The physics3 package*

Mingyu Xia†

Released 2026-02-07 v0.2A

Abstract

This is the document for physics3 package, which defines commands for typesetting math formulae faster and more simply. physics3 is a modularized package, each module provides its own function[1].

Contents

*https://github.com/myhsia/physics3, https://ctan.org/pkg/physics3

†xiamingyu@westlake.edu.cn

[1] If you are a user of the legacy physics package, see Section A before you start.

1 Introduction

The physics3 package aims to provide a bundle of commands for typesetting math faster in different modules. The commands provided by physics3 and its different modules are designed to be short and easy to memorize.

1.1 Packages required

The physics3 package itself only requires the keyval package, which is part of the latex-graphics bundle. Almost every LaTeX distribution will include this bundle.

Different modules of physics3 might require different packages. It will be explained in the following sections that which module requires which package.

The physics3 package requires LaTeX 2_ε kernel released after 2020/10. Please make sure that your LaTeX distribution is not too old.

1.2 Loading physics3 and its modules

Just like loading any package, write

```
\usepackage{physics3}
```

in the preamble to load the physics3 package. In the current version, physics3 doesn't provide a package option.

physics3 itself doesn't provide many features. You need to load different modules of physics3 by using the `\usephxmodu` command only *after* loading this package in the preamble to have different features applied to your document.

`\usephxmodu` `\usephxmodu [⟨options⟩] {⟨module⟩}`

The usage of `\usephxmodu` is similar to `\usepackage`, so you can load several modules in one line. For example,

```
\usephxmodu [tightbraces = true] {ab}
\usephxmodu {ab.braket, doubleprod}
```

The options of a physics3 module can be a comma-separated key-value list. These two lines load the `ab` module with option `tightbraces = true` and load `ab.braket` and `doubleprod` modules.

The following section introduce all the user-level modules of physics3. The modules have similar names like ⟨*module*⟩`.legacy` are designed to provide solutions to maintain documents written with the legacy physics package. It's not suggested to use them in a new document.

2 Modules of physics3

2.1 Features of the bare physics3 package

The following commands are available once you load physics3 in preamble.

\delopen
\delclose

`\delopen` ⟨*math delimiter*⟩ ... `\delclose` ⟨*math delimeter*⟩

`\delopen` and `\delclose`, followed by a math delimiter. They can be regarded as abbreviations of "open delimiter" and "close delimiter". If you had heard of the mleftright package. You can regard `\delopen` and `\delclose` as a simpler version of `\mleft` and `\mright`.

For example,

```
$ 0 \left(\frac12\right) 3 $\quad
$ 0 \delopen(\frac12\delclose) 3 $
```

$$0\left(\frac{1}{2}\right)3 \quad 0\left(\frac{1}{2}\right)3$$

`\biggg` and `\Biggg`, followed by a math delimiter. They are even bigger than `\Bigg`. `\biggg` and `\Biggg` may be useful when you need to write something really tall in math mode, but most OpenType math font do not support `\langle` (or U+27E8) and `\rangle` (or U+27E9) in this large size. Take an example,

```
$ \Biggg(\biggg(\Bigg(\bigg(
  \Big(\big(()\big)\Big)
  \bigg)\Bigg)\biggg)\Biggg) $
```

$$\Biggg(\biggg(\Bigg(\bigg(\Big(\big(()\big)\Big)\bigg)\Bigg)\biggg)\Biggg)$$

`\bigggl`, `\bigggm`, `\bigggr`, `\Bigggl`, `\Bigggm` and `\Bigggr` are also supported.

2.2 The ab module — automatic braces

This module provides the command `\ab`. The `\ab` command, as a shorthand of "automatic braces", would specify the size of the following pair of delimiters. The delimiters after `\ab` should not be out of the range described by the following chart.

\ab

```
\ab ( ... )     \ab [ ... ]
\ab\{ ... \} or \ab \lbrace ...  \rbrace
\ab < ... >  or \ab \langle ...  \rangle
\ab | ... |  or \ab \vert   ...  \vert
\ab\| ... \| or \ab \Vert   ...  \Vert
```

For example, it's illegal to write an "`\ab(`" without a "`)`"; it's also illegal to write `\ab=foo=`. Take some correct examples:

```
$ \ab ( \frac12 )  $ \quad
$ \ab [ \frac12 ]  $ \quad
$ \ab\{ \frac12 \} $
```

$$\left(\frac{1}{2}\right) \quad \left[\frac{1}{2}\right] \quad \left\{\frac{1}{2}\right\}$$

You can also write a command from `\big` to `\Biggg` between `\ab` and the first delimiter, which means to specify the size of delimiters manually. Also, you can write a star (`*`) between `\ab` and the first delimiter, to prevent `\ab` from setting the size of delimiters. For example,

```
$ \ab      <\frac12>  $\quad
$ \ab\biggg|\frac12|  $\quad
$ \ab*    \|\frac12\| $
```

$$\left\langle\frac{1}{2}\right\rangle \quad \left|\frac{1}{2}\right| \quad \|\frac{1}{2}\|$$

Always remember, do not put an `\ab` separately at the end of math mode like `$\ab$`, because `\ab` will try to absorb the following math shift character (`$`) as its argument.

TEXhackers note: The `ab` module uses "document commands" module of LaTeX 2_ε kernel (source file: ltcmd.dtx). This LaTeX 2_ε kernel module provides a document-level command parser. `\ab` is a complex encapsulation of some internal document-level commands. Take an example, if you define a document-level command like this:

```
\NewDocumentCommand \foo { r() } {::#1::}
```

You can write `\foo(bar)` legally, but `\foo(()` will be regarded illegal when you write another document-level command or end the paragraph. Similarly, things like `\ab(()` will also cause errors.

The `ab` module also provides $\backslash X$`ab` commands, where X can be `p`, `b`, `B`, `a`, `v` and `V`. These commands take a normal argument but not an argument delimited with paired delimiters. For example,

```
\def\0{\frac12}
$ \pab{\0} $ $ \bab{\0} $ $ \Bab{\0} $
$ \aab{\0} $ $ \vab{\0} $ $ \Vab{\0} $
```

$$\left(\frac{1}{2}\right) \left[\frac{1}{2}\right] \left\{\frac{1}{2}\right\} \left\langle\frac{1}{2}\right\rangle \left|\frac{1}{2}\right| \left\|\frac{1}{2}\right\|$$

These $\backslash X$`ab` commands can take an optional star and an optional `[`$\langle$*biggg*$\rangle$`]` argument. Star stands for using the default sizes. For example,

```
\def\0{n+\frac12}
$ \pab[Big]{\0} $\quad $ \bab*{\0} $
```

$$\left(n+\frac{1}{2}\right) \quad \left[n+\frac{1}{2}\right]$$

`tightbraces` = $\langle$**true**|*false*$\rangle$: Influences whether thin skips are reserved around the paired delimiters. It only works with the automatically sized delimiters.

