

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`

v1.2: 2026-03-23

<https://ctan.org/pkg/mathfixs>

<https://github.com/nbeisert/latex-pkg-nb>

Abstract

`mathfixs` is a $\text{\LaTeX} 2_{\epsilon}$ 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	2
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	10
3	Information	13
3.1	Copyright	13
3.2	Files and Installation	13
3.3	Related CTAN Packages	14
3.4	Revision History	14
A	Sample	15
B	Implementation	19

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, L^AT_EX 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:

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

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.

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

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 T_EXbook, 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.:

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

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 (`\scriptscriptstyle`).

\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.:

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

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

\genfrac The macro `\genfrac` of the package **amsmath** is modified suitably when the latter is loaded.

\dfrac This also affects the macros `\dfrac` and `\tfrac` for fractions in styles `\displaystyle` and

\tfrac `\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 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. **fracdelimclass** 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 **Feature vfrac.** Common or vulgar fractions such as $\frac{1}{2}$ 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 The option `vfracclass=class` can be used to customise the math class of the vulgar fraction. **vfracskippre** The options `vfracskippre=muskip` and `vfracskippost=muskip` defines the (negative) skip around the slash (must be given in math units μ). **vfracskippost**

2.3 Roots

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

$$y\sqrt{x}z, \quad y\sqrt{x}z, \quad y\sqrt[4]{x}z, \quad y\sqrt[5]{x}z, \quad y\sqrt[3]{x}z, \quad y\sqrt[123]{x}z$$

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 **Feature root.** The package provides an alternative mechanism for the placement of the `\sqrt` 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

$$y\sqrt{x}z, \quad y\sqrt{x}z, \quad y\sqrt[4]{x}z, \quad y\sqrt[5]{x}z, \quad y\sqrt[3]{x}z, \quad y\sqrt[123]{x}z$$

rootclass The option `rootclass=class` can be used to customise the math class of the root. The op-
rootskipend
rootskippre
rootskippost

tion `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 dx) 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.

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

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

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.:

$$12c\.\Delta x\.\Delta y\cdot z \quad \text{or} \quad \mathinner{\int x\.\,dx}$$

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 `\preGamma`, 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 `\mathbf{bold}` 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 `\prealpha`, 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 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 Feature mathbold. The feature **mathbold** defines a bold italic math font `\mathbf{bold}`. To typeset bold italic letters simply use `\mathbf{bold}{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 `\mathbf{bold}`, 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
<code>\exp</code>	exp	exponential function
<code>\log</code>	log	logarithm function

<code>\lg</code>	lg	common logarithm function
<code>\ln</code>	ln	natural logarithm function

Trigonometric functions:

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

Hyperbolic functions:

macro	output	description
<code>\sinh</code>	sinh	hyperbolic sine function
<code>\cosh</code>	cosh	hyperbolic cosine function
<code>\tanh</code>	tanh	hyperbolic tangent function
<code>\coth</code>	coth	hyperbolic cotangent function

Extremal values:

macro	output	description
<code>\max</code>	max	maximum of a set
<code>\min</code>	min	minimum of a set
<code>\sup</code>	sup	supremum of a set
<code>\inf</code>	inf	infimum of a set

Limits:

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

Assorted functions and operators:

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

Modulo statements:

macro	output	description
<code>\bmod</code>	$X \bmod Y$	modulo statement
<code>\pmod</code>	$X \pmod Y$	modulo statement in parentheses
<code>\mod</code>	$X \mod Y$	modulo statement with space (<code>amsmath</code> 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	<code>\sgn</code>	sgn	signum function
res	<code>\res</code>	res	residue operator
lcm	<code>\lcm</code>	lcm	least common multiple
span	<code>\Span</code>	span	span
diag	<code>\diag</code>	diag	diagonal matrix
spec	<code>\spec</code>	spec	spectrum
const	<code>\const</code>	const	anything constant
genop			all of the above

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

feature	macro	output	description
id	<code>\id</code>	id	identity map
tr	<code>\tr</code>	tr	trace
ad	<code>\ad</code>	ad	adjoint (action)
Vol	<code>\Vol</code>	Vol	Volume

reim 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 `\Re` and `\Im` given by ‘ \Re ’ and ‘ \Im ’. The feature **reim** overwrites the predefined macros `\Re` and `\Im` with a textual representation:

macro	output	description
<code>\Re</code>	Re	real part of a complex number
<code>\Im</code>	Im	imaginary part of a complex number

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

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

macro	output	description
<code>\arccot</code>	<code>arccot</code>	inverse cotangent function
<code>\arcsec</code>	<code>arcsec</code>	inverse secant function
<code>\arccsc</code>	<code>arccsc</code>	inverse cosecant function

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

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

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

macro	output	description
<code>\dom</code>	<code>dom</code>	domain
<code>\codom</code>	<code>codom</code>	codomain
<code>\supp</code>	<code>supp</code>	support
<code>\im</code>	<code>im</code>	image
<code>\coker</code>	<code>coker</code>	cokernel
<code>\rank</code>	<code>rank</code>	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 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
<code>\Hom</code>	<code>Hom</code>	homomorphisms
<code>\End</code>	<code>End</code>	endomorphisms
<code>\Isom</code>	<code>Isom</code>	isomorphisms
<code>\Aut</code>	<code>Aut</code>	automorphisms

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

macro	output	description
-------	--------	-------------

<code>\grad</code>	grad	gradient
<code>\div</code>	div	divergence
<code>\curl</code>	curl	curl
<code>\curl</code>	rot	curl (alternative form using <code>vecdiff*</code> or <code>vecrot</code>)

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 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 $e^{i\pi} = -1$).
iunit The unit element of some algebraic structure similarly deserves mention as it could be type-
piconst set in various forms. The package provides macros for these with diverse representations:
unitel

feature	macro	output	description
<code>econst</code>	<code>\econst</code>	e	Euler’s number (upright)
<code>econst*</code>		e	(math italic)
<code>iunit</code>	<code>\iunit</code>	i	imaginary unit (upright)
<code>iunit*</code>		i	(math italic)
<code>iunit*nb</code>		\imath	(NB’s silly circle version)
<code>piconst</code>	<code>\piconst</code>	π	circle constant π (upright, if only ...)
<code>piconst*</code>		π	(math italic)
<code>unitel1</code>	<code>\unitel</code>	$\mathbb{1}$	unit element (doublestroke ‘1’)
<code>unitel1*</code>		1	(plain ‘1’)
<code>unitelvar</code>			(variants, see table below)

Each macro and representation is provided as an individual feature `feature`[`={\cmd}`] which offers the option to customise the macro name to `\cmd` (or use several representations side-by-side). The following table lists variants for the unit element representation.

	output/ <i>var</i>	output/ <i>var</i>	output/ <i>var</i>
dsfont ‘1’	$\mathbb{1}$ 1ds (default)	$\mathbb{1}$ 1dsr (serif)	$\mathbb{1}$ 1dss (sans)
bbm ‘1’	$\mathbb{1}$ 1bbm (serif)	$\mathbb{1}$ 1bbmss (sans)	$\mathbb{1}$ 1bbmtt (typewriter)
bbold ‘1’	$\mathbb{1}$ 1bbold (sans)		
bold ‘1’	$\mathbf{1}$ 1bf (extended)	$\mathbf{1}$ 1bfb (narrower)	$\mathbf{1}$ 1bfbx (extended)
letter ‘e’	e e	e e*	
letter ‘E’	E E	E E*	
‘I’ / ‘id’	I I	I I*	id id

The doublestroke/blackboard bold representations require installation of the further packages `dsfont`/`doublestroke`, `bbm`/`bbm-macros` and/or `bbold`. Inclusion of the packages in your document is not required, as `mathfixs` will try to include the fonts anyway. The default form `unitel1` will pick the first available form from the above packages and resort to the bold form otherwise.

