard [27]. 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, \F, \K.`

`\mathset` The `\mathset` command enables you to change the behavior of all these macros in a global way. By default, `\mathset` is an alias for `\mathbf`, but if you prefer openwork letters, you can simply place `\renewcommand\mathset{\mathbb}` where you want, for instance in the preamble after loading the `amsmath` package (which provides the “blackboard bold” typeface, also loaded by `amssymb`).

`\onlymathC` The macro `\onlymathC` is designed for cases when `\C` is already defined, but only in text mode (usually when loading the Russian language with `babel` or `polyglossia`). The macro preserves the original definition for text mode and allows you to use `\C` for the complex number set in math mode. For this purpose, simply call `\onlymathC` once in the preamble or anywhere in the document.

`\ds` The `\displaystyle` command is very common, so the `\ds` alias is provided. Not only it eases typing but also it makes source code more readable.

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

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

`\lbar` Large bars over expressions are obtained 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 “h”) with the `\hlbar` command, to correct uneven bars heights.

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

`\eqdef` The `\eqdef` macro writes the equality symbol topped with ‘def’, or with ‘Δ’ for `\eqdef*` (thanks to the `TeX` command `\stackrel`):

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

`\unbr` `\unbr` is an alias for `\underbrace`¹¹, making source code more compact.

¹¹The `mathtools` package by Morten Høgholm and Lars Madsen [8] provides a new and improved version of the `\underbrace` command, along with many other useful macros. It is loaded by `mismath`.

$$\backslash[(QAP)^n = \backslashunbr{\backslashQAP\backslashmul\backslashQAP\backslashmul\backslashcdots\backslashmul\backslashQAP}_{n\text{ times}}\backslash] \quad (QAP)^n = \underbrace{QAP \times QAP \times \cdots \times QAP}_{n \text{ times}}$$

`\iif` `\iif` is an alias for “if and only if”, to be used in text mode.

2.5 Improved spacing in mathematical formulas

`\then` The `\then` macro produces the symbol \Rightarrow surrounded by large spaces just like the
`\txt` standard macro `\iff` does it with \Leftrightarrow . Similarly, the `\txt`, based on the `\text` macro from the `amstext` package (loaded by `amsmath`), leaves em quad spaces (`\quad`) around the text. See the following example:

$$\backslash[\backslashln x=a \backslashthen x=e^a, \backslashtxt{rather than} \backslashln x=a \backslashLongrightarrow x=e^a \backslash] \\ \ln x = a \Rightarrow x = e^a, \text{ rather than } \ln x = a \Rightarrow x = e^a$$

`\mul` The multiplication symbol obtained with `\times` produces the same spacing as addition or subtraction operators, whereas division obtained with `/` is closer to its operands. This actually hides the priority of multiplication over $+$ and $-$. That’s why we provide the `\mul` macro, behaving like `/` (ordinary symbol) and leaving less space around than `\times`:

$$\lambda + \alpha \times b - \beta \times c, \text{ obtained with } \backslashmul, \text{ is better than } \lambda + \alpha \times b - \beta \times c.$$

When using `\mul` before a function name or around a `\left \dots \right` structure, the space may be too large on one side of `\mul`. To ensure the same amount of space on both sides of `\mul`, you can use thin negative spaces `\!` or enclose the function or structure with braces:

$$x \times \sin x, \text{ obtained with } x \backslashmul \{ \backslashsin x \}, \text{ is slightly better than } x \times \sin x.$$

$$\backslashsin \! \{ \backslashleft(\backslashfrac{\backslashpi}{3} \backslashright) \} \backslashmul 2 \\ \text{gives } \sin\left(\frac{\pi}{3}\right) \times 2, \text{ which is better than } \sin\left(\frac{\pi}{3}\right) \times 2.$$

The thin negative space after the function name is not relative to `\mul`, but is due to the fact that spaces around a `\left \dots \right` structure are bigger than those produced by single parenthesis (`\dots`).

`\pow` In the same way, when typesetting an exponent after a closing *big* parenthesis produced by `\right)`, the exponent appears to be a little too far from the parenthesis. To address this issue, the `\pow{<expr>}{<pow>}` command is provided, which sets `<expr>` between parentheses and adjusts the positioning of the exponent `<pow>` slightly closer to the right parenthesis¹². Compare:

$$e^a \sim \left(1 + \frac{a}{n}\right)^n \text{ which may be better than } e^a \sim \left(1 + \frac{a}{n}\right)^n.$$

`\abs` The correct typesetting of absolute value (or modular for a complex number) is achieved using `\lvert \dots \rvert`, rather than `|`, as the latter doesn’t maintain proper

¹²This macro gives bad results with normal-sized parenthesis.

spacing in some situations (when a sign follows the open delimiter). For bars whose height has to adapt to the content, we can use `\left\vert ... \right\vert` or, more simply, the `\abs{...}` command, which is equivalent¹³.