2.3 The ab.braket and braket module — Dirac bra-ket notation

This two modules both contain four basic commands

\bra, \ket, \braket, \ketbra

These commands in the two modules share the same goal, but differ in syntax.

ab.braket A star (*) or a size command can follow these commands, which are similar to the syntaxes of \ab module. The size commands can take the control sequences:

\big, \Big, \bigg, \Bigg, \biggg or \Biggg.

braket A star (*) or a square-bracket-delimited size option, the size option can take the following values:

big, Big, bigg, Bigg, biggg or Biggg.

The star in the two modules stands for "do not size the bra-ket automatically". Four basic commands' syntaxes are described as follows respectively. **Please notice that the two module are conflict with each other, so don't use them together.** One can choose one module according to personal habit.

\bra	\bra ⟨* or size command⟩ <⟨subformula⟩|	--> ab.braket module
\ket	\ket ⟨* or size command⟩ |⟨subformula⟩>	--> ab.braket module
	\bra ⟨* or [⟨size option⟩]⟩ {⟨subformula⟩}	--> braket module
	\ket ⟨* or [⟨size option⟩]⟩ {⟨subformula⟩}	--> braket module

- In ab.braket module, the argument of \bra should be delimited with < and |, and the argument of \ket should be delimited with | and >.

If you want to write "$>$" and "$<$" for relations in the argument of \bra and \ket, you can write \mathrel{>} and \mathrel{<}.

- In braket module, \bra and \ket take one mandatory argument, which should be braced with { and }.

For example, in ab.braket module,

```
\def\0{\frac\phi2}
$ \bra<\0| $\quad $ \bra*<\0|      $%
           \quad $ \bra\Big<\phi| $\\[1ex]
$ \ket|\0> $\quad $ \ket*|\0>      $%
           \quad $ \ket\Big|\psi> $
```

$$\left\langle \frac{\phi}{2} \right| \quad \langle \frac{\phi}{2} | \quad \left\langle \phi \right|$$

$$\left| \frac{\phi}{2} \right\rangle \quad | \frac{\phi}{2} \rangle \quad \left| \psi \right\rangle$$

and in braket module

```
\def\0{\frac\phi2}
$ \bra  \0 $\quad $ \bra*     \0   $%
           \quad $ \bra[Big] \psi $\\[1ex]
$ \ket  \0 $\quad $ \ket*     \0   $%
           \quad $ \ket[Big] \psi $
```

$$\left\langle\frac{\phi}{2}\right| \quad \langle\frac{\phi}{2}| \quad \left\langle\psi\right|$$
$$\left|\frac{\phi}{2}\right\rangle \quad |\frac{\phi}{2}\rangle \quad \left|\psi\right\rangle$$

\braket `\braket` ⟨* or size command⟩ <⟨subformula⟩> --> ab.braket module
`\braket` ⟨* or [⟨size option, i⟩]⟩ i*{⟨subformula⟩} --> braket module

---

- In `ab.braket` module, the argument of `\braket` should be delimited with `<` and `>`, every "`|`" will be regarded as an extensible vertical bar in the ⟨*subformula*⟩ argument. If you want to write "$>$" and "$<$" for relations in the argument of `\braket` and `\ketbra` in this module, you can write `\>` and `\<`. It is quite different from `\mathrel{>}` or `\mathrel{<}` because in these commands' argument, `>` and `<` will be redefined.

- In `braket` module, the `\braket` command, in default, can take two arguments. If you want `\braket` to take one or three arguments, you can specific the number of arguments by number `i` in the square bracket, separate from the size option with a comma.

For example, in `ab.braket` and `braket` module,

```
\def\0{\frac\phi2}
$ \braket     <   \hat A     >      $\quad
$ \braket     <\0|\hat A|\psi>      $\quad
$ \braket\big <\0|\psi>             $\\[1ex]
$ \braket*    <\0|\psi>             $\quad
$ \braket\Big <\0|\psi>             $\quad
$ \ab|{\braket<\psi|\hat A|\psi>}|  $
```

$$\left\langle\hat{A}\right\rangle \quad \left\langle\frac{\phi}{2}\middle|\hat{A}\middle|\psi\right\rangle \quad \langle\frac{\phi}{2}|\psi\rangle$$
$$\langle\frac{\phi}{2}|\psi\rangle \quad \left\langle\frac{\phi}{2}\middle|\psi\right\rangle \quad \left|\left\langle\psi\middle|\hat{A}\middle|\psi\right\rangle\right|$$

Commands from `ab.braket` should not be placed barely in `\ab|`⟨*subformula*⟩`|` or errors will arise. So, adding braces like the last line in the example above.

```
\def\0{\frac\phi2}
$ \braket  [1]     {\hat A}         $\quad
$ \braket  [3] {\0}{\hat A}{\psi}   $\quad
$ \braket[big] {\0}{\psi}           $\\[1ex]
$ \braket*     {\0}{\psi}           $\quad
$ \braket[Big] {\0}{\psi}           $\quad
$ \ab|{\braket[3]\psi{\hat A}\psi}| $
```

$$\left\langle\hat{A}\right\rangle \quad \left\langle\frac{\phi}{2}\middle|\hat{A}\middle|\psi\right\rangle \quad \langle\frac{\phi}{2}|\psi\rangle$$
$$\langle\frac{\phi}{2}|\psi\rangle \quad \left\langle\frac{\phi}{2}\middle|\psi\right\rangle \quad \left|\left\langle\psi\middle|\hat{A}\middle|\psi\right\rangle\right|$$

\ketbra `\ketbra` ⟨* or size command⟩ ⟨optional formula⟩ `\ketbra` ⟨* or [⟨size option⟩]⟩ ⟨optional formula⟩

`|`⟨subformula$_1$⟩`>` `<`⟨subformula$_2$⟩`|` `{`⟨subformula$_1$⟩`}` `{`⟨subformula$_2$⟩`}`

--> ab.braket module

--> braket module

- In `ab.braket` module, the argument of `\ketbra` should be delimited with `|` and `|`, `>` and `<` will be regarded as extensible ⟩ and ⟨ in the argument.
- In `braket` module, the `\ketbra` command takes two mandatory arguments by default, and one optional argument between the two mandatory arguments (⟩ and ⟨).