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. $2e^{\sin x} F$). 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 Derivative Symbols. The package provides some assorted elementary symbols and compounds:

feature	macro	output	description
der	<code>\der</code>	d	derivative symbol (upright)
der*	<code>\der</code>	d	derivative symbol (math italic)
diff	<code>\diff{x}</code>	dx	coordinate differential (upright)
diff*	<code>\diff{x}</code>	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.

trans Conjugation Operations. The package provides macros for the matrix and vector operations transposition, hermitian conjugation, complex conjugation and dualisation in various representations:

feature	macro	output	description
trans	<code>\trans</code>	\top	transposition (<code>\top</code>)
trans*		\top	alternate symbol (<code>\intercal</code> , requires <code>amssymb</code>)
transT		T	letter ‘T’
transT*		T	letter ‘T’ (<code>\scriptscriptsize</code>)
transt		t	letter ‘t’
hconj	<code>\hconj</code>	\dagger	hermitian conjugation (<code>\dagger</code>)
hconjH		H	letter ‘H’
hconjH*		H	letter ‘H’ (<code>\scriptscriptsize</code>)
hconjh		h	letter ‘h’
conj	<code>\conj</code>	$*$	complex conjugation (<code>\ast</code>)
dual	<code>\dual</code>	$*$	dualisation (<code>\ast</code>)

The macros are intended to be used within superscripts. Each macro and representation is provided as an individual feature `feature[={\cmd}]` which offers the option to customise the macro name to `\cmd`.

transfont The default font for letter representations of transposition and hermitian conjugation is `\mathsf`; it can be adjusted by `transfont={mathfont}`.

Assorted Symbols. The package provides some assorted elementary symbols and compounds:

feature	macro	output	description
<code>order</code>	<code>\order</code>	o	anything of order less than
<code>Order</code>	<code>\Order</code>	O	anything of order at most
<code>Order*</code>	<code>\Order</code>	\mathcal{O}	anything of order at most (calligraphic)
<code>lapl</code>	<code>\lapl</code>	Δ	Laplace operator
<code>defeq</code>	<code>\defeq</code>	$:=$	is defined as
<code>eqdef</code>	<code>\eqdef</code>	$=:$	defines
<code>dualop</code>	<code>\dualop</code>	\star	dual operator (e.g. Hodge)
<code>dualop*</code>	<code>\dualop</code>	$*$	dual operator (alternative form)

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 Number Sets. 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{b}` of `amssymb` if available at the time of loading, otherwise `\mathbf{b}`; it can be adjusted by `numsetfont={mathfont}`.

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

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

grp Standard Groups. The feature `grp` provides the macro `\grp` to typeset a group name which is commonly denoted by upright (mostly capital) letters such as GL or SO . The feature `grp={\cmd}` offers the option to customise the macro name to `\cmd`. The feature `groups` defines a couple of common group names.

grpfin The basic finite groups are provided by the feature `grpfin` as follows:

macro	output	description
<code>\grpcyc</code>	\mathbb{Z}	cyclic group
<code>\grpcyc</code>	\mathbb{C}	cyclic group (alternative form using <code>grpcycC</code> or <code>grpfin*</code>)
<code>\grp dih</code>	\mathbb{D}	dihedral group
<code>\grp perm</code>	\mathbb{S}	permutation group
<code>\grp alt</code>	\mathbb{A}	alternating group

grp Lie The basic groups of Lie type are provided by the feature `grp Lie` as follows:

macro	output	description
<code>\grp GL</code>	GL	general linear group
<code>\grp SL</code>	SL	special linear group
<code>\grp PGL</code>	PGL	projective linear group

<code>\grpPSL</code>	PSL	projective special linear group
<code>\grpU</code>	U	unitary group
<code>\grpSU</code>	SU	special unitary group
<code>\grpPU</code>	PU	projective unitary group
<code>\grpPSU</code>	PSU	projective special unitary group
<code>\grpO</code>	O	orthogonal group
<code>\grpSO</code>	SO	special orthogonal group
<code>\grpPSO</code>	PSO	projective special orthogonal group
<code>\grpSpin</code>	Spin	spin group
<code>\grpPin</code>	Pin	pin group
<code>\grpSp</code>	Sp	symplectic group
<code>\grpMp</code>	Mp	metaplectic group

`grpdynkin` The seven classes of groups associated to Dynkin diagrams are provided by the feature `grpdynkin` as follows:

macro	output	description
<code>\grpA</code>	A	group of type A
<code>\grpB</code>	B	group of type B
<code>\grpC</code>	C	group of type C
<code>\grpD</code>	D	group of type D
<code>\grpE</code>	E	group of (exceptional) type E
<code>\grpF</code>	F	group of (exceptional) type F
<code>\grpG</code>	G	group of (exceptional) type G

3 Information

3.1 Copyright

Copyright © 2018–2026 Niklas Beisert

This work may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in <https://www.latex-project.org/lppl.txt> and version 1.3c or later is part of all distributions of L^AT_EX version 2008 or later.

This work has the LPPL maintenance status ‘maintained’.

The Current Maintainer of this work is Niklas Beisert.

This work consists of the files `README.txt`, `mathfixs.ins` and `mathfixs.dtx` as well as the derived files listed below.

3.2 Files and Installation

The package consists of the files:

<code>README.txt</code>	readme file
<code>mathfixs.ins</code>	installation file
<code>mathfixs.dtx</code>	source file
<code>mathfixs.sty</code>	package file
<code>mathfixs-samp.tex</code>	sample file
<code>mathfixs.pdf</code>	documentation

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

- Run \LaTeX on `mathfixs.ins` to create the package `mathfixs.sty` and further documentation files.
- Copy the file `mathfixs.sty` to an appropriate directory of your \LaTeX distribution, e.g. `texmf-root/tex/latex/mathfixs`.
- Alternatively, you may copy `mathfixs.sty` to your project directory.
- Run (pdf) \LaTeX on `mathfixs.dtx` to compile the documentation `mathfixs.pdf` (this file).

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.
- This package cooperates with the package `beamer` in not changing colours for the root exponent. Compatibility with the `beamer` package has been tested with v3.77 (2026-01-22).

3.4 Revision History

v1.2: 2026-03-23

- added unit element `\unitel` in various representations
- adjoint action `\ad`, volume `\Vol` and dual operator `\dualop`
- added transposition `\trans`, hermitian conjugation `\hconj`, complex conjugation `\conj` and dualisation `\dual`
- added standard finite and Lie groups `\grp...`
- fix for not applying `math text` colour for root exponent in `beamer`
- use `\muexpr` to protect muskip expressions
- use elementary register manipulations
- package maintenance

v1.1.3: 2025-03-25

- declared more features as robust commands (for use in titles, captions, etc.; thanks to Jonáš Dujava for bug report)
- maintenance and manual update

v1.12: 2024-11-18

- fixed loading without `amsmath`

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 hyperbolic 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

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:

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

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

```
2 \usepackage{geometry}
3 \geometry{layout=a4paper}
4 \geometry{paper=a4paper}
5 \geometry{margin=3cm}
6 \parindent0pt
```

Include amsmath, amssymb and the mathfixs package:

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

Declare features to be implemented globally:

```
9 \ProvideMathFix{greekcaps,mathbold}
10 \ProvideMathFix{frac,rfrac,vfrac}
11 \ProvideMathFix{multskip}
12 \ProvideMathFix{der,diff}
```

Start document body:

```
13 \begin{document}
```

Fractions.