`\lfrac` The `\lfrac` macro behaves like `\frac` but with thicker spaces around the arguments, making the corresponding fraction bar slightly longer:

$$\backslash[\backslash\bar{Z} = \backslashlfrac{\backslash\bar{z_1-z_2}}{\backslash\bar{z_1+z_2}} \backslash] \quad \overline{Z} = \frac{\overline{z_1 - z_2}}{\overline{z_1 + z_2}}$$

`[ibackets]` Open intervals are commonly represented with parenthesis, e.g. $(0, +\infty)$, but sometimes square brackets are used, especially in French mathematics: $]0, +\infty[$. In that specific case, the space around the square brackets is often inappropriate, as in the expression $x \in]0, +\infty[$. To address this issue, we have redefined the brackets in the `ibackets` package [26]. This one can be optionally¹⁴ 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`.

In our code, the symbols `[` and `]` are set as ‘active’ characters, behaving like ordinary characters and not as delimiters in most cases. Therefore, a line break could occur between the two brackets, but it is always possible to transform them into delimiters using `\left` and `\right`.

However, when a bracket is *immediately* followed by a `+` or `-` character, it becomes an open delimiter. Therefore, when the left bound contains an operator sign, *you don't 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 blank 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 $[a, b] + [c, d]$.

Besides, there are other approaches, for example the `\interval` macro from the `interval` package [18], or `\DeclarePairedDelimiters` from the `mathtools` package [8] (but the latter is incompatible with `ibackets` for managing brackets).

`[decimalcomma]` In many countries, except notably in English-speaking countries, the comma is used as a decimal separator for numbers. However, in the math mode of \LaTeX , 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 is not appropriate 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 decimal comma in math mode: `icomma` by Walter Schmidt [19] and `nccomma` by Alexander I. Rozhenko [20]. The second package takes a more generic approach, however it poses several compatibility

¹³We could also define `\abs` using `\DeclarePairedDelimiter` from the `mathtools` package [8].

¹⁴This functionality is optional because there is a conflict when using another command for open intervals with square brackets defined by `\DeclarePairedDelimiter` from `mathtools` [8].

issues, in particular when running through Lua \LaTeX , using `unicode-math` and calling `\setmathfont`. Therefore we propose the `decimalcomma` package [21], functionally identical to that of `nccomma` but with lighter code and without the aforementioned incompatibility. It can be loaded by `mismath` using the `decimalcomma` package option¹⁵.

2.6 Environments for systems of equations and small matrices

`system` The `system` environment, defined in the `mismath` package, is used to represent a system of equations:

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

`\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 bracket. The `\systemsep` length allows you to adjust the gap between the bracket and the 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` and the same spacing as of the `cases` environment is obtained with `\renewcommand{\systemsep}{}`. The `\systemsep` command allows for greater flexibility in adjusting the spacing within the `system` environment.

`system[\coldef]` By default, a system is written like an array environment with only one column, left aligned. However the `system` environment has an optional argument that allows to create 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:

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

`\systemstretch` The default spacing between the lines of 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}{\stretch}`, where `\stretch` is the desired value 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 for is 1.2. Furthermore you can also use the end of the line with a spacing option, as demonstrated above with `\backslash\backslash[1ex]`, to control the spacing between specific lines in the system.

¹⁵`ibridges` and `decimalcomma` are the only options specific to the `mismath` package.

Another example with `\begin{system}[rl@{\quad}l]`¹⁶:

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

Let's also mention the `systeme` package [22] which provides a lighter syntax and automatic alignments for linear systems. Additionally, there is the `spalign` package [23], which offers a convenient and easy syntax for systems and matrices with visually appealing alignments.

`spmatrix`

The `amsmath` package offers several environments to typeset matrices : For example, the `pmatrix` environment surrounds the matrix with parenthesis, and the `smallmatrix` environment creates a smaller matrix suitable for insertion within a text line. We provide a combination of the these both functionalities with the `spmatrix` environment: `\vec{u}\begin{spmatrix}-1\\2\end{spmatrix}` yielding $\vec{u}\begin{pmatrix}-1\\2\end{pmatrix}$.

The `mathtools` package enhances the `amsmath` matrix environments and also provides a small matrix environment with parenthesis: `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 space before the left parenthesis is unfortunately too narrow compared to the space inside the parenthesis. To illustrate this, consider the following comparison: $\vec{u}\begin{pmatrix}-1\\2\end{pmatrix}$ (using `mismath`'s `spmatrix`) vs. $\vec{u}\begin{pmatrix}-1\\2\end{pmatrix}$ (using `mathtools` `psmallmatrix`).

For typesetting various kinds of matrices, let's mention the excellent `nicematrix` package by François Pantigny [24].

2.7 Displaymath in double columns

`mathcols`

The `mathcols` environment allows you to arrange “long” calculations in double columns, separated with a central rule, as shown in the following example. However, to use this feature, the `multicol` package must be loaded in the preamble. The `mathcols` environment activates mathematical mode in display style and uses an aligned environment.

$$\begin{array}{l|l} \frac{1}{2 \times \left(\frac{1}{4}\right)^n + 1} \geq 0.999 & \iff 4^n \geq 1998 \\ \iff 1 \geq 1.998 \left(\frac{1}{4}\right)^n + 0.999 & \iff n \ln 4 \geq \ln(1998) \\ \iff 0.001 \geq \frac{1.998}{4^n} & \iff n \geq \frac{\ln(1998)}{\ln 4} \approx 5.4 \\ & \iff n \geq 6 \end{array}$$

`\changeacol`

The `\changeacol` macro is used to switch to the next column, and alignments within the columns is done using the classic delimiters `&`, to separate entries, and `\\`, to start a new row.