For example, in `ab.braket` and `braket` module,

```
\def\0{\frac\phi2}
$ \ketbra         | \0 >< \psi | $\quad
$ \ketbra*        | \0 >< \psi | $\quad
$ \ketbra\Big | \0 >_x^y< \psi | $
```

$\left|\frac{\phi}{2}\right\rangle\left\langle\psi\right|$ $|\frac{\phi}{2}\rangle\langle\psi|$ $\left|\frac{\phi}{2}\right\rangle_x^y\left\langle\psi\right|$

```
\def\0{\frac\phi2}
$ \ketbra        {\0} {\psi} $\quad
$ \ketbra*       {\0} {\psi} $\quad
$ \ketbra  [Big] {\0} [_x^y] {\psi} $
```

$\left|\frac{\phi}{2}\right\rangle\left\langle\psi\right|$ $|\frac{\phi}{2}\rangle\langle\psi|$ $\left|\frac{\phi}{2}\right\rangle_x^y\left\langle\psi\right|$

For convince, the two modules also contain three advanced commands

`\bknorm`, `\kbproj`, `\expval`

These commands have the same syntax in the two modules.

\bknorm `\bknorm` ⟨*⟩ [⟨size⟩] {⟨basis⟩}
\kbproj `\kbproj` ⟨*⟩ [⟨size⟩] {⟨basis⟩}
\expval `\expval` ⟨*⟩ [⟨basis, size⟩] {⟨operator⟩}

For example,

```
\def \0{\frac\alpha2}
\edef\1{\uparrow,\downarrow}
$ \bknorm \0 $\quad $ \bknorm [big]\0 $
              \quad $ \bknorm*\0 $\\[2ex]
$ \kbproj \0 $\quad $ \kbproj [big]\0 $
              \quad $ \kbproj*\0 $\\[2ex]
$ \expval [Big, \1] {\hat A} $\quad
$ \expval*[Big, \psi]\0 $
```

$\left\langle\frac{\alpha}{2}\middle|\frac{\alpha}{2}\right\rangle$ $\langle\frac{\alpha}{2}|\frac{\alpha}{2}\rangle$ $\langle\frac{\alpha}{2}|\frac{\alpha}{2}\rangle$

$\left|\frac{\alpha}{2}\right\rangle\left\langle\frac{\alpha}{2}\right|$ $|\frac{\alpha}{2}\rangle\langle\frac{\alpha}{2}|$ $|\frac{\alpha}{2}\rangle\langle\frac{\alpha}{2}|$

$\left\langle\uparrow,\downarrow\middle|\hat{A}\middle|\uparrow,\downarrow\right\rangle$ $\langle\psi|\frac{\alpha}{2}|\psi\rangle$

2.4 The doubleprod module — tensors' double product operator

This module provides the \doublecross and \doubledot commands, which are regarded as binary operators by TEX. Take an example of this module:

`$ A \doublecross B \doubledot C $`	$A \overset{\times}{\times} B : C$

crossscale, dotscale = ⟨*fp num*⟩: Scales of "×" and "·" (Default: .8 and 1).

crossopenup, dotopenup = ⟨*fp num*⟩: Spaces between "×"s and "·"s = ⟨*fp num*⟩ * font size (Default: .02 and .2).

doubledot, crosssymbol = ⟨*symbol*⟩: Pieces of \doublecross and \doubledot (Default: \times, \ldotp).

Their default values can be configured in module option. For example,

```
\usephxmodu[crossscale  = .75, dotscale  = 1.2,
            crossopenup = .05, dotopenup = .25]{doubleprod}
```

2.5 The diagmat module — simple diagonal matrices

This module requires the mathtools package and provides six *diagmat commands:

\diagmat
\pdiagmat
\bdiagmat
\Bdiagmat
\vdiagmat
\Vdiagmat

\⟨*delimiter type*⟩diagmat⟨***⟩ [⟨*options*⟩] {⟨*diag*⟩}

where ⟨*diag*⟩ is the diagonal of the diagonal matrix. The entries should be separated by commas. Prefixes like p, b, V have the same meanings as pmatrix, bmatrix and Vmatrix in the mathtools package, a star (*) follows the *diagmat commands correspond to *smallmatrix in the mathtools package.

empty = ⟨*token list*⟩: Value of \diagmat's empty entries (Default is blanck).

align = ⟨*l* | *c* | *r*⟩: Align of \diagmat's diagonal entries (Default: r).

Their default values can be configured in the module option like this:

```
\usephxmodu[empty = \cdot, align = r]{diagmat}
```

For example,

```
$ \pdiagmat [empty = \mathbf O] {
    \diagmat*[align = r]{1, \sqrt[3]4},
    \diagmat*{a, b} } $,
$ \bdiagmat* {1, -1, -1, -1} $
```

$$\begin{pmatrix} 1 & \mathbf{O} & \\ \mathbf{O} & \sqrt[3]{4} & \\ & & \begin{matrix} a & \mathbf{O} \\ \mathbf{O} & b \end{matrix} \end{pmatrix}, \quad \begin{bmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & -1 \end{bmatrix}$$

2.6 The xmat module — matrices with formatted entries

This module requires the mathtools package and provides six *xmat commands:

\xmat
\pxmat
\bxmat
\Bxmat
\vxmat
\Vxmat

\⟨*delimeter type*⟩xmat⟨***⟩ [⟨*options*⟩] {⟨*entry*⟩} {⟨*row label*⟩} {⟨*col label*⟩}

Prefixes like p, b, V have the same meanings as pmatrix, bmatrix and Vmatrix in the mathtools package, a star (*) follows the *xmat commands correspond to *smallmatrix in the mathtools package. For example,

```
$ \vxmat*{a}{2}{3} $
```

$$\left|\begin{smallmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{smallmatrix}\right|$$

align = ⟨*l* | *c* | *r*⟩: Similarly to the diagmat module.

If ⟨*rows/col label*⟩ contains non-digit characters, dots will be added. For example,

```
$ \bxmat*[showtop = 1, showleft = 2,
          align   = l]{X}{m}{n} $
```

$$\begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix}$$

showtop, **showleft** = ⟨*int num*⟩: Numbers of rows and columns to be shown at the top and the left side (Default: MaxMatrixCols - 2).

