The mathfixs Package

Niklas Beisert

Institut für Theoretische Physik
Eidgenössische Technische Hochschule Zürich
Wolfgang-Pauli-Strasse 27, 8093 Zürich, Switzerland

nbeisert@itp.phys.ethz.ch

2024/10/25, v1.11

Abstract

mathfixs is a LaTeX 2ε package to fix some odd behaviour in math mode such as spacing around fractions and roots, math symbols within bold text as well as capital Greek letters. It also adds some related macros.

Contents

1 Introduction **1**

2 Usage **2**
- 2.1 Package Features . . . 2
- 2.2 Fractions . . . 2
- 2.3 Roots . . . 4
- 2.4 Space Representing Multiplication Signs . . . 5
- 2.5 Greek Letters . . . 5
- 2.6 Bold Fonts . . . 6
- 2.7 Mathematical Functions and Operators . . . 6
- 2.8 Particular Symbols . . . 9

3 Information **11**
- 3.1 Copyright . . . 11
- 3.2 Files and Installation . . . 11
- 3.3 Related CTAN Packages . . . 12
- 3.4 Revision History . . . 12

A Sample File **13**

B Implementation **16**

1 Introduction

Undoubtedly, TeX and LaTeX are excellent at producing visually appealing mathematical expressions. However, some commonly used macros produce output which sometimes appears unbalanced under certain conditions. Depending on the level of sophistication, such artefacts are typically either ignored or fixed by some manual adjustments wherever they appear. This package addresses some of the issues encountered commonly (by the author), and provides fixes or additional macros to handle such situations. At present the package is mainly concerned with:

- spacing of fractions, roots and symbols
- alternative versions of fractions
- representation of (capital) Greek letters
- math symbols within bold text

Additional fixes and features may be added to the package in later versions. All of these features can be selected and customised to some extent to accommodate for the desired style.

2 Usage

To use the package mathfixs add the command

```
\usepackage{mathfixs}
```

to the preamble of your LaTeX document. Furthermore you must select the desired features explicitly as described below, otherwise the package will have no effect. This is to avoid having to agree on all the provided features and to allow for forward compatibility: the package should remain open for future extensions which may modify the default behaviour in an unintended fashion.

2.1 Package Features

`\ProvideMathFix` Features and options can be selected by the commands:

`\usepackage[`*opts*`]{mathfixs}`
or `\PassOptionsToPackage{`*opts*`}{mathfixs}`
or `\ProvideMathFix{`*opts*`}`

`\PassOptionsToPackage` must be used before `\usepackage`; `\ProvideMathFix` must be used afterwards. Note that if `\ProvideMathFix` is invoked within a block, its definitions are valid only locally within the block. *opts* is a comma-separated list of features or options. The available features and options are described below.

2.2 Fractions

The package offers some improvements for the spacing of fractions as well as definitions to typeset small rational numbers.

`frac` **Feature `frac`.** In many situations, LaTeX typesets fractions with an insufficient amount of surrounding space. For example, the space between two consecutive fractions almost make them appear as a single one:

`x\frac{a+b}{c+d}\frac{e}{f}.` $\longrightarrow$ $x\frac{a+b}{c+d}\frac{e}{f}.$

Also the space towards other symbols and punctuation marks arguably is too small. To make expressions more legible, sequences of fractions are often coded with various custom spaces (‘\,’, ‘\:’, ‘\;’, ‘\ ’, ‘~’, etc.) inserted, e.g.

`x\,\frac{a+b}{c+d}\;\frac{e}{f}\ .` $\longrightarrow$ $$x\,\frac{a+b}{c+d}\;\frac{e}{f}\ .$$

This looks better, but requires a lot of intervention and depends very much on personal conventions.

The underlying technical reason for the shortcoming in spacing around fractions is that the latter are defined to have the math class of ordinary objects (`\mathord`) which results in no surrounding space. A simple resolution is to assign the math class of inner objects (`\mathinner`) to fractions as described in the TEXbook, e.g.

```
x\mathinner{\frac{a+b}{c+d}}\mathinner{\frac{e}{f}}.
```

Inner objects behave almost like ordinary objects, but some space is added between them and other inner or ordinary objects, e.g.:

$$x\mathinner{\frac{a+b}{c+d}}\mathinner{\frac{e}{f}}.$$

Importantly, no space is added between inner objects and the ends of the math expression or subexpression in parentheses. Also, no space is generated in the script styles (`\[script]scriptstyle`).

`\frac` The feature `frac` redefines the macro `\frac` such that all fractions have the inner math class producing some surrounding space in selected situations, e.g.:

`x\frac{a+b}{c+d}\frac{e}{f}.` $\longrightarrow$ $$x\mathinner{\frac{a+b}{c+d}}\mathinner{\frac{e}{f}}.$$

This makes adding custom space around fractions unnecessary in almost all situations, and leads to a rather uniform appearance without further adjustments.

`\genfrac` `\dfrac` `\tfrac` The macro `\genfrac` of the package amsmath is modified suitably when the latter is loaded. This also affects the macros `\dfrac` and `\tfrac` for fractions in styles `\displaystyle` and `\textstyle`, respectively. For `\genfrac` fractions with delimiters, the inner math class is not applied because the default math class is appropriate.

Note that when the `frac` feature is used, spaces around fractions can be eliminated by using:

`\mathord{\frac{`*num*`}{`*denom*`}}`

or simply (a block enclosed by braces `{...}` is considered an object with ordinary math class):

`{\frac{`*num*`}{`*denom*`}}`

Furthermore, note that unlike the original macro `\frac`, the redefined macro must be enclosed in a block when passed as an argument. For example, `x^\frac{1}{2}` produces an error and must be written as `x^{\frac{1}{2}}`.

`fracclass` `fracdelimclass` The option `fracclass=`*class* can be used to customise the math class of fractions to any desired command *class* accepting the expression for the fraction as its single parameter. Likewise, the option `fracdelimclass=`*class* customises the math class of fractions with delimiters generated by `\genfrac` such as `\[d|t]binom`.

rfrac **Feature rfrac.** Rational numbers as coefficients to some simple terms often stick out visually from displayed math expressions, e.g.:

$$\frac{1}{2}x^2 + \frac{1}{6}y^3 + \frac{1}{4}x^2y^2$$

Arguably, typesetting such fractions in text style has a more uniform appearance:

$$\tfrac{1}{2}x^2 + \tfrac{1}{6}y^3 + \tfrac{1}{4}x^2y^2$$

\rfrac The feature `rfrac` defines the macro `\rfrac` to typeset a fraction in text style or smaller. It is similar to `\tfrac` of the package amsmath, but it will not choose text style when in the script styles and it will not produce surrounding space when the feature `frac` is used. The optional argument `rfrac={\`*cmd*`}` specifies an alternative command name `\`*cmd* for `\rfrac`.

vfrac \vfrac **Feature vfrac.** Common or vulgar fractions such as $^1\!/_2$ can be typeset in a simple fashion as a superscript numerator followed by a slash and a subscript denominator. The spacing around the slash should be reduced to join the expression. The package provides this representation as the feature `vfrac` defining the macro `\vfrac`. It is similar to `\nicefrac` of the package nicefrac or the more advanced implementation `\sfrac` of the package xfrac. As vulgar fractions are often (mainly) used within plain text, the macro `\vfrac` also works in text mode. The optional argument `vfrac={\`*cmd*`}` specifies an alternative command name `\`*cmd* for `\vfrac`.

vfracclass vfracskippre vfracskippost The option `vfracclass=`*class* can be used to customise the math class of the vulgar fraction. The options `vfracskippre=`*muskip* and `vfracskippost=`*muskip* defines the (negative) skip around the slash (must be given in math units `mu`).