¹⁶`@{...}` sets inter-column space.

```

\begin{mathcols}
    & \frac{1}{2} \mul {\pow{\frac{1}{4}}{n}} + 1} \geq 0.999 \\\
    \iff & 1 \geq 1.998 \pow{\frac{1}{4}}{n} + 0.999 \\\
    \iff & 0.001 \geq \frac{1.998}{4^n} \\\
\changeacol
    & \iff 4^n \geq 1998 \\\
    & \iff n \ln 4 \geq \ln(1998) \\\
    & \iff n \geq \frac{\ln(1998)}{\ln 4} \approx 5.4 \\\
    & \iff n \geq 6
\end{mathcols}

```

2.8 Old commands

Here is a summary table of old commands that were used until version 2.2. These commands are still functional and will be maintained for the time being, but a warning message indicates the new alternative. They used to work only in the preamble, affecting the entire document globally, and lacked an inverse switch. These old commands can now be replaced by the more versatile and powerful `\MathUp` macro, which can be used anywhere in the document or preamble and has an inverse switch `\MathIt`.

Old command	New alternative
<code>\enumber</code>	<code>\MathUp{e}</code>
<code>\inumber</code>	<code>\MathUp{i}</code>
<code>\jnumber</code>	<code>\MathUp{j}</code>
<code>\PEupright</code>	<code>\MathProba{PE}</code>

You can also utilize `\MathNumbers` instead of `\MathUp` with an argument containing all the constants you want to be typeset in roman (among ‘e, i, j’).

Additionally you can include V in the argument of `\MathProba` to refer to variance, (or even use `\MathUp{P}\MathUp{E}`).

In version 2.3 we attempted to replace these old commands with package options based on keyval. However, we found that this method was less efficient and have decided to abandon it. As a result, the command `\mismatchset` is now obsolete. Additionally, the command, `\paren`, which was used before version 2.0, is no longer supported.

3 Implementation

We load certain packages conditionally to avoid ‘option clash’ errors in cases where these packages have been previously loaded with other options.

```

1 \newif\ifmm@ibricks % initialized to false
2 \newif\ifmm@decimalcomma
3 \DeclareOption{ibricks}{\mm@ibrickstrue}
4 \DeclareOption{decimalcomma}{\mm@decimalcommatrue}
5 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{amsmath}}

```

```

6\ProcessOptions \relax
7\@ifpackageloaded{amsmath}{\RequirePackage{amsmath}}
8\@ifpackageloaded{mathtools}{\RequirePackage{mathtools}}
9\@ifpackageloaded{esvect}{\RequirePackage[b]{esvect}}
10\RequirePackage{ifthen}
11%\RequirePackage{xparse} % provides \NewDocumentCommand, now in LaTeX3
12\RequirePackage{xspace}
13\RequirePackage{iftex}
14\RequirePackage{etoolbox} % provides \AtEndPreamble
15\RequirePackage{xkeyval}
16

```

The package `unicode-math` causes some compatibility issues with the options `ibrrackets` or `decimalcomma`: the respective packages must be loaded *after* `unicode-math`, but `mismath` (like `amsmath`) must be loaded *before* `unicode-math`. And to complicate matters, `unicode-math` defines all its commands by `\AtBeginDocument`. Therefore we used the command `\AtBeginDocument` within `\AtEndPreamble` (from the `etoolbox` package).

Moreover the command `\mathbfsfit` (used for tensors) is already defined in `unicode-math` and will not be redefined if `unicode-math` is loaded.

```

17\newif\ifmm@unicodemath
18\newif\ifmm@multicol
19\AtEndPreamble{% necessary to work with unicode-math
20  \@ifpackageloaded{multicol}{\mm@multicoltrue}{\mm@multicolfalse}
21  \@ifpackageloaded{unicode-math}{\mm@unicodemathtrue}{
22    \mm@unicodemathfalse
23    \DeclareMathAlphabet{\mathbfsfit}{\encodingdefault}%
24      {\sfdefault}{bx}{it}}
25  \AtBeginDocument{% necessary to work with unicode-math
26    \ifmm@ibrrackets\RequirePackage{ibrrackets}\fi
27    \ifmm@decimalcomma\RequirePackage{decimalcomma}\fi
28  }
29}
30

```

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

```

31{\catcode'\|=z@ \catcode'\=12 \gdef\bslash{\}} % \bslash command
32

```

`\mm@warning` The next three internal macros serve as meta commands for conditionally defining
`\mm@macro` macros while providing a warning message if the macro already exists. These macros
`\mm@operator` can be useful in other packages as well.

```

33\newcommand\mm@warning[1]{
34  \PackageWarningNoLine{mismath}{
35    Command \bslash #1 already exist and will not be redefined}

```

```

36 }
37 \newcommand\mm@macro[2]{
38   \@ifundefined{#1}{
39     \expandafter\def\csname #1\endcsname{#2}
40   }\mm@warning{#1}}
41 }
42 \NewDocumentCommand\mm@operator{0{#3}mm}{%
43   \@ifundefined{#1}{
44     \DeclareMathOperator{#2}{#3}
45   }\mm@warning{#1}}
46 }
47

```

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`.