Their default values can be configured in the module option like this:

```
\usephxmodu[showtop = 1, showleft = 2]{xmat}
```

This will also influence "\xmat"s with digital ⟨*row label*⟩ and ⟨*col label*⟩ they are larger than the values corresponding to showtop and showleft. For example,

```
$ \pxmat*[showtop  = 1,
          showleft = 2]{A}{8}{8} $
```

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{18} \\ \vdots & \vdots & \ddots & \vdots \\ A_{81} & A_{82} & \cdots & A_{88} \end{pmatrix}$$

The format option allows users to use a new entry format.

format = ⟨*custom entry format in terms of #1, #2, and #3*⟩:

#1 : The common entry, or the first mandatory ⟨*entry*⟩ argument of \xmat;
#2 : The row index; #3 : The column index.

This option should be only used in the optional argument of \xmat. For example,

```
$ \Bxmat*[showtop = 2, showleft = 1,
          format = \texttt{#1[#2][#3]}]xmn $
```

$$\left\{\begin{matrix} \texttt{x[1][1]} & \cdots & \texttt{x[1][n]} \\ \texttt{x[2][1]} & \cdots & \texttt{x[2][n]} \\ \vdots & \ddots & \vdots \\ \texttt{x[m][1]} & \cdots & \texttt{x[m][n]} \end{matrix}\right\}$$

2.7 The operator module — log-like and nabla-related operators

This module provides a series of commands for log-like operators and some commands for nabla-related operators (∇)

```
\asin   \acos   \atan   \acsc   \asec   \acot   \erf
\erfc   \Tr     \tr     \rank   \Res    \res    \Re
\Im     \PV     \pv     \upe    \iu     \grad   \curl
\div    \laplacian      \identity       \Identity
```

- The first fifteen commands yield what they look like in math mode
- `\PV` yields "$\mathcal{P}$" as an ordinary symbol and `\pv` yields "$\mathrm{p.v.}$".
- `\Re` and `\Im` are redefined as "Re" and "Im". $\Re$ and $\Im$ are redefined as `\Resymbol` and `\Imsymbol`, in default.
- The "÷" symbol was redefined as `\divsymbol`.
- This module requires the fixdif package with file date `2023/01/31` at minimum.

ReIm = ⟨*true* | **false**⟩: Determines whether to redefine `\Re` and `\Im`. If you want to reserve the definition of `\Re` and `\Im`, you can write like this:

```
\usephxmodu[ReIm = false]{operator}
```

This module also provides the `\identity` and `\Identity` commands for inputting the double-stroke "ı" (1) and "I" (I) respectively.

`\identity` `\identity [⟨font family⟩]` `\Identity [⟨font family⟩]`
`\Identity`

The optional argument can config the font for the double-stroke glyphs, with the default value `bbold`. One can also config it with the module option.

identity = ⟨*font family*⟩: Default font family for the double-stroke `\identity` and `\Identity`.

For example,

```
% \usephxmodu[identity = bbold]{operator}
% \usephxmodu{braket}
$ \asin x $, $ \PV f(z) = \pv f(z)  $\\[1ex]
$ \erfc(x) = 1 - \erf(x)            $\\[1ex]
$ \identity = \Identity[bbold] = \int
  \d^2 \alpha \kbproj \alpha/\pi    $\\[1ex]
$ \Re z = \Im z = \exp(\iu\pi) = -1 $\\[1ex]
$ \grad V$, $\div (x,y,z)$, $ \curl(x,y,z) $
```

$$\asin x,\ \mathcal{P} f(z) = \mathrm{p.v.}\, f(z)$$
$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$$
$$1 = \mathbb{I} = \int \mathrm{d}^2\alpha |\alpha\rangle\langle\alpha| / \pi$$
$$\mathrm{Re}\, z = \mathrm{Im}\, z = \exp(\mathrm{i}\pi) = -1$$
$$\nabla V,\ \nabla \cdot (x, y, z),\ \nabla \times (x, y, z)$$

2.8 The `ab.legacy` module

\abs
\norm
\eval
\peval
\beval
\order

`\⟨cmd⟩*[⟨biggg⟩]{⟨subformula⟩}.`

This module provides the left commands. They share the same syntax as Star and ⟨*biggg*⟩ are optional. Star stands for "use the default size". For example,

```
\def\0{1+\frac12}
$ \abs\0 $, $ \norm[Big]\0 $, $ \order*\0 $
\def\0{1+\frac x2}
$ \eval      {\0}_a^b $,
$ \peval*    {\0}_a^b $,
$ \beval[big]{\0}_a^b $
```

$$\left|1+\frac12\right|, \left\|1+\frac12\right\|, \mathcal{O}(1+\frac12)$$
$$1+\frac{x}{2}\Big|_a^b, (1+\frac{x}{2}|_a^b, [1+\frac{x}{2}|_a^b$$

You can set the "order" symbol in this module through the `order` option like this:

```
\usephxmodu[order = O]{ab.legacy}
```

Then `$\order{N}$` yields $O(N)$.

2.9 The `qtext.legacy` module

The `qtext.legacy` module provides the `\q`⟨*foo*⟩ commands for `\quad`-wrapped texts. These commands have the same syntax as physics. For example,

```
$ A \qq {foo bar} B $\\
$ A \qq*{foo bar} B $\\
$ C \qcc D \qcc*  E $\\
$ F \qif G \qthen H $
```

A foo bar B
Afoo bar B
C c.c Dc.c E
F if G then H

All the commands described in §2.4 of physics documentation are supported when using `qtext.legacy` module, but I don't recommend using this module unless you are maintaining a document written with physics's `\q`⟨*foo*⟩ commands.

A For the legacy physics users

This section describes physics3 package for those who are used to the physics package, which is only a simple reference manual for:

- Frequent users of the legacy physics package;
- Those who have to maintain a document written with physics;
- Users who failed to use unicode-math with physics.

In this section, the modules of physics3 will be introduced in the same order as the physics documentation.

A.1 Legacy problems with physics package

The physics package provides `\qty` command for automatic-sizing braces. The `\qty` command would cause conflict with the siunitx package, which provides a unified method to typeset numbers and units correctly.

Besides, after you loaded physics, when you type `\homework` you will get Maxwell equations and Schrödinger equation. The `\homework` command is "declared" in `physics.sty` but it was not described in the documentation. That is, if you have defined `\homework` before loading physics package, physics would overwrite the definition "silently".

The vector-notation part of physics uses amsmath's (more exactly, `amsbsy.sty`'s) `\boldsymbol` command to generate bold vectors. Commands for cross/dot product are defined with `\boldsymbol`. `\boldsymbol` uses `\mathversion`, a LaTeX 2_ε kernel command that works well with traditional TFM-based fonts but fails when using unicode-math.