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

Fraction spacing:

```
15 \[
16 x\frac{a+b}{c+d}\frac{e}{f}.
17 \]
```

Vulgar fractions:

```
18 Recipe for $\approx 3$ pancakes (scale accordingly):
19 Whisk together one large egg, $\frac{1}{8}$ \ell$ milk,
20 $50\mathinner{\mathrm{g}}$ flour
21 (mix $\frac{2}{3}$ wholemeal flour and $\frac{1}{3}$ white flour); fry in pan.
22 Feel free to use more flour than indicated.
```

Rational numbers:

```
23 \[
24 \frac{1}{2}x^2 + \frac{1}{6} y^3 + \frac{1}{4} x^2 y^2
25 \]
```

Radicals.

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

Original radicals spacing:

```
27 \[
28 y\sqrt{x}z,\quad
29 y\sqrt[]{x}z,\quad
30 y\sqrt[i]{x}z,\quad
31 y\sqrt[n]{x}z,\quad
32 y\sqrt[3]{x}z,\quad
33 y\sqrt[123]{x}z
34 \]
```

Radicals spacing (feature introduced in document body):

```
35 \ProvideMathFix{root}
36 \[
37 y\sqrt{x}z,\quad
38 y\sqrt[]{x}z,\quad
39 y\sqrt[i]{x}z,\quad
```



```

40 y\sqrt[n]{x}z,\qquad
41 y\sqrt[3]{x}z,\qquad
42 y\sqrt[123]{x}z
43 \]

```

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

```

44 \[
45 \ProvideMathFix{rootclose}
46 y\sqrt{x}z,\qquad
47 y\sqrt[] {x}z,\qquad
48 y\sqrt[i]{x}z,\qquad
49 y\sqrt[n]{x}z,\qquad
50 y\sqrt[3]{x}z,\qquad
51 y\sqrt[123]{x}z
52 \]

```

Radicals with class `\mathinner`:

```

53 \[
54 \ProvideMathFix{rootclass={\mathinner}}
55 y\sqrt{x}z,\qquad
56 y\sqrt[] {x}z,\qquad
57 y\sqrt[i]{x}z,\qquad
58 y\sqrt[n]{x}z,\qquad
59 y\sqrt[3]{x}z,\qquad
60 y\sqrt[123]{x}z
61 \]

```

Spaces Representing Multiplication and Derivative Symbols.

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

Explicit spacing:

```

63 \[
64 \int x\.\der x
65 \]

```

Implicit spacing:

```

66 \[
67 \int x\diff{x}
68 \]

```

Greek Letters.

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

Capital letters:

```

70 \[
71 \Gamma \qquad
72 \mathrm{\Gamma} \qquad
73 c\operatorname{\Gamma}(x)
74 \]

```

Bold Text with Math Symbols.

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

Bold text:

```
76 \[
77 aG\alpha\Gamma
78 \qquad\mathbf{aG\alpha\Gamma}
79 \qquad\ProvideMathFix{greeklower}\mathbf{aG\alpha\Gamma}
80 \]
```

Real and Imaginary Parts.

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

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

```
82 \[
83 \Re(z),\Im(z)
84 \qquad\ProvideMathFix{reim}
85 \Re(z),\Im(z)
86 \qquad\ProvideMathFix{reim={\nRe\nIm}}
87 \nRe(z),\nIm(z)
88 \]
```

Functions, Operators, Symbols.

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

Some examples of vector differential operators:

```
90 \[
91 \ProvideMathFix{vecdiff,lapl,defeq}
92 \lapl f=\div\grad f,
93 \qquad
94 \lapl v=\grad\div v-\curl\curl v,
95 \]
```

Some examples of mathematical constants:

```
96 \[
97 \ProvideMathFix{econst,iunit,piconst}
98 \econst^{\iunit\piconst}=-1,
99 \qquad
100 \ProvideMathFix{econst*,iunit*,piconst*}
101 \econst^{\iunit\piconst}=-1,
102 \qquad
103 \ProvideMathFix{econst,defeq,iunit*nb,numsets}
104 z\defeq x+\iunit y=r\econst^{\iunit\theta}\in\Complex,
105 \quad x,y,r,\theta\in\Real
106 \]
```

Some examples of operators and symbols:

```
107 \[
108 \ProvideMathFix{res,hyp,iunit*,piconst*}
109 \res_{z=\iunit \piconst n} \csch(z)=(-1)^n
110 \]
```

End of document body:

```
111 \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:

```
112 \RequirePackage{keyval}
```

General Definitions.

`\mafx@letcs` Define some auxiliary macros for the package. `\mafx@letcs` makes a `\let` assignment on the csname `#1`. `\mafx@mathopfont` encapsules `\operator@font` within a block.

```
113 \def\mafx@letcs#1{\expandafter\let\csname#1\endcsname}
114 \DeclareRobustCommand{\mafx@mathopfont}[1]{\operator@font#1}
```

Automatically Bold Maths.

`\bfseries` 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:

```
115 \expandafter\DeclareRobustCommand\expandafter\mafx@bfseries\expandafter
116   {\bfseries\ifmmode\else\boldmath\fi}
117 \expandafter\DeclareRobustCommand\expandafter\mafx@mdseries\expandafter
118   {\mdseries\ifmmode\else\unboldmath\fi}
119 \expandafter\DeclareRobustCommand\expandafter\mafx@normalfont\expandafter
120   {\normalfont\ifmmode\else\unboldmath\fi}
```

`autobold` Implement the autobold fix:

```
121 \define@key{mafx@}{autobold}[]{%
122   \let\bfseries=\mafx@bfseries
123   \let\mdseries=\mafx@mdseries
124   \let\normalfont=\mafx@normalfont
125 }
```

Bold Italic Math.

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

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

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

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

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

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

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

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

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

```
133 \DeclareRobustCommand{\mafx@frac}[2]{%
132 \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:

```
133 \DeclareRobustCommand{\mafx@genfrac}[5]{\begingroup
134 \ifx#2\@@overwithdelims\let\mafx@frac@class\mafx@frac@delimclass\fi
135 \ifx#2\@@abovewithdelims\let\mafx@frac@class\mafx@frac@delimclass\fi
136 \ifx#2\@@atopwithdelims\let\mafx@frac@class\mafx@frac@delimclass\fi
137 \mafx@frac@class{#1{\begingroup#4\endgroup#2#3\relax#5}}%
138 \endgroup}
```

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

```
139 \define@key{mafx@}{frac}[]{}%
140 \let\frac=\mafx@frac
141 \let\genfrac=\mafx@genfrac
142 }
```

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

```
fracdelimclass 143 \define@key{mafx@}{fracclass}{\def\mafx@frac@class{#1}}
144 \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.: $\frac{1}{2}$:

```
145 \DeclareRobustCommand{\mafx@rfrac}[2]{\mathchoice
146 {\textstyle{\begingroup#1\endgroup\@@over#2}}%
147 {\begingroup#1\endgroup\@@over#2}%
148 {\begingroup#1\endgroup\@@over#2}%
149 {\begingroup#1\endgroup\@@over#2}}
```

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

```
150 \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:

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

`\vfrac` Define a vulgar representation of a rational number, e.g.: $\frac{1}{2}$. Automatically switch to math mode if in text mode:

```
154 \DeclareRobustCommand{\mafx@vfrac}[2]{\ifmmode
```

```

155 \mafx@vfrac@class{\textstyle
156 ~{#1}\mkern-\muexpr\mafx@vfrac@preskip\relax
157 /\mkern-\muexpr\mafx@vfrac@postskip\relax_{#2}}%
158 \else$\mafx@vfrac{#1}{#2}$\fi}

```

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

```

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