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

```

48 \@ifundefined{mathup}{
49   \providecommand*\mathup{[1]{\operatorfont #1}}
50   }\mm@warning{mathup} } % also in kpfonts (and unicode-math)
51 \mm@macro{e}{\mathup{e}}
52 \AtBeginDocument{\let\oldi\i \let\oldj\j
53   \renewcommand{\i}{\TextOrMath{\oldi}{\mathup{i}}}
54   \renewcommand{\j}{\TextOrMath{\oldj}{\mathup{j}}} }
55

```

`\MathFamily`

The following macros `\MathUp` and `\MathIt` are switches that transform any chosen letter in math mode to roman or italic style. These switches can be used anywhere in the document or 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 `Lua®TeX`, 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.

```

56 \newcount\mm@charcode
57 \newcount\mm@charclass
58 \newcount\mm@charfam
59 \newcount\mm@charslot
60
61 \newcommand*\MathFamily[2]{%
62   \mm@charfam=#2

```



```

63 \ifluatex
64   \mm@charclass=\Umathcharclass'#1
65   %\mm@charfam=\Umathcharfam'#1
66   \mm@charslot=\Umathcharslot'#1
67   \Umathcode'#1= \mm@charclass \mm@charfam \mm@charslot
68 \else
69   \mm@charcode=\mathcode'#1
70   % extract charclass
71   \@tempcnta=\mm@charcode
72   \divide\@tempcnta by "1000
73   \multiply\@tempcnta by "1000 % charclass
74   \mm@charclass=\@tempcnta
75   % extract charslot
76   \@tempcnta=\mm@charcode
77   \@tempcntb=\mm@charcode
78   \divide\@tempcnta by "100
79   \multiply\@tempcnta by "100 % charclass + charfam
80   \advance\@tempcntb by -\@tempcnta % charslot
81   \mm@charslot=\@tempcntb
82   % construct charcode
83   \mm@charcode=\mm@charclass
84   \multiply\mm@charfam by "100
85   \advance\mm@charcode by \mm@charfam
86   \advance\mm@charcode by \mm@charslot
87   \mathcode'#1=\mm@charcode
88 \fi
89 }
90
91 \newcommand*\MathUp[1]{%
92   \ifx\@onlypreamble\@notprerr % not in preamble
93     \ifmm@unicodemath
94       \DeclareMathSymbol{#1}{\mathalpha}{operators}{'#1}
95     \else
96       \MathFamily{#1}{0}
97     \fi
98   \else % in preamble
99     \AtBeginDocument{
100       \ifmm@unicodemath
101         \DeclareMathSymbol{#1}{\mathalpha}{operators}{'#1}
102       \else
103         \MathFamily{#1}{0}
104       \fi
105     }
106   \fi
107 }
108
109 \newcommand*\MathIt[1]{%
110   \ifx\@onlypreamble\@notprerr % not in preamble
111     \ifmm@unicodemath
112       \DeclareMathSymbol{#1}{\mathalpha}{letters}{'#1}

```

```

113     \else
114         \MathFamily{#1}{1}
115     \fi
116 \else % in preamble
117 \AtBeginDocument{
118     \ifmm@unicodemath
119         \DeclareMathSymbol{#1}{\mathalpha}{letters}{‘#1}
120     \else
121         \MathFamily{#1}{1}
122     \fi
123 }
124 \fi
125 }
126

```

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.

In addition to `\MathUp` and `\MathIt`, we also offer the following two commands to set a group of letters in roman typeface: one for mathematical constants, among ‘e, i, j’, and the other for probability operators, among ‘P, E, V’.

```

127 \newcommand*\MathNumbers[1]{%
128     \in@{e}{#1} \ifin@ \MathUp{e} \fi
129     \in@{i}{#1} \ifin@ \MathUp{i} \fi
130     \in@{j}{#1} \ifin@ \MathUp{j} \fi
131 }
132
133 \newcommand*\MathProba[1]{%
134     \in@{P}{#1} \ifin@ \MathUp{P} \fi
135     \in@{E}{#1} \ifin@ \MathUp{E} \fi
136     \in@{V}{#1} \ifin@ \MathUp{V} \fi
137 }
138

```

`\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 `\apply` macro, e.g. `\apply\macro{arg1,arg2}` expands to `\macro{arg1}\macro{arg2}`. Thus `\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 Olák, 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 instance, if you try `\def\letter{A} \MathUp{\letter}` it will fail because the control sequence `\letter` is not strictly equivalent here to the single character ‘A’.

```

139 \def\apply#1#2{\apply@#1#2,\apply@,}
140 \def\apply@#1#2,{\ifx\apply@#2\empty
141   \else #1{#2}\afterfi@{\apply@#1}\fi}
142 \def\afterfi@#1#2\fi{\fi#1}
143
144 \newcommand*\MathNormal[1]{% list argument
145   \apply\MathIt{#1}
146 }
147