2.3 Roots

The TeX implementations of radicals, mostly those with exponents, have some uneven spacing, e.g.:

$$y\sqrt{x}z, \qquad y\sqrt{x}z, \qquad y\sqrt[i]{x}z, \qquad y\sqrt[n]{x}z, \qquad y\sqrt[3]{x}z, \qquad y\sqrt[123]{x}z$$

They work best if no exponent is specified or if the exponent is a single numerical digit. The spacing is slightly different if the exponent is n or i.

root \sqrt \root **Feature root.** The package provides an alternative mechanism for the placement of the exponent next to the radical. It first measures the extents of the exponent and the radical sign and then places them accordingly. It also adds some space at the end of the radicand. The refined definition produces the following representation without further spacing adjustments

$$y\sqrt{x}\,z, \qquad y\sqrt{x}\,z, \qquad y\sqrt[i]{x}\,z, \qquad y\sqrt[n]{x}\,z, \qquad y\sqrt[3]{x}\,z, \qquad y\sqrt[123]{x}\,z$$

rootclass rootskipend rootskippre rootskippost The option `rootclass=`*class* can be used to customise the math class of the root. The option `rootskipend=`*muskip* defines the additional skip at the end of the radicand within the radical (must be given in math units `mu`). The options `rootskippre=`*muskip* and `rootskippost=`*muskip* define additional skip around the radical.

rootclose The option `rootclose` adds a closing mark to the end of the top bar of radicals:

$$\sqrt{a}$$

The optional argument `rootclose=`*height* specifies the starting height with a default value of 0.8.

2.4 Space Representing Multiplication Signs

Mathematical expressions regularly omit explicit multiplication signs between factors, e.g. x^2yz. All is well unless some of the factors are compound expressions such as Δx or differentials dx (or $\mathrm{d}x$) where additional space is typically inserted by commands like '`\,`'.

A simple method to declare (or define) compound expressions with a suitable amount of surrounding space is to encapsulate them in a `\mathinner` block, e.g.

`12c\mathinner{\Delta x}\mathinner{\Delta y}z` $\longrightarrow$ $12c\mathinner{\Delta x}\mathinner{\Delta y}$

`\int x\mathinner{dx}` $\longrightarrow$ $\int x\mathinner{dx}$

Importantly, `\mathinner` will not add any space at the ends of an expression (or a subexpression in parentheses).

`multskip` `\.` **Feature `multskip`.** Furthermore, the package provides the feature `multskip` to encode a space representing an omitted multiplication sign by the short command '`\.`', e.g.:

`12c\.\Delta x\.\Delta y\.z` or `\int x\.dx`

The macro '`\.`' produces some amount of space in math mode (while retaining its original definition as an accent in text mode) and by default it is equivalent to '`\,`'. However, the macro '`\.`' is intended to describe this space unambiguously as a multiplication sign. In particular, the amount of space can be configured by the option `multskip=`*muskip* (given in math units `mu`), or the command could be customised further to a visually different representation.

2.5 Greek Letters

TEX defines Greek letters in math mode somewhat differently from Latin letters. The following two features redefine to the standard TEX Greek letters.

`greekcaps` **Feature `greekcaps`.** Capital Greek letters are declared as operators instead of plain letters, and therefore they are typeset in an upright shape. To typeset them in italic shape (as all the other letters representing variables) one can use `\mathnormal`, but this may be tedious if most capital Greek letters in a document actually represent variables.

The feature `greekcaps` redefines the 11 capital Greek letters `\Gamma`, `\Delta`, `\Theta`, `\Lambda`, `\Xi`, `\Pi`, `\Sigma`, `\Upsilon`, `\Phi`, `\Psi` and `\Omega` provided by TEX to have a default italic shape. Upright capital Greek letters remain accessible by switching to the upright shape using `\mathrm` or by declaring them as part of an operator, e.g. `\operatorname` or `\DeclareMathOperator` (package amsopt within amsmath).

The optional argument `greekcaps=`*pre* will instead define the macros `\`*pre*`Gamma`, etc., and keep the original definitions.

`greeklower` **Feature `greeklower`.** Lowercase Greek letters are fixed to the default math italic font because (unfortunately) no upright counterparts exist in the Compute Modern family of fonts.

The feature `greeklower` redefines the 23 lowercase Greek letters `\alpha`, ..., `\omega` as well as their 6 variants `\varepsilon`, `\vartheta`, `\varpi`, `\varphi`, `\varrho` and `\varsigma` defined by TEX to be elements of the regular math alphabet. This allows them to be typeset in different font series such as `\mathbold` described below. However, when trying to cast them in upright shape they will be typeset as altogether different symbols, e.g. `\mathrm{\alpha}` becomes the ligature 'ff' which occupies the same slot in the TEX font encodings OML vs. OT1. Hence, some caution is needed when this feature is used. Note that math font redefinitions may clash with this feature.

The optional argument `greeklower=`*pre* will instead define the macros `\`*pre*`alpha`, etc., and keep the original definitions.

2.6 Bold Fonts

Combining bold fonts and math mode in LATEX sometimes yields unintuitive results. The following two features assist the intuitive use of bold fonts in math mode.

autobold **Feature `autobold`.** Sectioning commands such as `\chapter`, `\section`, `\paragraph`, etc., but also `\maketitle` often change the font series to bold. However, any math symbols contained in the titles are typeset in the regular, non-bold series by default leading to a non-uniform appearance or calling for further manual adjustments using `\boldmath`. One might argue that it is bad practice to have math symbols in titles in the first place, but this point of view perhaps does not apply universally.

\bfseries \mdseries The feature `autobold` overwrites the LATEX font commands `\bfseries`, `\mdseries` as well as `\normalfont` to automatically switch the math fonts to their bold version. Derived commands such as `\section`, etc., will also switch the math fonts to the bold series.

mathbold \mathbold **Feature `mathbold`.** The feature `mathbold` defines a bold italic math font `\mathbold`. To typeset bold italic letters simply use `\mathbold{abc}`. To make this work for lowercase Greek letters as well, the feature `greeklower` must be activated. The optional argument `mathbold={\`*cmd*`}` specifies an alternative command name `\`*cmd* for `\mathbold`, e.g. `mathbold={\mathbit}`.

2.7 Mathematical Functions and Operators

LATEX defines many basic standard mathematical functions and operators, see below. The following features fill some gaps in the collection of functions and they provide worthwhile alternative representations for some functions and operators. All of the following features require that the package amsopn (part of amsmath) is loaded *before* mathfixs.

Native LATEX Functions and Operators. As a reference, LATEX supplies the following functions and operators. All of them are overwritten by amsopn within amsmath.

Exponential and logarithm functions:

macro	output	description
`\exp`	exp	exponential function
`\log`	log	logarithm function
`\lg`	lg	common logarithm function
`\ln`	ln	natural logarithm function

Trigonometric functions:

macro	output	description
`\sin`	sin	sine function
`\cos`	cos	cosine function
`\tan`	tan	tangent function
`\cot`	cot	cotangent function
`\sec`	sec	secant function
`\csc`	csc	cosecant function
`\arcsin`	arcsin	inverse sine function
`\arccos`	arccos	inverse cosine function
`\arctan`	arctan	inverse tangent function

Hyperbolic functions:

macro	output	description
`\sinh`	sinh	hyperbolic sine function
`\cosh`	cosh	hyperbolic cosine function
`\tanh`	tanh	hyperbolic tangent function
`\coth`	coth	hyperbolic cotangent function

Extremal values:

macro	output	description
`\max`	max	maximum of a set
`\min`	min	minimum of a set
`\sup`	sup	supremum of a set
`\inf`	inf	infimum of a set

Limits:

macro	output	description
`\lim`	lim	limit operator
`\limsup`	lim sup	limit superior operator
`\liminf`	lim inf	limit inferior operator
`\injlim`	inj lim	? (amsopn/amsmath only)
`\projlim`	proj lim	? (amsopn/amsmath only)

Assorted functions and operators:

macro	output	description
`\arg`	arg	argument of a complex number
`\det`	det	determinant of a matrix
`\ker`	ker	kernel of a map
`\dim`	dim	dimension of ...
`\deg`	deg	degree of ...
`\gcd`	gcd	greatest common divisor
`\hom`	hom	?
`\Pr`	Pr	?