```

vfracclass Define configurable skip parameters for `\vfrac`:

```

vfracskippre 160 \define@key{mafx@}{vfracclass}{\def\mafx@vfrac@class{#1}}
vfracskippost 161 \define@key{mafx@}{vfracskippre}{\def\mafx@vfrac@preskip{#1}}
162 \define@key{mafx@}{vfracskippost}{\def\mafx@vfrac@postskip{#1}}

```

Interaction with beamerbasecolor. Suppress changing colors in inner math structures when beamer class is loaded:

```

163 \let\mafx@beamerbasecolor@fix\@empty
164 \AddToHook{package/beamerbasecolor/after}{%
165 \def\mafx@beamerbasecolor@fix{\donotcoloroutermaths}}

```

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:

```

166 \let\mafx@root@close=\@empty
167 \def\mafx@root@class{}
168 \def\mafx@root@endskip{0.6667\thinmuskip}
169 \def\mafx@root@preskip{0mu}
170 \def\mafx@root@postskip{0.3333\thinmuskip}

```

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

```

171 \DeclareRobustCommand\mafx@sqrt{\ifnextchar[\@sqrt{\@sqrt{}}}%
172 \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`:

```

173 \def\mafx@root#1\of{%
174 \setbox\rootbox\hbox{\mafx@beamerbasecolor@fix$\math@scriptscriptstyle{%
175 \gdef\mafx@gleftroot{0}\gdef\mafx@guproot{0}}%
176 \def\leftroot##1{\gdef\mafx@gleftroot{##1}}%
177 \def\uproot##1{\gdef\mafx@guproot{##1}}%
178 #1}$}%
179 \let\mafx@leftroot=\mafx@gleftroot
180 \let\mafx@uproot=\mafx@guproot
181 \mathpalette\r@@t}

```

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

```

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

```

Determine the font selector for the current style:

```

183 \ifx#1\scriptstyle\let\mafx@tmp@fontsel=\scriptfont\else
184 \ifx#1\scriptscriptstyle\let\mafx@tmp@fontsel=\scriptscriptfont\else
185 \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:

```

186 \sbox\z@{\$ \m@th#1\null\delimiterspace=\z@\radical\z@{#2}$}%
187 \dimen@ \ht\z@
188 \ifx#1\displaystyle
189 \advance\dimen@-\fontdimen8\textfont3\relax
190 \advance\dimen@-0.25\fontdimen5\textfont2\relax
191 \else
192 \advance\dimen@-1.25\fontdimen8\mafx@tmp@fontsel3\relax
193 \fi
194 \setbox\z@=\hbox{\$ \m@th#1\sqrtsign{%
195 \vrule width\z@ height\dimen@ depth\dp\z@}$}%

```

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:

```

196 \dimen@ \dimexpr0.6\wd\z@-\wd\rootbox\relax
197 \ifdim\dimen@>\z@\else\dimen@ \z@\fi
198 \kern\dimen@

```

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

```

199 \setbox\@ne=\hbox{\$ \m@th#1\mskip\muexpr\mafx@uproot mu\relax$}%
200 \dimen@ \dimexpr0.6\ht\z@-0.6\dp\z@+\wd\@ne\relax

```

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

```

201 \mkern-\muexpr\mafx@leftroot mu\relax
202 \raise\dimen@\copy\rootbox
203 \mkern\muexpr\mafx@leftroot mu\relax
204 \kern-0.6\wd\z@

```

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

```

205 \mkern\muexpr\mafx@root@preskip\relax
206 \sqrtsign{#2\mskip\muexpr\mafx@root@endskip\relax}%

```

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:

```

207 \ifx\mafx@root@close\@empty\else
208 \dimen@ \fontdimen8\mafx@tmp@fontsel3\relax
209 \lower\dimen@\hbox{%
210 \vrule width\dimen@ height\ht\z@ depth-\mafx@root@close\ht\z@}%
211 \fi

```

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

```

212 \mkern\mafx@root@postskip}}

```

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

```

213 \define@key{mafx@}{root}[]{}%

```

```

214 \let\sqrt=\mafx@sqrt
215 \let@sqrt=\mafx@@sqrt
216 \let\root=\mafx@root
217 \let\r@@t=\mafx@r@@t
218 }

```

```

rootclose Define configurable parameters for alternative root:
rootclass
rootskipend 219 \define@key{mafx@}{rootclose}[0.8]{\def\mafx@root@close{#1}}
rootskippre 220 \define@key{mafx@}{rootclass}{\let\mafx@root@class=#1}
221 \define@key{mafx@}{rootskipend}{\def\mafx@root@endskip{#1}}
rootskipbefore 222 \define@key{mafx@}{rootskippre}{\def\mafx@root@preskip{#1}}
223 \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:

```

224 \let\mafx@old@dot=\.
225 \def\mafx@dot@skip{\thinmuskip}
226 \def\mafx@dot{\mskip\muexpr\mafx@dot@skip\relax}
227 \DeclareRobustCommand{\mafx@per@dot}{%
228 \ifmmode\expandafter\mafx@dot\else\expandafter\mafx@old@dot\fi}

```

multskip Implement definition of \.:

```

229 \define@key{mafx@}{multskip}[\thinmuskip]{%
230 \let\.=\mafx@per@dot
231 \def\mafx@dot@skip{#1}%
232 }

```

Italic Capital Greek Letters.

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

```

233 \DeclareMathSymbol{\mafx@Gamma}{\mathalpha}{letters}{0}
234 \DeclareMathSymbol{\mafx@Delta}{\mathalpha}{letters}{1}
235 \DeclareMathSymbol{\mafx@Theta}{\mathalpha}{letters}{2}
236 \DeclareMathSymbol{\mafx@Lambda}{\mathalpha}{letters}{3}
237 \DeclareMathSymbol{\mafx@Xi}{\mathalpha}{letters}{4}
238 \DeclareMathSymbol{\mafx@Pi}{\mathalpha}{letters}{5}
239 \DeclareMathSymbol{\mafx@Sigma}{\mathalpha}{letters}{6}
240 \DeclareMathSymbol{\mafx@Upsilon}{\mathalpha}{letters}{7}
241 \DeclareMathSymbol{\mafx@Phi}{\mathalpha}{letters}{8}
242 \DeclareMathSymbol{\mafx@Psi}{\mathalpha}{letters}{9}
243 \DeclareMathSymbol{\mafx@Omega}{\mathalpha}{letters}{10}

```

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

```

244 \define@key{mafx@}{greekcaps}[]{%
245 \mafx@letcs{#1Gamma}=\mafx@Gamma
246 \mafx@letcs{#1Delta}=\mafx@Delta
247 \mafx@letcs{#1Theta}=\mafx@Theta

```

```

248 \mafx@letcs{#1Lambda}=\mafx@Lambda
249 \mafx@letcs{#1Xi}=\mafx@Xi
250 \mafx@letcs{#1Pi}=\mafx@Pi
251 \mafx@letcs{#1Sigma}=\mafx@Sigma
252 \mafx@letcs{#1Upsilon}=\mafx@Upsilon
253 \mafx@letcs{#1Phi}=\mafx@Phi
254 \mafx@letcs{#1Psi}=\mafx@Psi
255 \mafx@letcs{#1Omega}=\mafx@Omega
256 }

```

Lowercase Greek Letters in Standard Math Type.

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