```

The following commands were used until version 2.2 but still work. They were intended to set some letters in upright shape in math mode, but only worked in the preamble. This is now managed by the more powerful `\MathUp` command, and the old commands are maintained for compatibility reasons.

```

148 \newcommand{\enumber}{%
149   \PackageWarning{mismath}{Old command \string\enumber\space
150     is used. \MessageBreak
151     It can be replaced by \string\MathUp{e}}
152   \MathUp{e}
153 }
154 \newcommand{\inumber}{%
155   \PackageWarning{mismath}{Old command \string\inumber\space
156     is used. \MessageBreak
157     It can be replaced by \string\MathUp{i}}
158   \MathUp{i}
159 }
160 \newcommand{\jnumber}{
161   \PackageWarning{mismath}{Old command \string\jnumber\space
162     is used. \MessageBreak
163     It can be replaced by \string\MathUp{j}}
164   \MathUp{j}
165 }
166 \newcommand{\PEupright}{
167   \PackageWarning{mismath}{Old command \string\PEupright\space
168     is used. \MessageBreak
169     It can be replaced by \string\MathProba{PE}}
170   \MathUp{P}\MathUp{E}
171 }
172

```

Obtaining an upright Greek letter π must be handled differently. The switches are called `\pinumber` and `\pinormal` and can be used anywhere in the document.

But `\pinumber` must be called first in the preamble with an optional argument. This argument can be a valid command name that produces an upright pi letter (after having loading an appropriate package). When given without an argument in the preamble, `\pinumber` uses an LGR font encoding called `lmr`. A new feature (v2.11) is to use `\pinumber` with a `keyval` option to use many other Greek pi letters without loading a whole package, thus without altering the other (italic) Greek letters. We achieve

this with `\DeclareSymbolFont` and `\DeclareMathSymbol`. We just have to know the “name” of the desired symbol font. Compatibility with unicode-math is a bit tricky!

```

173 \newif\ifmm@lgr
174 \define@cmdkey{pinumber}[mm@]{lgrmath}[lgr]{\mm@lgrtrue}
175 \newif\ifmm@upgreek
176 \define@choicekey{pinumber}{upgreek}[\mm@upgreek@option]%
177   {Euler,Symbol,Symbolsmallscale}[Symbol]{\mm@upgreektrue}
178 \newif\ifmm@mathdesign
179 \define@choicekey{pinumber}{mathdesign}[\mm@mathdesign@option]%
180   {Utopia,Garamond,Charter}[Charter]{\mm@mathdesigntrue}
181 \define@boolkey{pinumber}[mm@]{fourier}[true]{ }
182 \newif\ifmm@kpfonts
183 \define@choicekey{pinumber}{kpfonts}[\mm@kp@option]%
184   {normal,light}[normal]{\mm@kpfontstrue}
185 \define@boolkey{pinumber}[mm@]{pxfonts}[true]{ }
186 \define@boolkey{pinumber}[mm@]{txfonts}[true]{ }
187
188 \newcommand*\pinumber[1][ ]{%
189   \ifthenelse{\equal{#1}{}}{% no argument given
190     \ifx\@onlypreamble\@notprerr % not in preamble
191       \@ifundefined{savedpi}{
192         \PackageWarning{mismath}{%
193           \string\pinumber\space
194           must be used in the preamble first}
195       }{\let\pi\savedpi}
196     \else % in the preamble
197       \AtBeginDocument{\let\itpi\pi}% must be here with unicode-math
198       \AtEndPreamble{\AtBeginDocument{
199         %\let\itpi\pi
200         \let\pi\relax
201         \DeclareFontEncoding{LGR}{}{ }
202         \DeclareSymbolFont{mm@grup}{LGR}{lgr}{m}{n}
203         \DeclareMathSymbol{\pi}{\mathalpha}{mm@grup}{"70}
204         \let\savedpi\pi
205       }}
206     \fi
207   }{% command name or keyval options, necessarily in the preamble
208     \AtBeginDocument{\let\itpi\pi}% must be here with unicode-math
209     \AtEndPreamble{\AtBeginDocument{
210       \@ifundefined{#1}{%
211         \setkeys{pinumber}{#1}
212         \let\pi\relax
213         \ifmm@lgr
214           \DeclareFontEncoding{LGR}{}{ }
215           \DeclareSymbolFont{mm@grup}{LGR}{\mm@lgrmath}{m}{n}
216           % may work with bold (b) instead of m
217           \DeclareMathSymbol{\pi}{\mathalpha}{mm@grup}{112}
218         \else\ifmm@upgreek