In the definition of `\imat`, `\xmat`, `\dmat` and `\admat` commands from physics, there is a `\newtoks` command which allocates a token list register and two `\newcount` commands allocating two count registers. Every time you write a command like `\imat` in your document, then one token list register and two count registers will be wasted. What's even worse is that, if you wrote really too many matrix commands from physics (for example, 32767 `\imats` in LuaLaTeX), there'd be no room for a new `\count`.

physics integrated all the functions in one file (`physics.sty`), that is, you cannot load one of the total seven parts of functions; you have to load the seven parts altogether, even included the extra `\homework` command we mentioned in the first paragraph.

Moreover, the code of `physics.sty` "abuses" the g-type arguments of xparse package. Therefore the syntax of physics package looks kind of weird. See here for more.

A.2 Loading physics3

The physics3 package includes different modules, among which every module focuses on one single function.

Write the following line in the preamble to load physics3:

```
\usepackage{physics3}
```

But this is not enough. physics3 contains different modules. If you want to load any module of physics3, write this line after loading physics3 package:

```
\usephxmodu{⟨module list⟩}
```

For example, "`\usephxmodu{ab, doubleprod}`" loads the `ab` module and the `doubleprod` module.

You can also load a module with options:

```
\usephxmodu[⟨option list⟩]{⟨module⟩}
```

For example, "`\usephxmodu[legacy]{ab}`" loads `ab` with the option "`legacy`".

Attention, if you used any font package in your document, remember that physics3 requires to be loaded *after* font packages.

A.3 List of commands

A.3.1 Automatic bracing

As mentioned in §A.1, the `\qty` command from physics will conflict with siunitx. The command for automatic braces in physics3 is `\ab`, short for **a**utomatic **b**races.

physics also provides the following commands:

```
\abs  \norm  \eval  \order  \comm  \acomm  \pb
```

These commands are not originally supported by physics3, but the first four commands can be used through the `ab.legacy` module of physics3:

```
\usephxmodu{ab.legacy}
```

Users of the legacy physics package should notice that the syntax of `\eval` has been changed. The `ab.legacy` module abandoned the `\eval(foo|`-like syntax. The new `\eval`'s syntax is just like other commands in this module. There are also two variants of `\eval` — `\peval` and `\beval`.

The `\comm`, `\acomm` and `\pb` (Poisson bracket) are not supported. But you can write like `\ab[foo,baz]` or `\bab{foo,baz}` instead.

By the way, you can set the "order" symbol in `ab.legacy` through the `order` option like this:

```
\usephxmodu[order = O]{ab.legacy}
```

Then `$\order(N)$` yields $O(N)$.

A.3.2 Vector notation

Unfortunately, there is not a plan for physics3 to support this part of physics completely, but the rest of this section will show some methods to maintain the document written with physics.

The `\vb(*)`, `\va(*)` and `\vu(*)` are not supported in any module of physics3. But these commands can be defined by copying the following lines below and pasting them in the preamble:

```
\makeatletter
\newcommand\vb{\@ifstar\boldsymbol\mathbf}
\newcommand\va[1]{\@ifstar{\vec{#1}}{\vec{\mathrm{#1}}}}
\newcommand\vu[1]{%
\@ifstar{\hat{\boldsymbol{#1}}}{\hat{\mathbf{#1}}}}
\makeatother
```

The `\boldsymbol` command requires the amsmath or bm package. If you prefer to use bm, you can also use the `\bm` command. What's more, if you tried the commands above, you might find that, the result of `\va` above is different from that of physics. This is because, if you choose to present a vector in bold, there's almost no need to put a `\vec` ($\vec{}$) sign above it.

However, the method above may not work well with unicode-math because there are so many OpenType math fonts without a bold version. When using unicode-math, it's recommended to use `\symbf` and `\symbfit` for a separate vector[2].

The `\vdot` and `\cross` commands are not supported in any module of physics3. Actually, there is no need to use a bold "$\cdot$" or "$\times$" for the products of two vectors. Using `\cdot` and `\times` is enough.

The commands related to "∇" are supported through `operator` module. These commands are `\grad`, `\div` and `\curl`. These commands should not be put in the end

[2] See Issue #1991 in the LaTeX 2_ε repository on GitHub, this bug will be resolved someday.

of a math formula either (just like `\ab`). Notice that the former `\div` command for a "÷" (if there exists one) is redefined as `\divsymbol`.

Actually, the nabla-related commands end with `\ab`. Thus, the subformula after these commands can be delimited with `()`, `[]` and `\{\}`.

The `operator` requires the fixdif package at least version 2.0 (file date on or after 2023/01/31).

A.3.3 Operators

There's no plan for physics3 to support this part of physics completely. The syntax in this part of physics (such as `\tan[2](x)`) abuses xparse.

It's suggested to write like this if you used the `ab` module:

```
$ \sin^2 \ab( \frac{\alpha}{2} ) $
```

Rather than take the superscript as an optional argument of the command of log-like functions.

The physics package provides a bundle of commands for log-like functions that have not been defined in the LaTeX 2_ε kernel. They can be used with the `operator` module; this module does not support the syntax of physics either.

The `\Re` and `\Im` commands are redefined as operators "Re" and "Im", while $\Re$ and $\Im$ are reserved as `\Resymbol` and `\Imsymbol`. $\Re$ and $\Im$ are ordinary symbols but Re and Im are operators.

A.3.4 Quick quad text

All the commands described in §2.4 of physics documentation are supported when using `qtext.legacy` module. Click here to see the `qtext.legacy` module.

A.3.5 Derivatives

There is no plan for physics3 to support this part of physics. If you want to typeset the differential operators on a better sense, you can try the fixdif package; if you want an easy way to type derivatives, you can try the derivative package. These two packages can be used together. For example,

```
% \usepackage{fixdif, derivative}
$ \pdv{f}{x,y,z} \d x $\\[1ex]
Math ($\d x$) v.s.\ Text (\d x)
```

$$\frac{\partial^3 f}{\partial x\,\partial y\,\partial z}\,\mathrm{d}x$$

Math ($\mathrm{d}x$) v.s. Text (x̣)

Here are the documentations of fixdif and derivative. fixdif's commands behave better in superscripts and subscripts.

A.3.6 Dirac bra-ket notation

There are two solutions to Dirac bra-ket in physics3 — `ab.braket` and `braket`. These two modules are *not* compatible and neither of them supports physics's syntax completely. Click here to see the `ab.braket` module and here to see the `braket` module.