```

257 \DeclareMathSymbol{\mafx@alpha}{\mathalpha}{letters}{11}
258 \DeclareMathSymbol{\mafx@beta}{\mathalpha}{letters}{12}
259 \DeclareMathSymbol{\mafx@gamma}{\mathalpha}{letters}{13}
260 \DeclareMathSymbol{\mafx@delta}{\mathalpha}{letters}{14}
261 \DeclareMathSymbol{\mafx@epsilon}{\mathalpha}{letters}{15}
262 \DeclareMathSymbol{\mafx@zeta}{\mathalpha}{letters}{16}
263 \DeclareMathSymbol{\mafx@eta}{\mathalpha}{letters}{17}
264 \DeclareMathSymbol{\mafx@theta}{\mathalpha}{letters}{18}
265 \DeclareMathSymbol{\mafx@iota}{\mathalpha}{letters}{19}
266 \DeclareMathSymbol{\mafx@kappa}{\mathalpha}{letters}{20}
267 \DeclareMathSymbol{\mafx@lambda}{\mathalpha}{letters}{21}
268 \DeclareMathSymbol{\mafx@mu}{\mathalpha}{letters}{22}
269 \DeclareMathSymbol{\mafx@nu}{\mathalpha}{letters}{23}
270 \DeclareMathSymbol{\mafx@xi}{\mathalpha}{letters}{24}
271 \DeclareMathSymbol{\mafx@pi}{\mathalpha}{letters}{25}
272 \DeclareMathSymbol{\mafx@rho}{\mathalpha}{letters}{26}
273 \DeclareMathSymbol{\mafx@sigma}{\mathalpha}{letters}{27}
274 \DeclareMathSymbol{\mafx@tau}{\mathalpha}{letters}{28}
275 \DeclareMathSymbol{\mafx@upsilon}{\mathalpha}{letters}{29}
276 \DeclareMathSymbol{\mafx@phi}{\mathalpha}{letters}{30}
277 \DeclareMathSymbol{\mafx@chi}{\mathalpha}{letters}{31}
278 \DeclareMathSymbol{\mafx@psi}{\mathalpha}{letters}{32}
279 \DeclareMathSymbol{\mafx@omega}{\mathalpha}{letters}{33}
280 \DeclareMathSymbol{\mafx@varepsilon}{\mathalpha}{letters}{34}
281 \DeclareMathSymbol{\mafx@vartheta}{\mathalpha}{letters}{35}
282 \DeclareMathSymbol{\mafx@varpi}{\mathalpha}{letters}{36}
283 \DeclareMathSymbol{\mafx@varphi}{\mathalpha}{letters}{39}
284 \DeclareMathSymbol{\mafx@varrho}{\mathalpha}{letters}{37}
285 \DeclareMathSymbol{\mafx@varsigma}{\mathalpha}{letters}{38}

```

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

```

286 \define@key{mafx@}{greeklower}[]{}
287 \mafx@letcs{#1alpha}=\mafx@alpha
288 \mafx@letcs{#1beta}=\mafx@beta
289 \mafx@letcs{#1gamma}=\mafx@gamma
290 \mafx@letcs{#1delta}=\mafx@delta
291 \mafx@letcs{#1epsilon}=\mafx@epsilon
292 \mafx@letcs{#1zeta}=\mafx@zeta
293 \mafx@letcs{#1eta}=\mafx@eta
294 \mafx@letcs{#1theta}=\mafx@theta
295 \mafx@letcs{#1iota}=\mafx@iota

```



```

296 \mafx@letcs{#1kappa}=\mafx@kappa
297 \mafx@letcs{#1lambda}=\mafx@lambda
298 \mafx@letcs{#1mu}=\mafx@mu
299 \mafx@letcs{#1nu}=\mafx@nu
300 \mafx@letcs{#1xi}=\mafx@xi
301 \mafx@letcs{#1pi}=\mafx@pi
302 \mafx@letcs{#1rho}=\mafx@rho
303 \mafx@letcs{#1sigma}=\mafx@sigma
304 \mafx@letcs{#1tau}=\mafx@tau
305 \mafx@letcs{#1upsilon}=\mafx@upsilon
306 \mafx@letcs{#1phi}=\mafx@phi
307 \mafx@letcs{#1chi}=\mafx@chi
308 \mafx@letcs{#1psi}=\mafx@psi
309 \mafx@letcs{#1omega}=\mafx@omega
310 \mafx@letcs{#1varepsilon}=\mafx@varepsilon
311 \mafx@letcs{#1vartheta}=\mafx@vartheta
312 \mafx@letcs{#1varpi}=\mafx@varpi
313 \mafx@letcs{#1varphi}=\mafx@varphi
314 \mafx@letcs{#1varrho}=\mafx@varrho
315 \mafx@letcs{#1varsigma}=\mafx@varsigma
316 }

```

Assorted Functions, Operators, Symbols. The following definitions require amsopn:

```
317 \ifdefined\DeclareMathOperator
```

Define macros for assorted functions, operators, symbols:

```

318 \DeclareMathOperator{\mafx@sgn}{sgn}
319 \DeclareMathOperator*{\mafx@res}{res}
320 \DeclareMathOperator{\mafx@lcm}{lcm}
321 \DeclareMathOperator{\mafx@span}{span}
322 \DeclareMathOperator{\mafx@diag}{diag}
323 \DeclareMathOperator{\mafx@spec}{spec}
324 \DeclareMathOperator{\mafx@const}{const}
325 \DeclareMathOperator{\mafx@id}{id}
326 \DeclareMathOperator{\mafx@tr}{tr}
327 \DeclareMathOperator{\mafx@ad}{ad}
328 \DeclareMathOperator{\mafx@Vol}{Vol}

```

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

```

329 \define@key{mafx@}{sgn}[\sgn]{\let#1=\mafx@sgn}
330 \define@key{mafx@}{res}[\res]{\let#1=\mafx@res}
331 \define@key{mafx@}{lcm}[\lcm]{\let#1=\mafx@lcm}
332 \define@key{mafx@}{diag}[\diag]{\let#1=\mafx@diag}
333 \define@key{mafx@}{span}[\Span]{\let#1=\mafx@span}
334 \define@key{mafx@}{spec}[\spec]{\let#1=\mafx@spec}
335 \define@key{mafx@}{const}[\const]{\let#1=\mafx@const}
336 \define@key{mafx@}{id}[\id]{\let#1=\mafx@id}
337 \define@key{mafx@}{tr}[\tr]{\let#1=\mafx@tr}
338 \define@key{mafx@}{ad}[\ad]{\let#1=\mafx@ad}
339 \define@key{mafx@}{Vol}[\Vol]{\let#1=\mafx@Vol}

```

genop Implement all assorted functions, operators and symbols:

```

340 \define@key{mafx@}{genop}[]{}
341 \ProvideMathFix{sgn,res,lcm,diag,span,spec,const}%
342 }

```

Real and Imaginary Part Projectors in Text Form.

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

```
343 \let\mafx@orgRe=\Re
344 \let\mafx@orgIm=\Im
345 \DeclareMathOperator{\mafx@Re}{Re}
346 \DeclareMathOperator{\mafx@Im}{Im}
```

`reim` Implement textual notation for `\Re` and `\Im`:

```
347 \define@key{mafx@}{reim}[\Re\Im]{%
348   \expandafter\let\@firstoftwo#1=\mafx@Re
349   \expandafter\let\@secondoftwo#1=\mafx@Im
350 }
351 \define@key{mafx@}{reim*}[\Re\Im]{%
352   \expandafter\let\@firstoftwo#1=\mafx@orgRe
353   \expandafter\let\@secondoftwo#1=\mafx@orgIm
354 }
```

Remaining Inverse Trigonometric Functions.

`\arccot` Define remaining inverse trigonometric functions and their inverses:

```
\arcsec
\arccsc 355 \DeclareMathOperator{\mafx@arccot}{arccot}
356 \DeclareMathOperator{\mafx@arcsec}{arcsec}
357 \DeclareMathOperator{\mafx@arccsc}{arccsc}
```

`trig` Implement remaining trigonometric functions:

```
358 \define@key{mafx@}{trig}[]{%
359   \let\arccot=\mafx@arccot
360   \let\arcsec=\mafx@arcsec
361   \let\arccsc=\mafx@arccsc
362 }
```

Remaining Hyperbolic Functions.

`\sech` Define remaining hyperbolic functions and their inverses:

```
\csch
\ar...h 363 \DeclareMathOperator{\mafx@sech}{sech}
364 \DeclareMathOperator{\mafx@csch}{csch}
365 \DeclareMathOperator{\mafx@arsinh}{arsinh}
366 \DeclareMathOperator{\mafx@arcosh}{arcosh}
367 \DeclareMathOperator{\mafx@artanh}{artanh}
368 \DeclareMathOperator{\mafx@arcoth}{arcoth}
369 \DeclareMathOperator{\mafx@arsech}{arsech}
370 \DeclareMathOperator{\mafx@arcsch}{arcsch}
```

`hyp` Implement remaining hyperbolic functions:

```
371 \define@key{mafx@}{hyp}[]{%
372   \let\sech=\mafx@sech
373   \let\csch=\mafx@csch
374   \let\arsinh=\mafx@arsinh
375   \let\arcosh=\mafx@arcosh
376   \let\artanh=\mafx@artanh}
```

```

377 \let\arcosh=\mafx@arcosh
378 \let\arsech=\mafx@arsech
379 \let\arcsch=\mafx@arcsch
380 }

```

Characteristics of Maps. Define macros for characteristics of maps:

```

381 \DeclareMathOperator{\mafx@dom}{dom}
382 \DeclareMathOperator{\mafx@supp}{supp}
383 \DeclareMathOperator{\mafx@codom}{codom}
384 \DeclareMathOperator{\mafx@im}{im}
385 \DeclareMathOperator{\mafx@rank}{rank}
386 \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:

```

387 \define@key{mafx@}{mapchar}[\im]{%
388 \let\dom=\mafx@dom
389 \let\supp=\mafx@supp
390 \let\codom=\mafx@codom
391 \let\#1=\mafx@im
392 \let\rank=\mafx@rank
393 \let\coker=\mafx@coker
394 }

```

Classes of Structure-Preserving Maps.

```

\Hom Define classes of structure-preserving maps:
\End
\Isom 395 \DeclareMathOperator{\mafx@Hom}{Hom}
\Aut 396 \DeclareMathOperator{\mafx@end}{End}
397 \DeclareMathOperator{\mafx@Isom}{Isom}
398 \DeclareMathOperator{\mafx@Aut}{Aut}

```

mapclass Implement classes of structure-preserving maps:

```

399 \define@key{mafx@}{mapclass}[]{%
400 \let\Hom=\mafx@Hom
401 \let\End=\mafx@end
402 \let\Isom=\mafx@Isom
403 \let\Aut=\mafx@Aut
404 }

```

Differential Operators for Vectors.

```

\grad Define differential operators gradient, divergence and curl (alternative form rot):
\div
\curl 405 \DeclareMathOperator{\mafx@grad}{grad}
406 \DeclareMathOperator{\mafx@div}{div}
407 \DeclareMathOperator{\mafx@curl}{curl}
408 \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:

```

409 \define@key{mafx@}{vecdiff}[]{%

```

```

410 \let\grad=\mafx@grad
411 \let\div=\mafx@div
412 \let\curl=\mafx@curl
413 }
414 \define@key{mafx@}{vecdiff*}[]{%
415 \let\grad=\mafx@grad
416 \let\div=\mafx@div
417 \let\curl=\mafx@rot
418 }
419 \define@key{mafx@}{vecrot}{\curl}{\let#1=\mafx@rot}

```

The above definitions required `amsopn`:

```

420 \fi

```

Mathematical Constants. Try to declare upright and italic lowercase pi. Use `\uppi` for upright `\pi` if available at the time of loading, otherwise use plain `\pi` hoping for availability of an upright version:

```

421 \ifdefined\uppi
422 \def\mafx@piup{\uppi}
423 \else\ifdefined\symup
424 \def\mafx@piup{\symup{\pi}}
425 \else
426 \def\mafx@piup{\operatorname{\pi}}
427 \fi
428 \ifdefined\symit
429 \def\mafx@piit{\symit{\pi}}
430 \else
431 \def\mafx@piit{\mathnormal{\pi}}
432 \fi

```

Declare some math alphabets for blackboard bold unit element:

```

433 \DeclareMathAlphabet{\mafx@mathbfb}{OT1}{cmr}{b}{n}
434 \DeclareMathAlphabet{\mafx@mathbfbx}{OT1}{cmr}{bx}{n}
435 \ifdefined\bbfamily\let\mafx@mathbbold=\mathbb\else\IfFileExists{bbold.sty}{%
436 \DeclareMathAlphabet{\mafx@mathbbold}{U}{bbold}{m}{n}}{\fi
437 \ifdefined\mathds\let\mafx@mathds=\mathds\else\IfFileExists{dsfont.sty}{%
438 \DeclareMathAlphabet{\mafx@mathds}{U}{dsrom}{m}{n}}{\fi
439 \IfFileExists{dsfont.sty}{%
440 \DeclareMathAlphabet{\mafx@mathdsr}{U}{dsrom}{m}{n}
441 \DeclareMathAlphabet{\mafx@mathdss}{U}{dsss}{m}{n}
442 }{\fi
443 \ifdefined\mathbbm\let\mafx@mathds=\mathds\else\IfFileExists{bbm.sty}{%
444 \DeclareMathAlphabet{\mafx@mathbbm}{U}{bbm}{m}{n}
445 \SetMathAlphabet{\mafx@mathbbm}{bold}{U}{bbm}{bx}{n}
446 \DeclareMathAlphabet{\mafx@mathbbmss}{U}{bbmss}{m}{n}
447 \SetMathAlphabet{\mafx@mathbbmss}{bold}{U}{bbmss}{bx}{n}
448 \DeclareMathAlphabet{\mafx@mathbbmtt}{U}{bbmtt}{m}{n}
449 }{\fi
450 \ifdefined\mafx@mathds\let\mafx@mathbb=\mafx@mathds
451 \else\ifdefined\mafx@mathbbm\let\mafx@mathbb=\mafx@mathbbm
452 \else\ifdefined\mafx@mathbbold\let\mafx@mathbb=\mafx@mathbold
453 \else\let\mafx@mathbb=\mathbf{\fi\fi\fi

```

```

\econst Define macros for mathematical constants:
\iunit
\piconst 454 \let\mafx@econst@class=\mathinner
\unitel

```

```

455 \DeclareRobustCommand{\mafx@econstup}{\mafx@econst@class{\operator@font e}}
456 \DeclareRobustCommand{\mafx@econstit}{\mafx@econst@class{\mathnormal{e}}}
457 \DeclareRobustCommand{\mafx@iunitup}{\operator@font i}
458 \DeclareRobustCommand{\mafx@iunitit}{\mathnormal{i}}
459 \DeclareRobustCommand{\mafx@iunitnb}{\mathnormal{\mathring{imath}}}
460 \DeclareRobustCommand{\mafx@piconstup}{\mafx@piup}
461 \DeclareRobustCommand{\mafx@piconstit}{\mafx@piit}
462 \DeclareRobustCommand{\mafx@unitelbb}{\mafx@mathbb{1}}
463 \DeclareRobustCommand{\mafx@unitelds}{\mafx@mathds{1}}
464 \DeclareRobustCommand{\mafx@uniteldsr}{\mafx@mathdsr{1}}
465 \DeclareRobustCommand{\mafx@uniteldss}{\mafx@mathdss{1}}
466 \DeclareRobustCommand{\mafx@unitelbbold}{\mafx@mathbbold{1}}
467 \DeclareRobustCommand{\mafx@unitelbbm}{\mafx@mathbbm{1}}
468 \DeclareRobustCommand{\mafx@unitelbbmss}{\mafx@mathbbmss{1}}
469 \DeclareRobustCommand{\mafx@unitelbbmtt}{\mafx@mathbbmtt{1}}
470 \DeclareRobustCommand{\mafx@unitelbf}{\mafx@mathbf{1}}
471 \DeclareRobustCommand{\mafx@unitelbfb}{\mafx@mathbfb{1}}
472 \DeclareRobustCommand{\mafx@unitelbfbx}{\mafx@mathbfbx{1}}
473 \DeclareRobustCommand{\mafx@uniteleup}{\operator@font e}
474 \DeclareRobustCommand{\mafx@uniteleit}{\mathnormal{e}}
475 \DeclareRobustCommand{\mafx@unitelEup}{\operator@font E}
476 \DeclareRobustCommand{\mafx@unitelEit}{\mathnormal{E}}
477 \DeclareRobustCommand{\mafx@unitelIup}{\operator@font I}
478 \DeclareRobustCommand{\mafx@unitelIit}{\mathnormal{I}}
479 \DeclareRobustCommand{\mafx@unitelid}{\operator@font id}
480 \DeclareRobustCommand{\mafx@unitels}{\operator@font 1}

```

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