```

```

220 \ifdefstring{\mm@upgreek@option}{Euler}{
221   \DeclareFontFamily{U}{eur}{\skewchar\font'177}
222   \DeclareFontShape{U}{eur}{m}{n}{%
223     <-6> eurm5 <6-8> eurm7 <8-> eurm10}{}
224   \DeclareFontShape{U}{eur}{b}{n}{%
225     <-6> eurb5 <6-8> eurb7 <8-> eurb10}{}
226   \DeclareSymbolFont{mm@grup}{U}{eur}{m}{n}
227   \DeclareMathSymbol{\pi}{\mathord}{mm@grup}{"19} % 25
228 }{
229 \ifdefstring{\mm@upgreek@option}{Symbol}{
230   \DeclareSymbolFont{mm@grup}{U}{psy}{m}{n}
231   \DeclareMathSymbol{\pi}{\mathord}{mm@grup}{'p}
232 }{
233 \ifdefstring{\mm@upgreek@option}{Symbolsmallscale}{
234   \DeclareFontFamily{U}{fsy}{}
235   \DeclareFontShape{U}{fsy}{m}{n}{<->s* [.9]psyr}{}
236   \DeclareSymbolFont{mm@grup}{U}{fsy}{m}{n}
237   \DeclareMathSymbol{\pi}{\mathord}{mm@grup}{'p}
238 }{}}
239
240 \else\ifmm@mathdesign
241 \ifdefstring{\mm@mathdesign@option}{Utopia}{
242   \DeclareSymbolFont{mm@grup}{OML}{mdput}{m}{n}
243 }{
244 \ifdefstring{\mm@mathdesign@option}{Garamond}{
245   \DeclareSymbolFont{mm@grup}{OML}{mdugm}{m}{n}
246 }{
247 \ifdefstring{\mm@mathdesign@option}{Charter}{
248   \DeclareSymbolFont{mm@grup}{OML}{mdbch}{m}{n}
249 }{}}
250
251 \else\ifmm@fourier
252   \DeclareFontEncoding{FML}{-}{-}
253   \DeclareSymbolFont{mm@grup}{FML}{futm}{m}{it}
254
255 \else\ifmm@kpfonts
256 \ifdefstring{\mm@kp@option}{normal}{
257   \DeclareSymbolFont{mm@grup}{U}{jkpmia}{m}{it}
258 }{
259 \ifdefstring{\mm@kp@option}{light}{
260   \DeclareSymbolFont{mm@grup}{U}{jkplmia}{m}{it}
261 }}
262
263 \else\ifmm@pxfonts
264   \DeclareSymbolFont{mm@grup}{U}{pxmia}{m}{it}
265
266 \else\ifmm@txfonts
267   \DeclareSymbolFont{mm@grup}{U}{txmia}{m}{it}
268 \fi\fi\fi\fi\fi
269

```

```

270         \DeclareMathSymbol{\pi}{\mathord}{mm@grup}{"19}
271         \fi\fi
272
273         \let\savedpi\pi
274     }\renewcommand{\pi}{\csname #1\endcsname}\let\savedpi\pi}
275 }
276 }
277 }
278
279 \newcommand{\pinormal}{%
280   \ifundefined{itpi}{
281     \PackageWarning{mismath}{Command \string\itpi\space undefined,
282     \MessageBreak
283     use \string\pinumber\space in the preamble first}
284   }{
285     \ifmm@unicodemath
286       \ifundefined{savedpi}{
287         \PackageError{mismath}{Before using \string \pinormal,
288         \MessageBreak
289         you must call \string\pinumber\space in the preamble}{}}
290       \fi
291       \let\pi\itpi
292     }
293   }
294

```

And now the commands for vectors (and tensors).

```

295 \newboolean{arrowvect}
296 \setboolean{arrowvect}{true}
297 \newcommand{\arrowvect}{\setboolean{arrowvect}{true}}
298 \newcommand{\boldvect}{\setboolean{arrowvect}{false}}
299 \newcommand{\boldvectcommand}{\boldsymbol} % from amsbsy package
300 \mm@macro{vect}{\ifthenelse{\boolean{arrowvect}}{
301   \vv}{\boldvectcommand}} % doesn't work well with \if... \fi
302 \newcommand*{\hvect}[1]{\vect{\vphantom{t}#1}}
303 \newcommand*{\hvec}[1]{\vec{\vphantom{t}#1}}
304
305 \newcommand*{\@norm}[1]{
306   \mbox{\raisebox{1.75pt}{\small$\bigl\Vert$}} #1
307   \mbox{\raisebox{1.75pt}{\small$\bigr\Vert$}} }
308 % works better than with relative length
309 \newcommand*{\@@norm}[1]{
310   \mbox{\footnotesize\raisebox{1pt}{\small$\Vert$}} #1
311   \mbox{\footnotesize\raisebox{1pt}{\small$\Vert$}} }
312 \newcommand*{\@@@norm}[1]{
313   \mbox{\tiny\raisebox{1pt}{\small$\Vert$}} #1
314   \mbox{\tiny\raisebox{1pt}{\small$\Vert$}} }
315 \ifundefined{norm}{\providecommand*{\norm}[1]{
316   \mathchoice{\@norm{#1}}{\@norm{#1}}{\@norm{#1}}{\@@@norm{#1}}
317   }