Modulo statements:

macro	output	description
`\bmod`	$X \bmod Y$	modulo statement
`\pmod`	$X \pmod{Y}$	modulo statement in parentheses
`\mod`	$X \mod Y$	modulo statement with space (amsmath only)

genop
...
Feature `genop`. The package defines several additional mathematical functions, operators and symbols which are not included in the LaTeX macro library. These may be provided by other packages as well, or they can easily be defined by `\DeclareMathOperator` or used by `\operatorname`. Nevertheless, it is convenient to have these objects declared by a simple switch rather than by issuing a formal LaTeX clause.

The additional macros are grouped by category and each category is included as a feature. The first category describes assorted mathematical functions, operators and symbols. Further categories are provided in the following paragraphs.

In order to evade clashes, the package allows to individually select the desired definitions from the assorted section with optionally defined macro names:

feature	macro	output	description
`sgn`	`\sgn`	sgn	signum function
`res`	`\res`	res	residue operator
`lcm`	`\lcm`	lcm	least common multiple
`span`	`\Span`	span	span
`diag`	`\diag`	diag	diagonal matrix
`spec`	`\spec`	spec	spectrum
`const`	`\const`	const	anything constant
`genop`			all of the above

The following two-letter operators are not included in the above class and must be implemented individually:

feature	macro	output	description
`id`	`\id`	id	identity map
`tr`	`\tr`	tr	trace

reim
\Re
\Im
Feature `reim`. LaTeX assigns the symbols '$\Re$' and '$\Im$' to projectors to the real and imaginary parts of complex numbers. Another established representation for these projectors is given by 'Re' and 'Im'. The feature `reim` overwrites the predefined macros `\Re` and `\Im` with a textual representation:

macro	output	description
`\Re`	Re	real part of a complex number
`\Im`	Im	imaginary part of a complex number

The optional argument `reim={`*cmdr**cmdi*`}` specifies alternative command names *cmdr* and *cmdi* for `\Re` and `\Im`.

trig
\arccot
\arcsec
\arccsc
Feature `trig`. LaTeX defines most trigonometric functions and their inverses like sin, cos and arctan, but three of their inverses are omitted. The feature `trig` supplies the remaining three inverse trigonometric functions:

macro	output	description
`\arccot`	arccot	inverse cotangent function
`\arcsec`	arcsec	inverse secant function
`\arccsc`	arccsc	inverse cosecant function

hyp
\sech
\csch
\ar...h
Feature `hyp`. LaTeX defines four hyperbolic functions sinh, cosh and tanh, but two of them and all inverses are omitted. The feature `hyp` supplies the remaining two hyperbolic functions along with all inverse hypberbolic functions:

macro	output	description
`\sech`	sech	hyperbolic secant function
`\csch`	csch	hyperbolic cosecant function
`\arsinh`	arsinh	inverse hyperbolic sine function
`\arcosh`	arcosh	inverse hyperbolic cosine function
`\artanh`	artanh	inverse hyperbolic tangent function
`\arcoth`	arcoth	inverse hyperbolic cotangent function
`\arsech`	arsech	inverse hyperbolic secant function
`\arcsch`	arcsch	inverse hyperbolic cosecant function

`mapchar` `\dom` `\codom` `\supp` `\im` `\coker` `\rank`

Feature `mapchar`. The feature `mapchar` supplies further characteristics of maps:

macro	output	description
`\dom`	dom	domain
`\codom`	codom	codomain
`\supp`	supp	support
`\im`	im	image
`\coker`	coker	cokernel
`\rank`	rank	rank

The optional argument `mapchar={\`*cmdim*`}` specifies an alternative command name `\`*cmdim* for `\im` which has only two letters and might easily clash with other definitions.

`mapclass` `\Hom` `\End` `\Isom` `\Aut`

Feature `mapclass`. The feature `mapclass` supplies symbols for the sets of homomorphisms, endomorphisms, isomorphisms and automorphisms as classes/categories of structure-preserving maps:

macro	output	description
`\Hom`	Hom	homomorphisms
`\End`	End	endomorphisms
`\Isom`	Isom	isomorphisms
`\Aut`	Aut	automorphisms

`vecdiff` `vecrot` `\grad` `\div` `\curl`

Feature `vecdiff`. The feature `vecdiff` supplies the three common differential operators gradient, divergence and curl for vectors:

macro	output	description
`\grad`	grad	gradient
`\div`	div	divergence
`\curl`	curl (rot)	curl (alternative form)

The feature `vecrot`[`={\`*cmd*`}`] supplies the alternative form 'rot' for curl/rotor on top of `\curl` with the option to specify an alternative command name. The feature `lapl`, see below, supplies the Laplace operator.

2.8 Particular Symbols

The package provides a couple of standard mathematical symbols. Of course, users can be trusted to define these symbols themselves or, more likely, use them straight away. Nevertheless, here we go ...

econst iunit piconst **Mathematical Constants.** Three of the most relevant mathematical constants are Euler's number e, the imaginary unit i and the circle's constant π (with the relation $e^{i\pi} = -1$). The package provides macros for these with either upright or math italic representations:

feature	macro	output	description
`econst`	`\econst`	e	Euler's number (upright)
`econst*`	`\econst`	e	Euler's number (math italic)
`iunit`	`\iunit`	i	imaginary unit (upright)
`iunit*`	`\iunit`	i	imaginary unit (math italic)
`iunit*nb`	`\iunit`	$\mathring{\imath}$	imaginary unit (NB's silly circle version)
`piconst`	`\piconst`	π	circle constant π (upright, if only ...)
`piconst*`	`\piconst`	π	circle constant π (math italic)

Each macro and representation is provided as an individual feature *feature*[`={\`*cmd*`}`] which offers the option to customise the macro name to `\`*cmd*.

econstclass As Euler's number is commonly used within exponentiation which looks dense in compound expressions (e.g. $2e^{\sin x}F$), `\econst` is defined as `\mathinner` which adds some space around the exponentiation in such a situation (e.g. $2\,e^{\sin x}\,F$). Use `{\econst}` if such spacing is undesired. The option `econstclass=`*class* can be used to customise the math class of `\econst`.

An upright version of `\pi` needs to be supplied for the feature `piconst` to work properly. If the macro `\uppi` from the package upgreek in the bundle was is available at the time of loading, it will be used. Otherwise an upright `\pi` can be supplied by package unicode-math. Note that the feature `piconst` (without `*`) may interfere with the feature `greeklower`.

\der \diff **Derivative Symbols.** The package provides some assorted elementary symbols and compounds:

feature	macro	output	description
`der`	`\der`	d	derivative symbol (upright)
`der*`	`\der`	d	derivative symbol (math italic)
`diff`	`\diff{`x`}`	$\mathrm{d}x$	coordinate differential (upright)
`diff*`	`\diff{`x`}`	dx	coordinate differential (math italic)

Each macro and representation is provided as an individual feature *feature*[`={\`*cmd*`}`] which offers the option to customise the macro name to `\`*cmd*.

The coordinate differential `\diff` is meant to be used within integrals and as a differential form, and it can be used in the denominator of differentiation expressions. In all of these, it is understood as a compound which should be separated from surrounding symbols. Therefore `\diff{`x`}` is declared as `\mathinner` which adds spacing where needed.

\order \Order \lapl \defeq \eqdef **Assorted Symbols.** The package provides some assorted elementary symbols and compounds:

feature	macro	output	description
`order`	`\order`	o	anything of order less than
`Order`	`\Order`	O	anything of order at most
`Order*`	`\Order`	$\mathcal{O}$	anything of order at most (calligraphic)
`lapl`	`\lapl`	$\mathit{\Delta}$	Laplace operator
`defeq`	`\defeq`	$:=$	is defined as
`eqdef`	`\eqdef`	$=:$	defines

Each macro and representation is provided as an individual feature *feature*[={*cmd*}] which offers the option to customise the macro name to *cmd*.