A.3.7 Matrix macros

Unfortunately, physics3 do not support the `\mqty` command from physics. If you are used to this command, you can write like this:

```
\newcommand\mqty[1]{\begin{matrix}#1\end{matrix}}
\newcommand\pmqty[1]{\begin{pmatrix}#1\end{pmatrix}}
$\ab(\mqty{foo})$ or $\pmqty{foo}$
```

These are equal to physics's `\mqty(foo)` (require amsmath).

physics3's `diagmat` module provides `\diagmat` command for diagonal matrices. `\pdiagmat`, `\bdiagmat`, `\Bdiagmat`, `\vdiagmat` and `\Vdiagmat` are also available.

physics3's `xmat` module provides `\xmat` command for matrices with formatted entries. `\pxmat`, `\bxmat`, `\Bxmat`, `\vxmat` and `\Vxmat` are also available.

B Code Implementation

B.1 The physics3 package

```
⟨*package⟩
```

We use phx as the namespace for physics3 modules.

```
⟨@@=phx⟩
\NeedsTeXFormat{LaTeX2e}[2020/10/01]
\def \phx@date    {2026-02-07}
\def \phx@version {v0.2A}
\ProvidesExplPackage {physics3} {\phx@date} {\phx@version}
  {Tools for typesetting math for physics.}
```

B.1.1 Common variables

\l__phx_tmpa_box

\phx@temp⟨*register type*⟩⟨*a or b*⟩

Some LaTeX 2_ε variables starting with "\phx@temp". These variables can be shared by any module of physics3.

```
\box_new:N \l__phx_tmpa_box
```

(*End of definition for* \l__phx_tmpa_box.)

B.1.2 Package requirements and module-loading methods

physics3 requires keyval (part of the graphics bundle) to process options of modules.

```
\RequirePackage{keyval}
\def\phx@true{true}
\def\phx@false{false}
```

__phx_define_key:nnnn

\phx@define@key {⟨*module*⟩} {⟨*key*⟩}[⟨*default value*⟩] {⟨*code*⟩}

__phx_setkeys:nn

\phx@setkeys {⟨*module*⟩} {⟨*key-val list*⟩}

__phx_processkeyopt:n

\phx@processkeyopt {⟨*module*⟩}

The position of \phx@processkeyopt in a physics3 module is just the same as the position of \ProcessOptions in a regular LaTeX package.

```
\cs_new_nopar:Npn \__phx_define_key:nnnn #1#2#3#4
  { \define@key {phx-#1} {#2} [#3] {#4} }
\cs_new_nopar:Npn \__phx_setkeys:nn     #1#2
  { \setkeys    {phx-#1} {#2}           }
\cs_new:Npn       \__phx_processkeyopt:n #1
  {
    \let\reserved@a\@empty
    \edef\reserved@a{\@ptionlist{\@currname.\@currext}}%
```

```
    \edef\reserved@a{\noexpand\phx@setkeys{#1}{\reserved@a}}%
    \reserved@a% the next line thanks to 'geometry'
    \AtEndOfPackage{\let\@unprocessedoptions\relax}
  }
\cs_set_eq:NN \phx@define@key    \__phx_define_key:nnnn
\cs_set_eq:NN \phx@setkeys       \__phx_setkeys:nn
\cs_set_eq:NN \phx@processkeyopt \__phx_processkeyopt:n
```

(*End of definition for* `\__phx_define_key:nnnn`, `\__phx_setkeys:nn`, *and* `\__phx_processkeyopt:n`.)

We use almost the same way to load physics3 modules as LaTeX 2_ε kernel does. We use a lot of kernel commands in LaTeX 2_ε.

\usephxmodu
\RequirePhxmodu

`\usephxmodu [`⟨*key-val options*⟩`]{`⟨*module*⟩`} [`⟨*key-val options*⟩`]`
`\RequirePhxmodu [`⟨*key-val options*⟩`]{`⟨*module*⟩`} [`⟨*key-val options*⟩`]`

`\usephxmodu` is a user command, and `\RequirePhxmodu` is a developer command.

```
\def\usephxmodu{\phx@FWoptions\@pkgextension}
\let\RequirePhxmodu\usephxmodu
\@onlypreamble\usephxmodu
\def\phx@FWoptions#1{\@ifnextchar[%]
  {\phx@FW@ptions#1}{\phx@FW@ptions#1[]}}
\@onlypreamble\phx@FWoptions
\def\phx@FW@ptions#1[#2]#3{\@ifnextchar[%]
  {\phx@FW@pti@ns#1[{#2}]#3}{\phx@FW@pti@ns#1[{#2}]#3[]}}
\@onlypreamble\phx@FW@ptions
\def\phx@FW@pti@ns#1[#2]#3[#4]{%
  \def\reserved@b##1,{%
  \ifx\@nnil##1\relax\else
    \ifx\@nnil##1\@nnil\else
      \noexpand\@onefilewithoptions{phx-##1}[{\unexpanded{#2}}][{#4}]%
      \noexpand\@pkgextension
    \fi
    \expandafter\reserved@b
  \fi}%
  \edef\reserved@a{\zap@space#3~\@empty}%
  \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
  \reserved@a}
\@onlypreamble\phx@FW@pti@ns
```

(*End of definition for* `\usephxmodu` *and* `\RequirePhxmodu`. *These functions are documented on page* 2.)

B.1.3 The (used to be) common module

The code below used to be the automatically-loaded common module, but now we load it together with physics3's code. This change may bring better performance in Windows system.

Check if unicode-math loaded and (re)define the vert symbols. The \relax's at the ends of \vert and \Vert's definitions must not be removed. They are for \ifx to compare. unicode-math sets these symbols \fam1, \symoperators is equal to 1 in LaTeX 2_ε kernel. Moreover, we make \mid as a delimiter but it may not work.

```
\AtBeginDocument{\ifcsname symrm\endcsname
  \protected\def\|{\Udelimiter 0 \symoperators "2016 }%
  \protected\def\vert{\Udelimiter 0 \symoperators "007C\relax}%
  \protected\def\Vert{\Udelimiter 0 \symoperators "2016\relax}%
  \protected\def\mid{\Udelimiter 3 \symoperators "007C }%
\fi}
\protected\def\Vert{\delimiter"026B30D\relax}
\protected\def\mid{\delimiter"326A30C }
```

\delopen \delopen ⟨*left delimiter*⟩

\delclose \delclose ⟨*right delimiter*⟩

Actually in TeX, \left and \right will enclose the subformula as "inner", but \delopen and \delclose will make the subformula an empty open node and a non-empty close node.

```
\DeclareRobustCommand\delopen{\mathopen{}\mathclose\bgroup\left}
\DeclareRobustCommand\delclose{\aftergroup\egroup\right}
% Extension to 2e kernel's or amsmath's biggggg commands.
```

(*End of definition for* \delopen *and* \delclose*. These functions are documented on page* 3.)