```

```

318}{\mm@warning{norm}} % bad result with libertineustlmath
319
320\newcommand{\tensor}{\mathbfsfit} % isomath uses \mathsfbfit
321

```

Classic identifiers are presented below.

```

322\mm@macro{di}{\mathop{\}\!\mathup{d}}
323\newcommand\probastyle{}
324\let\Par\P % end of paragraph symbol
325\renewcommand{\P}{\operatorname{\probastyle{P}}}
326\mm@macro{E}{\operatorname{\probastyle{E}}}
327\mm@macro{V}{\operatorname{\probastyle{V}}}
328
329\mm@operator{adj}{adj}
330\mm@operator{Aut}{Aut}
331\mm@operator{codim}{codim}
332\mm@operator{Conv}{Conv}
333\mm@operator{cov}{cov}
334\mm@operator{Cov}{Cov}
335\mm@macro{curl}{\operatorname{\vect{\mathup{curl}}}}
336\mm@operator{divg}{\divg}{div}
337\mm@operator{End}{End}
338
339\mm@operator{erf}{erf}
340\mm@macro{grad}{\operatorname{\vect{\mathup{grad}}}}
341\mm@operator{id}{id} % mathop or mathord?
342\mm@operator{Id}{Id}
343\mm@operator{im}{im}
344\let\oldIm\Im \renewcommand{\Im}{\operatorname{Im}}
345\mm@operator{lb}{lb}
346\mm@operator{lcm}{lcm}
347
348\mm@operator{rank}{rank}
349\let\oldRe\Re \renewcommand{\Re}{\operatorname{Re}}
350\mm@macro{rot}{\operatorname{\vect{\mathup{rot}}}}
351\mm@operator{sgn}{sgn}
352\mm@operator{sinc}{sinc}
353\mm@operator{spa}{\spa}{span}
354\mm@operator{tr}{tr}
355\mm@operator{var}{var}
356\mm@operator{Var}{Var}
357\mm@operator{Zu}{\Zu}{Z}
358
359\mm@operator{arccot}{arccot}
360\mm@operator{sech}{sech}
361\mm@operator{csch}{csch}
362\mm@operator{arsinh}{arsinh}
363\mm@operator{arcosh}{arcosh}
364\mm@operator{artanh}{artanh}
365\mm@operator{arcoth}{arcoth}

```

```

366 \mm@operator{\arsech}{arsech}
367 \mm@operator{\arcsch}{arcsch}
368
369 \mm@operator[bigO]{\bigO}{\mathcal{O}}
370 \mm@operator[bigO]{\bigO}{\mathcal{O}}
371 \mm@operator[lito]{\lito}{\mathcal{O}}
372

```

And finally we present the remaining macros.

With Cyrillic languages, the command `\C` may already be defined (only for text mode). Thus, it will not be redefined by `mismath`. However, one may still want to use our `\C` macro only for math mode without interfering the definition of the text `\C`, therefore the `\onlymathC` macro.

```

373 \mm@macro{mathset}{\mathbf}
374 \mm@macro{R}{\mathset{R}}
375 \mm@macro{C}{\mathset{C}}
376 \providecommand\onlymathC{\let\oldC\C
377   \renewcommand{\C}{\TextOrMath{\oldC}{\mathset{C}}}}
378 \mm@macro{N}{\mathset{N}}
379 \mm@macro{Z}{\mathset{Z}}
380 \mm@macro{Q}{\mathset{Q}}
381 \mm@macro{F}{\mathset{F}}
382 \mm@macro{K}{\mathset{K}}
383
384 \mm@macro{ds}{\displaystyle}
385 \mm@macro{dlim}{\lim\limits}
386 \mm@macro{dsum}{\sum\limits}
387 \mm@macro{dprod}{\prod\limits}
388 \mm@macro{dcup}{\bigcup\limits}
389 \mm@macro{dcap}{\bigcap\limits}
390
391 \mm@macro{lbar}{\overline}
392 \ifundefined{hlbar}{
393   \providecommand*\hlbar[1]{\overline{\vphantom{t}#1}}}
394   \mm@warning{hlbar} }
395 \newcommand\@eqdef{\stackrel{\mathup{def}}{=}}
396 \newcommand\@eqdef{\stackrel{\mathrm{\Delta}}{=}}
397 \mm@macro{eqdef}{\ifstar{\@eqdef}{\@eqdef}}
398 \mm@macro{unbr}{\underbrace}
399 \mm@macro{iif}{if and only if\hspace}
400

```

Above, we have used `\mathrm` before `\Delta` in case of defining capital Greek letters in italics (for example with the `fixmath` package).