The defining symbols `\defeq` and `\eqdef` both typeset the colon to be vertically aligned with the equation sign.

numset
numsetfont
\numset

Feature `numset`. The feature `numset` provides the macro `\numset` to typeset a number set such as the integers which is commonly denoted by a (blackboard) bold letter such as $\mathbb{Z}$ or $\mathbf{Z}$. The feature `numset={\`*cmd*`}` offers the option to customise the macro name to *cmd*. The default font for this feature is `\mathbb` of amssymb if available at the time of loading, otherwise `\mathbf`; it can be adjusted by `numsetfont={`*mathfont*`}`.

numsets

The standard number fields are provided by the feature `numsets` as follows:

macro	output	description
`\Natural`	$\mathbb{N}$	set of natural numbers
`\Integer`	$\mathbb{Z}$	field of integer numbers
`\Rational`	$\mathbb{Q}$	field of rational numbers
`\Real`	$\mathbb{R}$	field of real numbers
`\Complex`	$\mathbb{C}$	field of complex numbers
`\Quaternion`	$\mathbb{H}$	skew field of quaternions
`\Octonion`	$\mathbb{O}$	division algebra of quaternions

3 Information

3.1 Copyright

This work consists of the files `README.txt`, `mathfixs.ins` and `mathfixs.dtx` as well as the derived files `mathfixs.sty`, `mafxsamp.tex` and `mathfixs.pdf`.

3.2 Files and Installation

The package consists of the files:

`README.txt`	readme file
`mathfixs.ins`	installation file
`mathfixs.dtx`	source file
`mathfixs.sty`	package file
`mafxsamp.tex`	sample file
`mathfixs.pdf`	manual

The distribution consists of the files `README.txt`, `mathfixs.ins` and `mathfixs.dtx`.

- Run (pdf)LaTeX on `mathfixs.dtx` to compile the manual `mathfixs.pdf` (this file).

- Run LaTeX on `mathfixs.ins` to create the package `mathfixs.sty` and the sample `mafxsamp.tex`. Copy the file `mathfixs.sty` to an appropriate directory of your LaTeX distribution, e.g. *texmf-root*`/tex/latex/mathfixs`.

3.3 Related CTAN Packages

The package is related to other packages available at CTAN:

- This package uses the package keyval to process the options for the package, environments and macros. Compatibility with the keyval package has been tested with v1.15 (2014/10/28).
- This package is designed to be compatible with the package amsmath. To prevent amsmath from overwriting some features, it should be loaded *before* the present package. Compatibility with the amsmath package has been tested with v2.17a (2017/09/02).
- This package reproduces the functionality of the package fixmath v0.9 (2000/04/11) from the bundle was.
- Functionality to typeset common fractions is also provided by the packages nicefrac from the bundle units and xfrac offering a more advanced implementation.

3.4 Revision History

v1.11: 2024/10/25

- fixed misspelled definition `\End`

v1.1: 2024/10/23

- added feature `genop` for additional functions, operators and symbols
- added feature `reim` to change the definitions of `\Re` and `\Im` to 'Re' and 'Im'
- added features `trig` and `hyp` to complete the definitions of trigonometric and hypberbolic functions
- added feature `mapchar` to represent characteristics of maps
- added feature `mapclass` to represent structure-preserving maps
- added feature `vecdiff` to represent vector differential operators
- added representation for mathematical constants e, i, π
- added features `der` and `diff` to represent derivative symbols
- added assorted symbols
- added feature `numset` and `numset` to represent number sets

v1.01: 2018/12/30

- fix for `\vfrac` in aligned math

v1.0: 2018/01/17

- first version published on CTAN

A Sample File

In this section we provide an example of how to use some of the mathfixs features. We also test the behaviour in some special cases.

Preamble. Standard document class:

```
\documentclass[12pt]{article}
```

Use package geometry to adjust the page layout, adjust the paragraph shape:

```
\usepackage{geometry}
\geometry{layout=a4paper}
\geometry{paper=a4paper}
\geometry{margin=3cm}
\parindent0pt
```

Include amsmath, amssymb and the mathfixs package:

```
\RequirePackage{amsmath,amssymb}
\RequirePackage[autobold]{mathfixs}
```

Declare features to be implemented globally:

```
\ProvideMathFix{greekcaps,mathbold}
\ProvideMathFix{frac,rfrac,vfrac}
\ProvideMathFix{multskip}
\ProvideMathFix{der,diff}
```

Start document body:

```
\begin{document}
```

Fractions.

```
\paragraph{Fractions.}
```

Fraction spacing:

```
\[
x\frac{a+b}{c+d}\frac{e}{f}.
\]
```

Vulgar fractions:

```
Recipe for $\approx 3$ pancakes (scale accordingly):
Whisk together one large egg, $\vfrac{1}{8} \ell$ milk,
$50\mathinner{\mathrm{g}}$ flour
(mix \vfrac{2}{3} wholemeal flour and \vfrac{1}{3} white flour); fry in pan.
Feel free to use more flour than indicated.
```

Rational numbers:

```
\[
\rfrac{1}{2}x^2 + \rfrac{1}{6} y^3 + \rfrac{1}{4} x^2 y^2
\]
```

Radicals.

```
\paragraph{Radicals.}
```

Original radicals spacing:

```
\[
y\sqrt{x}z,\qquad
y\sqrt[]{x}z,\qquad
y\sqrt[i]{x}z,\qquad
y\sqrt[n]{x}z,\qquad
y\sqrt[3]{x}z,\qquad
y\sqrt[123]{x}z
\]
```

Radicals spacing (feature introduced in document body):

```
\ProvideMathFix{root}
\[
y\sqrt{x}z,\qquad
y\sqrt[]{x}z,\qquad
y\sqrt[i]{x}z,\qquad
y\sqrt[n]{x}z,\qquad
y\sqrt[3]{x}z,\qquad
y\sqrt[123]{x}z
\]
```

Radicals with closing mark (using a locally defined option):

```
\[
\ProvideMathFix{rootclose}
y\sqrt{x}z,\qquad
y\sqrt[]{x}z,\qquad
y\sqrt[i]{x}z,\qquad
y\sqrt[n]{x}z,\qquad
y\sqrt[3]{x}z,\qquad
y\sqrt[123]{x}z
\]
```

Radicals with class `\mathinner`:

```
\[
\ProvideMathFix{rootclass={\mathinner}}
y\sqrt{x}z,\qquad
y\sqrt[]{x}z,\qquad
y\sqrt[i]{x}z,\qquad
y\sqrt[n]{x}z,\qquad
y\sqrt[3]{x}z,\qquad
y\sqrt[123]{x}z
\]
```

Spaces Representing Multiplication and Derivative Symbols.

```
\paragraph{Spaces Representing Multiplication and Derivative Symbols.}
```

Explicit spacing:

```
\[
\int x\.\der x
\]
```

Implicit spacing:

```
\[
\int x\diff{x}
\]
```

Greek Letters.

```
\paragraph{Greek Letters.}
```

Capital letters:

```
\[
\Gamma \qquad
\mathrm{\Gamma} \qquad
c\operatorname{\Gamma}(x)
\]
```

Bold Text with Math Symbols.

```
\paragraph{Bold Text with Math Symbols: $abc123\alpha\Gamma$.}
```

Bold text:

```
\[
aG\alpha\Gamma
\qquad\mathbold{aG\alpha\Gamma}
\qquad\ProvideMathFix{greeklower}\mathbold{aG\alpha\Gamma}
\]
```

Real and Imaginary Parts.

```
\paragraph{Real and Imaginary Parts.}
```

Change representation for `\Re` and `\Im`:

```
\[
\Re(z),\Im(z)
\qquad\ProvideMathFix{reim}
\Re(z),\Im(z)
\qquad\ProvideMathFix{reim={\nRe\nIm}}
\nRe(z),\nIm(z)
\]
```

Functions, Operators, Symbols.

```
\paragraph{Functions, Operators, Symbols.}
```

Some examples of vector differential operators:

```
\[
\ProvideMathFix{vecdiff,lapl,defeq}
\lapl f=\div\grad f,
\qquad
\lapl v=\grad\div v-\curl\curl v,
\]
```

Some examples of mathematical constants:

```
\[
\ProvideMathFix{econst,iunit,piconst}
\econst^{\iunit\piconst}=-1,
\qquad
\ProvideMathFix{econst*,iunit*,piconst*}
\econst^{\iunit\piconst}=-1,
\qquad
\ProvideMathFix{econst,defeq,iunit*nb,numsets}
```

```
z\defeq x+\iunit y=r\econst^{\iunit\theta}\in\Complex,
\quad x,y,r,\theta\in\Real
\]
```

Some examples of operators and symbols:

```
\[
\ProvideMathFix{res,hyp,iunit*,piconst*}
\res_{z=\iunit \piconst n} \csch(z)=(-1)^n
\]
```

End of document body:

```
\end{document}
```

B Implementation

In this section we describe the package `mathfixs.sty`.