\bBigg@ is a command from amsmath. The code below should update with amsmath together.

```
\ifdefined\bBigg@
  \DeclareRobustCommand\biggg{\bBigg@{3}}
  \DeclareRobustCommand\Biggg{\bBigg@{3.5}}
\else
  \DeclareRobustCommand\biggg[1]{\leavevmode@ifvmode
    {\hbox{$\left#1\vbox to20.5\p@{}\right.\n@space$}}}
  \DeclareRobustCommand\Biggg[1]{\leavevmode@ifvmode
    {\hbox{$\left#1\vbox to23.5\p@{}\right.\n@space$}}}
  \AtBeginDocument{\ifdefined\bBigg@
    \DeclareRobustCommand\biggg{\bBigg@{3}}%
    \DeclareRobustCommand\Biggg{\bBigg@{3.5}}%
```

```
  \fi}
\fi
\DeclareRobustCommand\bigggl{\mathopen\biggg}
\DeclareRobustCommand\bigggm{\mathrel\biggg}
\DeclareRobustCommand\bigggr{\mathclose\biggg}
\DeclareRobustCommand\Bigggl{\mathopen\Biggg}
\DeclareRobustCommand\Bigggm{\mathrel\Biggg}
\DeclareRobustCommand\Bigggr{\mathclose\Biggg}
```

__phx_mathvphantom:n

\phx@mathvphantom {⟨*math mode material*⟩}

This command is just like \vphantom in LaTeX 2_ε kernel but only works in math mode.

```
\cs_new:Npn \__phx_mathvphantom:n #1
  {
    \hbox_set:Nn \l__phx_tmpa_box {}
    \tex_mathchoice:D
      {
        \hbox_set:Nn    \l_tmpa_box    { $\displaystyle#1$ }
        \box_set_ht:Nn  \l__phx_tmpa_box { \box_ht:N \l_tmpa_box }
        \box_set_dp:Nn  \l__phx_tmpa_box { \box_dp:N \l_tmpa_box }
        \box_use_drop:N \l__phx_tmpa_box
      }
      {
        \hbox_set:Nn    \l_tmpa_box    { $\textstyle#1$ }
        \box_set_ht:Nn  \l__phx_tmpa_box { \box_ht:N \l_tmpa_box }
        \box_set_dp:Nn  \l__phx_tmpa_box { \box_dp:N \l_tmpa_box }
        \box_use_drop:N \l__phx_tmpa_box
      }
      {
        \hbox_set:Nn    \l_tmpa_box    { $\scriptstyle#1$ }
        \box_set_ht:Nn  \l__phx_tmpa_box { \box_ht:N \l_tmpa_box }
        \box_set_dp:Nn  \l__phx_tmpa_box { \box_dp:N \l_tmpa_box }
        \box_use_drop:N \l__phx_tmpa_box
      }
      {
        \hbox_set:Nn    \l_tmpa_box    { $\scriptscriptstyle#1$ }
        \box_set_ht:Nn  \l__phx_tmpa_box { \box_ht:N \l_tmpa_box }
        \box_set_dp:Nn  \l__phx_tmpa_box { \box_dp:N \l_tmpa_box }
        \box_use_drop:N \l__phx_tmpa_box
      }
  }
\cs_set_eq:NN \phx@mathvphantom \__phx_mathvphantom:n
```

(*End of definition for* `\__phx_mathvphantom:n`.)

B.1.4 The (used to be) `explsetup` module

`\l__phx_tmpa_tl` `\l__phx_tmpb_tl`

Some common variables and functions for experimental LaTeX3 syntax.

```
\tl_new:N \l__phx_tmpa_tl
\tl_new:N \l__phx_tmpb_tl
```

(*End of definition for* `\l__phx_tmpa_tl` *and* `\l__phx_tmpb_tl`.)

`\phx@del*` `\phx@del\big` `\phx@del\Big` `\phx@del\bigg` `\phx@del\Bigg` `\phx@del\biggg` `\phx@del\Biggg`

The syntax seems not important. These following lines seems only for `\ifcsname` to judge if the commands are defined.

```
\def\phx@del#1#2#3{\phx@abopen#1#3\phx@abclose#2}
\expandafter\def\csname phx@del\string*\endcsname
  #1#2#3{\mathopen#1#3\mathclose#2}
\expandafter\def\csname phx@del\string\big\endcsname
  #1#2#3{\bigl#1#3\bigr#2}
\expandafter\def\csname phx@del\string\Big\endcsname
  #1#2#3{\Bigl#1#3\Bigr#2}
\expandafter\def\csname phx@del\string\bigg\endcsname
  #1#2#3{\biggl#1#3\biggr#2}
\expandafter\def\csname phx@del\string\Bigg\endcsname
  #1#2#3{\Biggl#1#3\Biggr#2}
\expandafter\def\csname phx@del\string\biggg\endcsname
  #1#2#3{\bigggl#1#3\bigggr#2}
\expandafter\def\csname phx@del\string\Biggg\endcsname
  #1#2#3{\Bigggl#1#3\Bigggr#2}
```

(*End of definition for* `\phx@del*` *and others.*)

Parsing the clist in optional argument.

```
\def\@phx@ev@do@pt#1,{\ifx#1\relax\@empty\else
  \edef\reserved@a{\zap@space#1~\@empty}%
  \ifx\reserved@a\@empty\else
    \ifcsname phx@del\expandafter \string \expandafter
      \csname \expandafter \detokenize{\reserved@a}\endcsname \endcsname
      \xdef\@phx@ev{\reserved@a}%
    \else
      \xdef\@phx@ev@basis{\reserved@a}%
    \fi
  \fi
  \expandafter\@phx@ev@do@pt\fi}
\def\phx@ev@doopt#1{%
  \edef\@phx@ev{\@empty}\edef\@phx@ev@basis{\@empty}\@phx@ev@do@pt#1,\relax,}
```

```
\file_input_stop:
```
⟨/package⟩

Temporally disable the namespace.

⟨@@=⟩

B.2 The ab module

⟨∗gibberish⟩

This module is important but the code is hard to read. One of the motivations I manage physics3 with DocStrip is that, when I tried to write a new module based on ab after 5 months when I maintained physics3 the last time, I found that I could not understand the code I wrote at all! Therefore, it's significant to comment out the alien code in ab.