```

iunit
piconst
unitel
481 \define@key{mafx@}{econst}{\econst}{\let#1=\mafx@econstup}
482 \define@key{mafx@}{econst*}{\econst}{\let#1=\mafx@econstit}
483 \define@key{mafx@}{econstclass}{\let\mafx@econst@class=#1}
484 \define@key{mafx@}{iunit}{\iunit}{\let#1=\mafx@iunitup}
485 \define@key{mafx@}{iunit*}{\iunit}{\let#1=\mafx@iunitit}
486 \define@key{mafx@}{iunit*nb}{\iunit}{\let#1=\mafx@iunitnb}
487 \define@key{mafx@}{piconst}{\piconst}{\let#1=\mafx@piconstup}
488 \define@key{mafx@}{piconst*}{\piconst}{\let#1=\mafx@piconstit}
489 \define@key{mafx@}{unitel1}{\unitel}{\let#1=\mafx@unitelbb}
490 \define@key{mafx@}{unitel1ds}{\unitel}{\let#1=\mafx@unitelds}
491 \define@key{mafx@}{unitel1dsr}{\unitel}{\let#1=\mafx@uniteldsr}
492 \define@key{mafx@}{unitel1dss}{\unitel}{\let#1=\mafx@uniteldss}
493 \define@key{mafx@}{unitel1bbm}{\unitel}{\let#1=\mafx@unitelbbm}
494 \define@key{mafx@}{unitel1bbmss}{\unitel}{\let#1=\mafx@unitelbbmss}
495 \define@key{mafx@}{unitel1bbmtt}{\unitel}{\let#1=\mafx@unitelbbmtt}
496 \define@key{mafx@}{unitel1bbold}{\unitel}{\let#1=\mafx@unitelbbold}
497 \define@key{mafx@}{unitel1bf}{\unitel}{\let#1=\mafx@unitelbf}
498 \define@key{mafx@}{unitel1bfb}{\unitel}{\let#1=\mafx@unitelbfb}
499 \define@key{mafx@}{unitel1bfbx}{\unitel}{\let#1=\mafx@unitelbfbx}
500 \define@key{mafx@}{unitel1*}{\unitel}{\let#1=\mafx@unitels}
501 \define@key{mafx@}{unitele}{\unitel}{\let#1=\mafx@uniteleup}
502 \define@key{mafx@}{unitele*}{\unitel}{\let#1=\mafx@uniteleit}
503 \define@key{mafx@}{unitelE}{\unitel}{\let#1=\mafx@unitelEup}
504 \define@key{mafx@}{unitelE*}{\unitel}{\let#1=\mafx@unitelEit}
505 \define@key{mafx@}{unitelI}{\unitel}{\let#1=\mafx@unitelIup}
506 \define@key{mafx@}{unitelI*}{\unitel}{\let#1=\mafx@unitelIit}
507 \define@key{mafx@}{unitelid}{\unitel}{\let#1=\mafx@unitelid}

```

Derivative Symbols. Define macros for derivative symbols:

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

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

```
512 \define@key{mafx@}{der}{\der}{\let#1=\mafx@derup}
513 \define@key{mafx@}{der*}{\der}{\let#1=\mafx@derit}
514 \define@key{mafx@}{diff}{\diff}{\let#1=\mafx@diffup}
515 \define@key{mafx@}{diff*}{\diff}{\let#1=\mafx@diffit}
```

Assorted Symbols. Define macros for assorted symbols:

```
516 \DeclareRobustCommand{\mafx@order}{\mathnormal{o}}
517 \DeclareRobustCommand{\mafx@Order}{\mathnormal{O}}
518 \DeclareRobustCommand{\mafx@OrderCal}{\mathcal{O}}
519 \DeclareRobustCommand{\mafx@defeq}{\mathrel{\mathop{=}}}
520 \DeclareRobustCommand{\mafx@eqdef}{=\mathrel{\mathop{=}}}
521 \DeclareRobustCommand{\mafx@lapl}{\mathnormal{\Delta}}
522 \DeclareRobustCommand{\mafx@dualop}{\star}
```

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

```
523 \define@key{mafx@}{order}{\order}{\let#1=\mafx@order}
524 \define@key{mafx@}{Order}{\Order}{\let#1=\mafx@Order}
525 \define@key{mafx@}{Order*}{\Order}{\let#1=\mafx@OrderCal}
526 \define@key{mafx@}{defeq}{\defeq}{\let#1=\mafx@defeq}
527 \define@key{mafx@}{eqdef}{\eqdef}{\let#1=\mafx@eqdef}
528 \define@key{mafx@}{lapl}{\lapl}{\let#1=\mafx@lapl}
529 \define@key{mafx@}{dualop}{\dualop}{\let#1=\mafx@dualop}
```

Number Sets.

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

```
530 \ifdefined\mathbb
531 \let\mafx@numset@font=\mathbb
532 \else
533 \let\mafx@numset@font=\mathbf
534 \fi
535 \DeclareRobustCommand{\mafx@numset}{\mafx@numset@font}
536 \DeclareRobustCommand{\mafx@numsetZ}{\mafx@numset{Z}}
537 \DeclareRobustCommand{\mafx@numsetQ}{\mafx@numset{Q}}
538 \DeclareRobustCommand{\mafx@numsetR}{\mafx@numset{R}}
539 \DeclareRobustCommand{\mafx@numsetC}{\mafx@numset{C}}
540 \DeclareRobustCommand{\mafx@numsetH}{\mafx@numset{H}}
541 \DeclareRobustCommand{\mafx@numsetN}{\mafx@numset{N}}
542 \DeclareRobustCommand{\mafx@numsetO}{\mafx@numset{O}}
```

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

```
numsetfont 543 \define@key{mafx@}{numset}{\numset}{\let#1=\mafx@numset}
numsets    544 \define@key{mafx@}{numsetfont}{\let\mafx@numset@font=#1}
          545 \define@key{mafx@}{numsets}[]{\%
          546 \let\Integer=\mafx@numsetZ
          547 \let\Rational=\mafx@numsetQ}
```

```

548 \let\Real=\mafx@numsetR
549 \let\Complex=\mafx@numsetC
550 \let\Quaternion=\mafx@numsetH
551 \let\Natural=\mafx@numsetN
552 \let\Octonion=\mafx@numsetO
553 }

```

Groups.

\grp Define macros for Lie groups. Use oeprator font \operator@font:

```
554 \DeclareRobustCommand{\mafx@grp}[1]{\mafx@mathopfont{#1}}
```

Define macros for basic finite groups. Use oeprator font \operator@font:

```

555 \DeclareRobustCommand{\mafx@grp@perm}{\mafx@grp{S}}
556 \DeclareRobustCommand{\mafx@grp@alt}{\mafx@grp{A}}
557 \DeclareRobustCommand{\mafx@grp@dih}{\mafx@grp{D}}
558 \DeclareRobustCommand{\mafx@grp@cycZ}{\mafx@grp{Z}}
559 \DeclareRobustCommand{\mafx@grp@cycC}{\mafx@grp{C}}

```

Define macros for groups of Lie type:

```

560 \DeclareRobustCommand{\mafx@grp@GL}{\mafx@grp{GL}}
561 \DeclareRobustCommand{\mafx@grp@SL}{\mafx@grp{SL}}
562 \DeclareRobustCommand{\mafx@grp@PGL}{\mafx@grp{PGL}}
563 \DeclareRobustCommand{\mafx@grp@PSL}{\mafx@grp{PSL}}
564 \DeclareRobustCommand{\mafx@grp@U}{\mafx@grp{U}}
565 \DeclareRobustCommand{\mafx@grp@SU}{\mafx@grp{SU}}
566 \DeclareRobustCommand{\mafx@grp@PU}{\mafx@grp{PU}}
567 \DeclareRobustCommand{\mafx@grp@PSU}{\mafx@grp{PSU}}
568 \DeclareRobustCommand{\mafx@grp@O}{\mafx@grp{O}}
569 \DeclareRobustCommand{\mafx@grp@SO}{\mafx@grp{SO}}
570 \DeclareRobustCommand{\mafx@grp@PSO}{\mafx@grp{PSO}}
571 \DeclareRobustCommand{\mafx@grp@Spin}{\mafx@grp{Spin}}
572 \DeclareRobustCommand{\mafx@grp@Pin}{\mafx@grp{Pin}}
573 \DeclareRobustCommand{\mafx@grp@Sp}{\mafx@grp{Sp}}
574 \DeclareRobustCommand{\mafx@grp@Mp}{\mafx@grp{Mp}}

```

Define macros for groups related to Dynkin diagrams:

```

575 \DeclareRobustCommand{\mafx@grp@A}{\mafx@grp{A}}
576 \DeclareRobustCommand{\mafx@grp@B}{\mafx@grp{B}}
577 \DeclareRobustCommand{\mafx@grp@C}{\mafx@grp{C}}
578 \DeclareRobustCommand{\mafx@grp@D}{\mafx@grp{D}}
579 \DeclareRobustCommand{\mafx@grp@E}{\mafx@grp{E}}
580 \DeclareRobustCommand{\mafx@grp@F}{\mafx@grp{F}}
581 \DeclareRobustCommand{\mafx@grp@G}{\mafx@grp{G}}

```

grp Implement Lie group and a collection of standard Lie groups by name:

```

grpfin 582 \define@key{mafx@}{grp}[grp]{\let#1=\mafx@grp}
grpcycC 583 \define@key{mafx@}{grpfin}[]{}
grplie 584 \let\grp@perm=\mafx@grp@perm
grpdynkin 585 \let\grp@alt=\mafx@grp@alt
586 \let\grp@dih=\mafx@grp@dih
587 \let\grp@cyc=\mafx@grp@cycZ
588 }
589 \define@key{mafx@}{grpfin*}[]{}
590 \let\grp@perm=\mafx@grp@perm

```

```

591 \let\grpalt=\mafx@grp@alt
592 \let\grpdih=\mafx@grp@dih
593 \let\grpcyc=\mafx@grp@cycC
594 }
595 \define@key{mafx@}{grpcycC}[\grpcyc]{\let#1=\mafx@grp@cycC}
596 \define@key{mafx@}{grplie}[]{}
597 \let\grpGL=\mafx@grp@GL
598 \let\grpSL=\mafx@grp@SL
599 \let\grpPGL=\mafx@grp@PGL
600 \let\grpPSL=\mafx@grp@PSL
601 \let\grpU=\mafx@grp@U
602 \let\grpSU=\mafx@grp@SU
603 \let\grpPU=\mafx@grp@PU
604 \let\grpPSU=\mafx@grp@PSU
605 \let\grpO=\mafx@grp@U
606 \let\grpSO=\mafx@grp@SO
607 \let\grpPSO=\mafx@grp@PSO
608 \let\grpSpin=\mafx@grp@Spin
609 \let\grpPin=\mafx@grp@Pin
610 \let\grpSp=\mafx@grp@Sp
611 \let\grpMp=\mafx@grp@Mp
612 }
613 \define@key{mafx@}{grpdynkin}[]{}
614 \let\grpA=\mafx@grp@A
615 \let\grpB=\mafx@grp@B
616 \let\grpC=\mafx@grp@C
617 \let\grpD=\mafx@grp@D
618 \let\grpE=\mafx@grp@E
619 \let\grpF=\mafx@grp@F
620 \let\grpG=\mafx@grp@G
621 }

```

Conjugation Operations.

`\trans` Define macros for conjugation operations:

```

\hconj
\conj 622 \def\mafx@miniscript#1{\mathchoice%
623   {\textstyle#1}{\textstyle#1}{\scriptscriptstyle#1}{\scriptscriptstyle#1}}
624 \let\mafx@trans@font=\mathsf
625 \DeclareRobustCommand{\mafx@trans}{\mafx@trans@font{t}}
626 \DeclareRobustCommand{\mafx@transT}{\mafx@trans@font{T}}
627 \DeclareRobustCommand{\mafx@transTs}{\mafx@trans@font{\mafx@miniscript T}}
628 \DeclareRobustCommand{\mafx@hconj}{\mafx@trans@font{h}}
629 \DeclareRobustCommand{\mafx@hconjH}{\mafx@trans@font{H}}
630 \DeclareRobustCommand{\mafx@hconjHs}{\mafx@trans@font{\mafx@miniscript H}}
631 \DeclareRobustCommand{\mafx@trans}{\top}
632 \DeclareRobustCommand{\mafx@transi}{\intercal}
633 \DeclareRobustCommand{\mafx@hconj}{\dagger}
634 \DeclareRobustCommand{\mafx@conj}{\ast}
635 \DeclareRobustCommand{\mafx@dual}{\ast}

```

`trans` Implement conjugation operations individually with the option to adjust their macro

`hconj` names:

```

\conj 636 \define@key{mafx@}{transfont}{\let\mafx@trans@font=#1}
637 \define@key{mafx@}{trans}[\trans]{\let#1=\mafx@trans}
638 \define@key{mafx@}{trans*}[\trans]{\let#1=\mafx@transi}
639 \define@key{mafx@}{transT}[\trans]{\let#1=\mafx@transT}

```



```

640 \define@key{mafx@}{transT*}[\trans]{\let#1=\mafx@transTs}
641 \define@key{mafx@}{transt}[\trans]{\let#1=\mafx@transt}
642 \define@key{mafx@}{hconj}[\hconj]{\let#1=\mafx@hconj}
643 \define@key{mafx@}{hconjH}[\hconj]{\let#1=\mafx@hconjH}
644 \define@key{mafx@}{hconjH*}[\hconj]{\let#1=\mafx@hconjHs}
645 \define@key{mafx@}{hconjh}[\hconj]{\let#1=\mafx@hconjh}
646 \define@key{mafx@}{conj}[\conj]{\let#1=\mafx@conj}
647 \define@key{mafx@}{dual}[\dual]{\let#1=\mafx@dual}

```

Package Options.

`\ProvideMathFix` Implement a particular fix or option:

```

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

```

Pass undeclared options on to keyval processing:

```

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

```

Process package options:

```

650 \ProcessOptions

```