Required Packages. The package loads the package keyval if not yet present. keyval is used for extended options processing:

```
\RequirePackage{keyval}
```

Automatically Bold Maths.

`\bfseries` `\mdseries` `\normalfont` Define replacements for `\bfseries`, `\mdseries` and `\normalfont` by appending `\boldmath` or `\unboldmath` (if not already in math mode where font selection works differently). The chain of `\expandafter` directives fills in the original definition:

```
\expandafter\def\expandafter\mafx@bfseries\expandafter
  {\bfseries\ifmmode\else\boldmath\fi}
\expandafter\def\expandafter\mafx@mdseries\expandafter
  {\mdseries\ifmmode\else\unboldmath\fi}
\expandafter\def\expandafter\mafx@normalfont\expandafter
  {\normalfont\ifmmode\else\unboldmath\fi}
```

`autobold` Implement the `autobold` fix:

```
\define@key{mafx@}{autobold}[]{%
  \let\bfseries=\mafx@bfseries
  \let\mdseries=\mafx@mdseries
  \let\normalfont=\mafx@normalfont
}
```

Bold Italic Math.

`mathbold` `\mathbold` Define a new math alphabet `\mathbold` as the bold series and italic shape of the standard computer modern math font (see package fixmath):

```
\DeclareMathAlphabet{\mafx@mathbold}{OML}{cmm}{b}{it}
```

`autobold` Implement `\mathbold` (or alterantive macro name):

```
\define@key{mafx@}{mathbold}[\mathbold]{\let#1=\mafx@mathbold}
```

Spacing Adjustment for Fractions. Save primitive definition of `\over` into `\@@over` (compatible with amsmath):

```
\ifdefined\@@over\else\let\@@over=\over\fi
```

Define math class selectors for fractions (without/with delimiters):

```
\let\mafx@frac@class=\mathinner
\def\mafx@frac@delimclass{\mathopen{}\mathclose}
```

`\frac` Define replacement for `\frac` which uses a configurable math class selector:

```
\DeclareRobustCommand{\mafx@frac}[2]{%
    \mafx@frac@class{\begingroup#1\endgroup\@@over#2}}
```

`\genfrac` Define replacement for `\@genfrac` (called by `\genfrac`) which uses the appropriate math class selector. If the package amsmath is not loaded, this definition will have no impact:

```
\def\mafx@@genfrac#1#2#3#4#5{\begingroup
  \ifx#2\@@overwithdelims\let\mafx@frac@class\mafx@frac@delimclass\fi
  \ifx#2\@@abovewithdelims\let\mafx@frac@class\mafx@frac@delimclass\fi
  \ifx#2\@@atopwithdelims\let\mafx@frac@class\mafx@frac@delimclass\fi
  \mafx@frac@class{#1{\begingroup#4\endgroup#2#3\relax#5}}%
  \endgroup}
```

`frac` Implement the replacement for `\frac` and `\genfrac`:

```
\define@key{mafx@}{frac}[]{%
  \let\frac=\mafx@frac
  \let\@genfrac=\mafx@@genfrac
}
```

`fracclass`
`fracdelimclass` Configure the math classes for fractions (without/with delimiters):

```
\define@key{mafx@}{fracclass}{\def\mafx@frac@class{#1}}
\define@key{mafx@}{fracdelimclass}{\def\mafx@frac@delimclass{#1}}
```

Small Fractions.

`\rfrac` Define a fraction in text style or smaller using the ordinary math class (no surrounding space), e.g.: $\frac{1}{2}$:

```
\DeclareRobustCommand{\mafx@rfrac}[2]{{\mathchoice
  {\textstyle{\begingroup#1\endgroup\@@over#2}}%
  {\begingroup#1\endgroup\@@over#2}%
  {\begingroup#1\endgroup\@@over#2}%
  {\begingroup#1\endgroup\@@over#2}}}
```

`rfrac` Implement `\rfrac` (or alternative macro name):

```
\define@key{mafx@}{rfrac}[\rfrac]{\let#1=\mafx@rfrac}
```

Vulgar Fractions. Define the math class and spacing parameters as (negative) spaces around the dividing slash:

```
\let\mafx@vfrac@class=\mathinner
\def\mafx@vfrac@preskip{\thinmuskip}
\def\mafx@vfrac@postskip{0.6667\thinmuskip}
```

\vfrac Define a vulgar representation of a rational number, e.g.: $^1\!/_2$. Automatically switch to math mode if in text mode:

```
\DeclareRobustCommand{\mafx@vfrac}[2]{\ifmmode%
  \mafx@vfrac@class{\textstyle%
    ^{#1}\mkern-\mafx@vfrac@preskip/\mkern-\mafx@vfrac@postskip_{#2}}%
  \else$\mafx@vfrac{#1}{#2}$\fi}
```

vfrac Implement \vfrac (or alternative macro name):

```
\define@key{mafx@}{vfrac}[\vfrac]{\let#1=\mafx@vfrac}
```

vfracclass vfracskippre vfracskippost Define configurable skip parameters for \vfrac:

```
\define@key{mafx@}{vfracclass}{\def\mafx@vfrac@class{#1}}
\define@key{mafx@}{vfracskippre}{\def\mafx@vfrac@preskip{#1}}
\define@key{mafx@}{vfracskippost}{\def\mafx@vfrac@postskip{#1}}
```

Spacing Adjustment for Roots. Define some parameters for the improved root macros. \mafx@root@close stores the relative height where the closing bar for the radical starts; empty disables the closing bar. \mafx@root@class stores the math class for a root. \mafx@root@endskip stores space to be added after the radicant but within the radical in math units. \mafx@root@preskip and \mafx@root@postskip store space to be added before or after the radical sign:

```
\let\mafx@root@close=\@empty
\def\mafx@root@class{}
\def\mafx@root@endskip{0.6667\thinmuskip}
\def\mafx@root@preskip{0mu}
\def\mafx@root@postskip{0.3333\thinmuskip}
```

\sqrt Replacement for \sqrt which passes an empty first argument instead of calling \sqrtsign:

```
\DeclareRobustCommand\mafx@sqrt{\@ifnextchar[\@sqrt{\@sqrt[]}}
\def\mafx@@sqrt[#1]{\root#1\of}
```

\root Replacement for \root ... \of which stores the \leftroot and \uproot parameters specified in the exponent (see package amsmath). It does so by first storing them in global parameters and then converting them to local ones to avoid interference with nested radicals. As usual, the macro then passes on to \r@@t with \mathpalette:

```
\def\mafx@root#1\of{%
  \setbox\rootbox\hbox{$\m@th\scriptscriptstyle{%
    \gdef\mafx@gleftroot{0}\gdef\mafx@guproot{0}%
    \def\leftroot##1{\gdef\mafx@gleftroot{##1}}%
    \def\uproot##1{\gdef\mafx@guproot{##1}}%
    #1}$}%
  \let\mafx@leftroot=\mafx@gleftroot
  \let\mafx@uproot=\mafx@guproot
  \mathpalette\r@@t}
```

The replacement for \r@@t typesets the radical by placing the exponent in an elevated box. First, select desirect math class:

```
\def\mafx@r@@t#1#2{\mafx@root@class{%
```

Determine the font selector for the current style:

```
  \ifx#1\scriptstyle\let\mafx@tmp@fontsel=\scriptfont\else
```

```
  \ifx#1\scriptscriptstyle\let\mafx@tmp@fontsel=\scriptscriptfont\else
  \let\mafx@tmp@fontsel=\textfont\fi\fi
```