The use of `\mbox{}` ensures that the space produced by `\` in the `\then` macro is not suppressed in tables.

```

401 \mm@macro{then}{\ \Longrightarrow \ \mbox{ } }
402 \ifundefined{txt}{
403   \providecommand*\txt[1]{\quad\text{#1}\quad} }

```



```

404 \mm@warning{txt} }
405 \mm@macro{mul}{\mathord{\times}}
406 \@ifundefined{pow}{
407   \providecommand*\pow{[2]{\left( #1 \right)^{\!#2}} }{
408   \mm@warning{pow} }
409 \@ifundefined{abs}{
410   \providecommand*\abs{[1]{\left\vert\!#1\right\vert} }{
411   \mm@warning{abs} }
412 \@ifundefined{lfrac}{
413   \providecommand*\lfrac{[2]{\frac{\!#1\!}{\!#2\!}} }{
414   \mm@warning{lfrac} }
415
416 \newcommand{\systemstretch}{1.2}
417 \newcommand{\systemsep}{\medspace}
418 \newenvironment{system}[1][1]{
419   \renewcommand{\arraystretch}{\systemstretch}
420   \setlength{\arraycolsep}{0.15em}
421   \left\{\begin{array}{@{\systemsep}#1@{}} %
422 }\end{array}\right.}
423
424 \newenvironment{spmatrix}{
425   \left(\begin{smallmatrix}
426 }\end{smallmatrix}\right)}
427
428 \newenvironment{mathcols}{% needs multicol package
429   \ifmm@multicol
430     \renewcommand{\columnseprule}{0.1pt}
431     \begin{multicols}{2}
432       \par\noindent\hfill
433       \begin{math}\begin{aligned}\displaystyle
434     \else
435       \PackageError{mismath}{The mathcols environment
436         needs the multicol package}{Add the package multicol
437         to your preamble.}
438     \fi
439   }{%
440     \end{aligned}\end{math} \hfill\mbox{}
441   \end{multicols}
442 }
443 \newcommand{\changeacol}{%
444   \end{aligned}\end{math} \hfill\mbox{}
445   \par\noindent\hfill
446   \begin{math}\begin{aligned}\displaystyle
447 }

```

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] *Typefaces for Symbols in Scientific Manuscripts*,
<https://www.physics.nist.gov/cuu/pdf/typefaces.pdf>.
- [3] *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>.
- [4] *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.
- [5] *esvect – Typesetting vectors with beautiful arrow with $\text{\LaTeX}2_{\epsilon}$* , Eddie Saudrais, CTAN, v1.3 2013/07/11.
- [6] *amsmath – $\mathcal{A}\mathcal{M}\mathcal{S}$ mathematical facilities for \LaTeX* , Frank Mittelbach, Rainer Schöpf, Michael Downes, Davis M. Jones, David Carlisle, CTAN, v2.17n 2022/04/08.
- [7] *Experimental Unicode mathematical typesetting: The unicode-math package*, Will Robertson, Philipp Stephani, Joseph Wright, Khaled Hosny, and others, CTAN, v0.8r 2023/08/13.
- [8] *The mathtools package*, Morten Høgholm, Lars Madsen, CTAN, v1.29 2022/06/29.
- [9] *The fixmath package for $\text{\LaTeX}2_{\epsilon}$* , Walter Schmidt, CTAN, v0.9 2000/04/11.
- [10] *isomath – Mathematical style for science and technology*, Günter Milde, CTAN, v0.6.1 2012/09/04.
- [11] *PM-ISOMath, The Poor Man ISO math bundle*, the pm-isomath package by Claudio Beccari, CTAN, v1.2.00 2021/08/04.
- [12] *The upgreek package for $\text{\LaTeX}2_{\epsilon}$* , Walter Schmidt, CTAN, v2.0 2003/02/12.
- [13] *The mathdesign package*, Paul Pichaureau, CTAN, v2.31 2013/08/29.
- [14] *The textalpha package* (part of the greek-fontenc bundle), Günter Milde, CTAN, v2.1 14/06/2022.
- [15] *Kp-Fonts – The Johannes Kepler project*, Christophe Caignaert, CTAN, v3.34 20/09/2022.
- [16] *Fourier-GUTenberg*, Michel Bovani, CTAN, v1.3 30/01/2005.
- [17] *The lgrmath package*, Jean-François B., CTAN, v1.0 2022/11/16.
- [18] *The interval package*, Lars Madsen, CTAN, v0.4 2019/03/06.

- [19] *The icomma package for L^AT_EX 2_ε*, Walter Schmidt, CTAN, v2.0 2002/03/10.
- [20] *The nccomma package*, Alexander I. Rozhenko, CTAN, v1.0 2005/02/10.
- [21] *The decimalcomma package*, Antoine Missier, CTAN, v1.4 2023/12/30.
- [22] *L'extension pour T_EX et L^AT_EX systeme*, Christian Tellechea, CTAN, v0.32 2019/01/13.
- [23] *The spalign package*, Joseph Rabinoff, CTAN, 2016/10/05.
- [24] *The package nicematrix*, François Pantigny, CTAN, v6.14 2023/02/18.
- [25] *L'extension frenchmath*, Antoine Missier, CTAN, v2.10 2024/02/25.
- [26] *Intelligent brackets – The ibrackets package*, Antoine Missier, CTAN, v1.2, 2023/07/26.
- [27] *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>.
- [28] *The L^AT_EX Companion*, Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley, 2nd edition, Pearson Education, 2004.