Generate a radical with empty radicand in cramped style (see package mathtools) at the desired height and depth and store in a box for subsequent measurements:

```
  \sbox\z@{$\m@th#1\nulldelimiterspace=\z@\radical\z@{#2}$}%
  \setlength\dimen@{\ht\z@}%
  \ifx#1\displaystyle
    \addtolength\dimen@{-\fontdimen8\textfont3}%
    \addtolength\dimen@{-0.25\fontdimen5\textfont2}%
  \else
    \addtolength\dimen@{-1.25\fontdimen8\mafx@tmp@fontsel3}%
  \fi
  \setbox\z@=\hbox{$\m@th#1\sqrtsign{%
    \vrule width0pt height\dimen@ depth\dp\z@}$}%
```

Shift to start of exponent box such that end will reside at 60% of radical sign. Do not shift if exponent box is sufficiently wide:

```
  \setlength\dimen@{-\wd\rootbox}%
  \addtolength\dimen@{0.6\wd\z@}%
  \ifdim\dimen@>0pt\else\setlength\dimen@{0pt}\fi%
  \kern\dimen@%
```

Compute elevation of exponent box as 60% of radical sign. Add length specified by `\uproot`:

```
  \setlength\dimen@{0.6\ht\z@}\addtolength\dimen@{-0.6\dp\z@}%
  \setbox\@ne\hbox{$\m@th#1\mskip\mafx@uproot mu$}%
  \addtolength\dimen@{\wd\@ne}%
```

Place exponent box and return to intended start of radical sign:

```
  \mkern-\mafx@leftroot mu%
  \raise\dimen@\copy\rootbox
  \mkern\mafx@leftroot mu%
  \kern-0.6\wd\z@%
```

Place radical adding extra kernings `\mafx@root@preskip` and `\mafx@root@endskip`:

```
  \mkern\mafx@root@preskip\sqrtsign{#2\mskip\mafx@root@endskip}%
```

Add a closing bar to the radical, see `http://en.wikibooks.org/wiki/LaTeX/Mathematics`. If `\mafx@root@close` is empty, ignore. Determine the thickness of the rule as font dimension x8. Draw a vertical rule starting from relative height `\mafx@root@close` of the radical:

```
  \ifx\mafx@root@close\@empty\else
  \setlength\dimen@{\fontdimen8\mafx@tmp@fontsel3}%
  \lower\dimen@\hbox{%
    \vrule width\dimen@ height\ht\z@ depth -\mafx@root@close\ht\z@}%
  \fi%
```

Finally, add kerning `\mafx@root@postskip`:

```
  \mkern\mafx@root@postskip}}
```

`root` Implement replacement `\sqrt` and `\root`:

```
\define@key{mafx@}{root}[]{%
  \let\sqrt=\mafx@sqrt
  \let\@sqrt=\mafx@@sqrt
  \let\root=\mafx@root
  \let\r@@t=\mafx@r@@t
```

```
}
```

rootclose rootclass rootskipend rootskippre rootskipbefore

Define configurable parameters for alternative root:

```
\define@key{mafx@}{rootclose}[0.8]{\def\mafx@root@close{#1}}
\define@key{mafx@}{rootclass}{\let\mafx@root@class=#1}
\define@key{mafx@}{rootskipend}{\def\mafx@root@endskip{#1}}
\define@key{mafx@}{rootskippre}{\def\mafx@root@preskip{#1}}
\define@key{mafx@}{rootskippost}{\def\mafx@root@postskip{#1}}
```

Space Representing Multiplication.

\\. Define \\. to represent a thin math skip when in math mode. Retain original definition as an accent in text mode. Switch is performed by temporarily storing the appropriate macro which is subsequently issued so that the correct number of arguments are fetched:

```
\let\mafx@old@dot=\.
\def\mafx@dot@skip{\thinmuskip}
\def\mafx@dot{\mskip\mafx@dot@skip}
\def\mafx@per@dot{\begingroup\ifmmode\def\mafx@tmp{\mafx@dot}\else
  \def\mafx@tmp{\mafx@old@dot}\fi\expandafter\endgroup\mafx@tmp}
```

multskip Implement definition of \\.:

```
\define@key{mafx@}{multskip}[\thinmuskip]{%
  \let\.=\mafx@per@dot
  \def\mafx@dot@skip{#1}%
}
```

Italic Capital Greek Letters.

\Gamma ... Define capital Greek letters as letters (instead of the default assignment as operators). This implementation follows the package fixmath:

```
\DeclareMathSymbol{\mafx@Gamma}{\mathalpha}{letters}{0}
\DeclareMathSymbol{\mafx@Delta}{\mathalpha}{letters}{1}
\DeclareMathSymbol{\mafx@Theta}{\mathalpha}{letters}{2}
\DeclareMathSymbol{\mafx@Lambda}{\mathalpha}{letters}{3}
\DeclareMathSymbol{\mafx@Xi}{\mathalpha}{letters}{4}
\DeclareMathSymbol{\mafx@Pi}{\mathalpha}{letters}{5}
\DeclareMathSymbol{\mafx@Sigma}{\mathalpha}{letters}{6}
\DeclareMathSymbol{\mafx@Upsilon}{\mathalpha}{letters}{7}
\DeclareMathSymbol{\mafx@Phi}{\mathalpha}{letters}{8}
\DeclareMathSymbol{\mafx@Psi}{\mathalpha}{letters}{9}
\DeclareMathSymbol{\mafx@Omega}{\mathalpha}{letters}{10}
```

greekcaps Implement italic capital Greek letters. The optional argument is used as a prefix for the definitions:

```
\define@key{mafx@}{greekcaps}[]{%
  \expandafter\let\csname #1Gamma\endcsname=\mafx@Gamma
  \expandafter\let\csname #1Delta\endcsname=\mafx@Delta
  \expandafter\let\csname #1Theta\endcsname=\mafx@Theta
  \expandafter\let\csname #1Lambda\endcsname=\mafx@Lambda
  \expandafter\let\csname #1Xi\endcsname=\mafx@Xi
  \expandafter\let\csname #1Pi\endcsname=\mafx@Pi
  \expandafter\let\csname #1Sigma\endcsname=\mafx@Sigma
```

```
  \expandafter\let\csname #1Upsilon\endcsname=\mafx@Upsilon
  \expandafter\let\csname #1Phi\endcsname=\mafx@Phi
  \expandafter\let\csname #1Psi\endcsname=\mafx@Psi
  \expandafter\let\csname #1Omega\endcsname=\mafx@Omega
}
```

Lowercase Greek Letters in Standard Math Type.

`\alpha` ... Define lowercase Greek letters in standard type `\mathalpha` (instead of the default assiment `\mathord`). This implementation follows the package fixmath:

```
\DeclareMathSymbol{\mafx@alpha}{\mathalpha}{letters}{11}
\DeclareMathSymbol{\mafx@beta}{\mathalpha}{letters}{12}
\DeclareMathSymbol{\mafx@gamma}{\mathalpha}{letters}{13}
\DeclareMathSymbol{\mafx@delta}{\mathalpha}{letters}{14}
\DeclareMathSymbol{\mafx@epsilon}{\mathalpha}{letters}{15}
\DeclareMathSymbol{\mafx@zeta}{\mathalpha}{letters}{16}
\DeclareMathSymbol{\mafx@eta}{\mathalpha}{letters}{17}
\DeclareMathSymbol{\mafx@theta}{\mathalpha}{letters}{18}
\DeclareMathSymbol{\mafx@iota}{\mathalpha}{letters}{19}
\DeclareMathSymbol{\mafx@kappa}{\mathalpha}{letters}{20}
\DeclareMathSymbol{\mafx@lambda}{\mathalpha}{letters}{21}
\DeclareMathSymbol{\mafx@mu}{\mathalpha}{letters}{22}
\DeclareMathSymbol{\mafx@nu}{\mathalpha}{letters}{23}
\DeclareMathSymbol{\mafx@xi}{\mathalpha}{letters}{24}
\DeclareMathSymbol{\mafx@pi}{\mathalpha}{letters}{25}
\DeclareMathSymbol{\mafx@rho}{\mathalpha}{letters}{26}
\DeclareMathSymbol{\mafx@sigma}{\mathalpha}{letters}{27}
\DeclareMathSymbol{\mafx@tau}{\mathalpha}{letters}{28}
\DeclareMathSymbol{\mafx@upsilon}{\mathalpha}{letters}{29}
\DeclareMathSymbol{\mafx@phi}{\mathalpha}{letters}{30}
\DeclareMathSymbol{\mafx@chi}{\mathalpha}{letters}{31}
\DeclareMathSymbol{\mafx@psi}{\mathalpha}{letters}{32}
\DeclareMathSymbol{\mafx@omega}{\mathalpha}{letters}{33}
\DeclareMathSymbol{\mafx@varepsilon}{\mathalpha}{letters}{34}
\DeclareMathSymbol{\mafx@vartheta}{\mathalpha}{letters}{35}
\DeclareMathSymbol{\mafx@varpi}{\mathalpha}{letters}{36}
\DeclareMathSymbol{\mafx@varphi}{\mathalpha}{letters}{39}
\DeclareMathSymbol{\mafx@varrho}{\mathalpha}{letters}{37}
\DeclareMathSymbol{\mafx@varsigma}{\mathalpha}{letters}{38}
```

`greeklower` Implement standard type lowercase Greek letters. The optional argument is used as a prefix for the definitions:

```
\define@key{mafx@}{greeklower}[]{%
  \expandafter\let\csname #1alpha\endcsname=\mafx@alpha
  \expandafter\let\csname #1beta\endcsname=\mafx@beta
  \expandafter\let\csname #1gamma\endcsname=\mafx@gamma
  \expandafter\let\csname #1delta\endcsname=\mafx@delta
  \expandafter\let\csname #1epsilon\endcsname=\mafx@epsilon
  \expandafter\let\csname #1zeta\endcsname=\mafx@zeta
  \expandafter\let\csname #1eta\endcsname=\mafx@eta
  \expandafter\let\csname #1theta\endcsname=\mafx@theta
  \expandafter\let\csname #1iota\endcsname=\mafx@iota
  \expandafter\let\csname #1kappa\endcsname=\mafx@kappa
  \expandafter\let\csname #1lambda\endcsname=\mafx@lambda
  \expandafter\let\csname #1mu\endcsname=\mafx@mu
  \expandafter\let\csname #1nu\endcsname=\mafx@nu
```

```
  \expandafter\let\csname #1xi\endcsname=\mafx@xi
  \expandafter\let\csname #1pi\endcsname=\mafx@pi
  \expandafter\let\csname #1rho\endcsname=\mafx@rho
  \expandafter\let\csname #1sigma\endcsname=\mafx@sigma
  \expandafter\let\csname #1tau\endcsname=\mafx@tau
  \expandafter\let\csname #1upsilon\endcsname=\mafx@upsilon
  \expandafter\let\csname #1phi\endcsname=\mafx@phi
  \expandafter\let\csname #1chi\endcsname=\mafx@chi
  \expandafter\let\csname #1psi\endcsname=\mafx@psi
  \expandafter\let\csname #1omega\endcsname=\mafx@omega
  \expandafter\let\csname #1varepsilon\endcsname=\mafx@varepsilon
  \expandafter\let\csname #1vartheta\endcsname=\mafx@vartheta
  \expandafter\let\csname #1varpi\endcsname=\mafx@varpi
  \expandafter\let\csname #1varphi\endcsname=\mafx@varphi
  \expandafter\let\csname #1varrho\endcsname=\mafx@varrho
  \expandafter\let\csname #1varsigma\endcsname=\mafx@varsigma
}
```

Assorted Functions, Operators, Symbols. Define macros for assorted functions, operators, symbols:

```
\DeclareMathOperator{\mafx@sgn}{sgn}
\DeclareMathOperator*{\mafx@res}{res}
\DeclareMathOperator{\mafx@lcm}{lcm}
\DeclareMathOperator{\mafx@span}{span}
\DeclareMathOperator{\mafx@diag}{diag}
\DeclareMathOperator{\mafx@spec}{spec}
\DeclareMathOperator{\mafx@const}{const}
\DeclareMathOperator{\mafx@id}{id}
\DeclareMathOperator{\mafx@tr}{tr}
```

Implement assorted functions, operators and symbols individually with the option to adjust their macro names:

```
\define@key{mafx@}{sgn}[\sgn]{\let#1=\mafx@sgn}
\define@key{mafx@}{res}[\res]{\let#1=\mafx@res}
\define@key{mafx@}{lcm}[\lcm]{\let#1=\mafx@lcm}
\define@key{mafx@}{diag}[\diag]{\let#1=\mafx@diag}
\define@key{mafx@}{span}[\Span]{\let#1=\mafx@span}
\define@key{mafx@}{spec}[\spec]{\let#1=\mafx@spec}
\define@key{mafx@}{const}[\const]{\let#1=\mafx@const}
\define@key{mafx@}{id}[\id]{\let#1=\mafx@id}
\define@key{mafx@}{tr}[\tr]{\let#1=\mafx@tr}
```

`genop` Implement all assorted functions, operators and symbols:

```
\define@key{mafx@}{genop}[]{%
  \ProvideMathFix{sgn,res,lcm,diag,span,spec,const}%
}
```

Real and Imaginary Part Projectors in Text Form.

`\Re` `\Im` Define a different notation for real and imaginary part projectors are the textual symbols ‘Re’ and ‘Im’:

```
\DeclareMathOperator{\mafx@Re}{Re}
\DeclareMathOperator{\mafx@Im}{Im}
```

reim Implement textual notation for \Re and \Im:

```
\define@key{mafx@}{reim}[\Re\Im]{%
  \expandafter\let\@firstoftwo#1=\mafx@Re
  \expandafter\let\@secondoftwo#1=\mafx@Im
}
```

Remaining Inverse Trigonometric Functions.

\arccot \arcsec \arccsc Define remaining inverse trigonometric functions and their inverses:

```
\DeclareMathOperator{\mafx@arccot}{arccot}
\DeclareMathOperator{\mafx@arcsec}{arcsec}
\DeclareMathOperator{\mafx@arccsc}{arccsc}
```

trig Implement remaining trigonometric functions:

```
\define@key{mafx@}{trig}[]{%
  \let\arccot=\mafx@arccot
  \let\arcsec=\mafx@arcsec
  \let\arccsc=\mafx@arccsc
}
```

Remaining Hyperbolic Functions.

\sech \csch \ar...h Define remaining hyperbolic functions and their inverses:

```
\DeclareMathOperator{\mafx@sech}{sech}
\DeclareMathOperator{\mafx@csch}{csch}
\DeclareMathOperator{\mafx@arsinh}{arsinh}
\DeclareMathOperator{\mafx@arcosh}{arcosh}
\DeclareMathOperator{\mafx@artanh}{artanh}
\DeclareMathOperator{\mafx@arcoth}{arcoth}
\DeclareMathOperator{\mafx@arsech}{arsech}
\DeclareMathOperator{\mafx@arcsch}{arcsch}
```

hyp Implement remaining hyperbolic functions:

```
\define@key{mafx@}{hyp}[]{%
  \let\sech=\mafx@sech
  \let\csch=\mafx@csch
  \let\arsinh=\mafx@arsinh
  \let\arcosh=\mafx@arcosh
  \let\artanh=\mafx@artanh
  \let\arcoth=\mafx@arcoth
  \let\arsech=\mafx@arsech
  \let\arcsch=\mafx@arcsch
}
```

Characteristics of Maps. Define macros for characteristics of maps:

```
\DeclareMathOperator{\mafx@dom}{dom}
\DeclareMathOperator{\mafx@supp}{supp}
\DeclareMathOperator{\mafx@codom}{codom}
\DeclareMathOperator{\mafx@im}{im}
\DeclareMathOperator{\mafx@rank}{rank}
\DeclareMathOperator{\mafx@coker}{coker}
```

Implement characteristics of maps with the option to adjust the macro name for `\im` to evade potential clashes for two letter definition:

```
\define@key{mafx@}{mapchar}[\im]{%
  \let\dom=\mafx@dom
  \let\supp=\mafx@supp
  \let\codom=\mafx@codom
  \let#1=\mafx@im
  \let\rank=\mafx@rank
  \let\coker=\mafx@coker
}
```

Classes of Structure-Preserving Maps.

`\Hom` `\End` `\Isom` `\Aut` Define classes of structure-preserving maps:

```
\DeclareMathOperator{\mafx@Hom}{Hom}
\DeclareMathOperator{\mafx@End}{End}
\DeclareMathOperator{\mafx@Isom}{Isom}
\DeclareMathOperator{\mafx@Aut}{Aut}
```

`mapclass` Implement classes of structure-preserving maps:

```
\define@key{mafx@}{mapclass}[]{%
  \let\Hom=\mafx@Hom
  \let\End=\mafx@End
  \let\Isom=\mafx@Isom
  \let\Aut=\mafx@Aut
}
```

Differential Operators for Vectors.

`\grad` `\div` `\curl` Define differential operators gradient, divergence and curl (alternative form rot):

```
\DeclareMathOperator{\mafx@grad}{grad}
\DeclareMathOperator{\mafx@div}{div}
\DeclareMathOperator{\mafx@curl}{curl}
\DeclareMathOperator{\mafx@rot}{rot}
```

`vecdiff` Implement differential operators for vectors individually with the option to adjust their macro names. Implement alternative rot for curl/rotor with the option to adjust its macro name:

```
\define@key{mafx@}{vecdiff}[]{%
  \let\grad=\mafx@grad
  \let\div=\mafx@div
  \let\curl=\mafx@curl
}
\define@key{mafx@}{vecrot}[\curl]{\let#1=\mafx@rot}
```

Mathematical Constants.

`\econst` `\iunit` `\piconst` Define macros for mathematical constants. Use `\uppi` for upright `\pi` if available at the time of loading, otherwise use plain `\pi` hoping for availability of an upright version:

```
\let\mafx@econst@class=\mathinner
\DeclareRobustCommand{\mafx@econstup}{\mafx@econst@class{\operator@font e}}
```

```
\DeclareRobustCommand{\mafx@econstit}{\mafx@econst@class{\mathnormal{e}}}
\DeclareRobustCommand{\mafx@iunitup}{{\operator@font i}}
\DeclareRobustCommand{\mafx@iunitit}{{\mathnormal{i}}}
\DeclareRobustCommand{\mafx@iunitnb}{{\mathnormal{\mathring\imath}}}
\ifdefined\uppi
\DeclareRobustCommand{\mafx@piconstup}{{\uppi}}
\else\ifdefined\symup
\DeclareRobustCommand{\mafx@piconstup}{{\symup{\pi}}}
\else
\DeclareRobustCommand{\mafx@piconstup}{{\operator@font\pi}}
\fi\fi
\ifdefined\symit
\DeclareRobustCommand{\mafx@piconstit}{{\symit{\pi}}}
\else
\DeclareRobustCommand{\mafx@piconstit}{{\mathnormal{\pi}}}
\fi
```

econst numsetclass iunit piconst

Implement mathematical constants individually with the option to adjust their macro names:

```
\define@key{mafx@}{econst}[\econst]{\let#1=\mafx@econstup}
\define@key{mafx@}{econst*}[\econst]{\let#1=\mafx@econstit}
\define@key{mafx@}{econstclass}{\let\mafx@econst@class=#1}
\define@key{mafx@}{iunit}[\iunit]{\let#1=\mafx@iunitup}
\define@key{mafx@}{iunit*}[\iunit]{\let#1=\mafx@iunitit}
\define@key{mafx@}{iunit*nb}[\iunit]{\let#1=\mafx@iunitnb}
\define@key{mafx@}{piconst}[\piconst]{\let#1=\mafx@piconstup}
\define@key{mafx@}{piconst*}[\piconst]{\let#1=\mafx@piconstit}
```

Derivative Symbols. Define macros for derivative symbols:

```
\DeclareRobustCommand{\mafx@derup}{{\operator@font d}}
\DeclareRobustCommand{\mafx@derit}{{\mathnormal{d}}}
\DeclareRobustCommand{\mafx@diffup}[1]{\mathinner{\mafx@derup#1}}
\DeclareRobustCommand{\mafx@diffit}[1]{\mathinner{\mafx@derit#1}}
```

Implement derivative symbols individually with the option to adjust their macro names:

```
\define@key{mafx@}{der}[\der]{\let#1=\mafx@derup}
\define@key{mafx@}{der*}[\der]{\let#1=\mafx@derit}
\define@key{mafx@}{diff}[\diff]{\let#1=\mafx@diffup}
\define@key{mafx@}{diff*}[\diff]{\let#1=\mafx@diffit}
```

Assorted Symbols. Define macros for assorted symbols:

```
\DeclareRobustCommand{\mafx@order}{\mathnormal{o}}
\DeclareRobustCommand{\mafx@Order}{\mathnormal{O}}
\DeclareRobustCommand{\mafx@OrderCal}{\mathcal{O}}
\DeclareRobustCommand{\mafx@defeq}{\mathrel{\mathop:}=}
\DeclareRobustCommand{\mafx@eqdef}{=\mathrel{\mathop:}}
\DeclareRobustCommand{\mafx@lapl}{\mathnormal{\Delta}}
```

Implement assorted symbols individually with the option to adjust their macro names:

```
\define@key{mafx@}{order}[\order]{\let#1=\mafx@order}
\define@key{mafx@}{Order}[\Order]{\let#1=\mafx@Order}
\define@key{mafx@}{Order*}[\Order]{\let#1=\mafx@OrderCal}
\define@key{mafx@}{defeq}[\defeq]{\let#1=\mafx@defeq}
\define@key{mafx@}{eqdef}[\eqdef]{\let#1=\mafx@eqdef}
\define@key{mafx@}{lapl}[\lapl]{\let#1=\mafx@lapl}
```

Number Sets.

\numset Define macros for number sets. Use default font `\mathbb` if available, otherwise `\mathbf`:

```
\ifdefined\mathbb
\let\mafx@numset@font=\mathbb
\else
\let\mafx@numset@font=\mathbf
\fi
\DeclareRobustCommand{\mafx@numset}{\mafx@numset@font}
\DeclareRobustCommand{\mafx@numsetZ}{\mafx@numset{Z}}
\DeclareRobustCommand{\mafx@numsetQ}{\mafx@numset{Q}}
\DeclareRobustCommand{\mafx@numsetR}{\mafx@numset{R}}
\DeclareRobustCommand{\mafx@numsetC}{\mafx@numset{C}}
\DeclareRobustCommand{\mafx@numsetH}{\mafx@numset{H}}
\DeclareRobustCommand{\mafx@numsetN}{\mafx@numset{N}}
\DeclareRobustCommand{\mafx@numsetO}{\mafx@numset{O}}
```

numset numsetfont numsets Implement number set font and a collection of standard number sets by name:

```
\define@key{mafx@}{numset}[\numset]{\let#1=\mafx@numset}
\define@key{mafx@}{numsetfont}{\let\mafx@numset@font=#1}
\define@key{mafx@}{numsets}[]{%
  \let\Integer=\mafx@numsetZ
  \let\Rational=\mafx@numsetQ
  \let\Real=\mafx@numsetR
  \let\Complex=\mafx@numsetC
  \let\Quaternion=\mafx@numsetH
  \let\Natural=\mafx@numsetN
  \let\Octonion=\mafx@numsetO
}
```

Package Options.

\ProvideMathFix Implement a particular fix or option:

```
\newcommand{\ProvideMathFix}[1]{\setkeys{mafx@}{#1}}
```

Pass undeclared options on to keyval processing:

```
\DeclareOption*{\expandafter\ProvideMathFix\expandafter{\CurrentOption}}
```

Process package options:

```
\ProcessOptions
```