⟨/gibberish⟩

⟨∗ab⟩

```
\ProvidesFile {phx-ab.sty} [\phx@date\ ‘ab’ (autobraces) module of physics3]
```

If you don't know when to use \phx@define@key, \phx@setkeys and \phx@processkeyopt in a module, see ahead. In ab, the |tightbraces| option can control if the automatically-sized braces are tight or not. Do you remember \delopen and \delclose?

```
\phx@define@key{ab}{tightbraces}{true}{\def\@phx@abtight{#1}}
\phx@setkeys{ab}{tightbraces=true}
\phx@processkeyopt{ab}
```

\phx@abopen
\phx@abclose

\phx@abopen ⟨*left delimiter*⟩
\phx@abclose ⟨*right delimiter*⟩

They are defined either {\delopen, \delclose} or {\left, \right}. If a module requires ab, these two commands are likely to be used.

```
\ifx\@phx@abtight\phx@true
  \let\phx@abopen\delopen
  \let\phx@abclose\delclose
\else
  \let\phx@abopen\left
  \let\phx@abclose\right
\fi
```

(*End of definition for* \phx@abopen *and* \phx@abclose.)

B.2.1 The implementation of \ab

This is the alienest part of ab. It's better to draw something rather than write boring comments. First let's take a look at \ab's syntax. After \ab should be a pair of delimiters; take () as an example. Between \ab and "(" can be a biggg command or star, or even nothing. \ab is defined as follows:

\ab ← **begindef**
  \phx@d@lx {mb} {ab}
enddef

where ab is the branch name of \ab(), and mb is the branch name of \ab\big() and \ab*(). Then let's see the syntax of \phx@d@lx.

\phx@d@lx {⟨*biggg or star branch name*⟩}
{⟨*automatic branch name*⟩} {#3}

Here exists an #3. #3 is one token immediately following \ab, which can be {a biggg command or a star} or a "(", under our assumption.

\phx@d@lx is defined as follows:

\phx@d@lx ← **begindef**(#1: biggg or star branch name, ⟨*mb*⟩; #2: automatic branch name, ⟨*ab*⟩; #3, the token after \ab)
  if#3 == biggg or #3 == star (⇔ csname {phx@del\string#3} is defined)**then**
    let ⟨*next cs*⟩ = csname {phx@d@lx⟨*mb*⟩}
  else
    let ⟨*next cs*⟩ = csname {phx@d@lx⟨*ab*⟩}
  endif
  ⟨*next cs*⟩ #3
enddef

The condition should be true when #3 is \big or *, and it should be false when #3 is "(". Accordingly, in math mode,

\ab \big (→ \phx@d@lxmb \big (
\ab (→ \phx@d@lxab (

Then we meet two new commands — \phx@d@lxmb and \phx@d@lxab. Syntax is as follows.

```
\phx@d@lxmb ⟨biggg or *⟩ ⟨left delimiter⟩
   ⟨subformula⟩ ⟨right delimiter⟩
\phx@d@lxab             ⟨left delimiter⟩
   ⟨subformula⟩ ⟨right delimiter⟩
```

Notice that ab and mb in the above commands are names of \ab's two branches — they are like namespaces. \phx@d@lxmb and \phx@d@lxab are defined by the following two lines:

```
\phx@d@l@genxm{mb} \phx@d@l@genxa{ab}
```

\phx@d@l@genxm and \phx@d@l@genxa are defined as follows:

```
\phx@d@l@genxm ← begindef(#1: biggg or star branch name, ⟨mb⟩)
                  \phx@d@lx⟨mb⟩ ← begindef(##1: biggg or star;
                                           ##2: left delimiter)
                                     \begingroup
                                     if##1 == starthen
                                       ⟨temp⟩ ← \relax
                                     else
                                       ⟨temp⟩ ← ##1
                                     endif
                                     csname{phx@⟨mb⟩@\string##2}
                                       ⟨temp⟩ ##2
                                     % requires an \endgroup af-
                                       ter the right delimiter
                                   enddef
                enddef
\phx@d@l@genxa ← begindef(#1: automatic branch name, ⟨ab⟩)
                  \phx@d@lx⟨ab⟩ ← begindef(##1: left delimiter)
                                     csname{phx@⟨ab⟩@\string##1}
                                       ##1
                                   enddef
                enddef
```

So we can get

```
\ab \big (   →   \begingroup csname {phx@mb@(} \big   (
\ab *     (   →   \begingroup csname {phx@mb@(} \relax (
\ab       (   →                 csname {phx@ab@(} (
```

The csnames above (`\phx@mb@(` and `\phx@ab@()` are generated with `\phx@AB@gen`.

`\phx@AB@gen` {⟨*branch name*⟩} ⟨*left delimiter*⟩
{⟨*arg spec*⟩} {⟨*definition*⟩}

If ⟨*branch name*⟩ is `mb`, {⟨*arg spec*⟩} should be `mr()`, where `m` is for biggg or star; If ⟨*branch name*⟩ is `ab`, {⟨*arg spec*⟩} should be `r()`.

TEXhackers note: The "`(`" in the example above must not be replaced by a subformula braced by a pair of `{}`.

`\phx@AB@gen`

`\phx@AB@gen` {⟨*branch name*⟩} ⟨*left delimiter*⟩
{⟨*arg spec*⟩} {⟨*definition*⟩}

```
\def\phx@AB@gen#1#2{\expandafter
  \DeclareDocumentCommand\csname phx@#1@\string#2\endcsname}
    \phx@AB@gen{ab}({r()}%
      {\phx@abopen(#1\phx@abclose)}
    \phx@AB@gen{ab}[{r[]}%
      {\phx@abopen[#1\phx@abclose]}
    \phx@AB@gen{ab}\{{r\{\}}%
      {\phx@abopen\{#1\phx@abclose\}}
    \phx@AB@gen{ab}|{r||}%
      {\phx@abopen|#1\phx@abclose|}
    \phx@AB@gen{ab}\|{r\|\|}%
      {\phx@abopen\|#1\phx@abclose\|}
    \phx@AB@gen{ab}<{r<>}%
      {\phx@abopen<#1\phx@abclose>}
    \phx@AB@gen{ab}\lbrace{r\lbrace\rbrace}%
      {\phx@abopen\lbrace#1\phx@abclose\rbrace}
    \phx@AB@gen{ab}\vert{r\vert\vert}%
      {\phx@abopen\vert#1\phx@abclose\vert}
    \phx@AB@gen{ab}\Vert{r\Vert\Vert}%
      {\phx@abopen\Vert#1\phx@abclose\Vert}
    \phx@AB@gen{ab}\langle{r\langle\rangle}%
      {\phx@abopen\langle#1\phx@abclose\rangle}
```

\endgroup's in the end of the following definitions are corresponding to \begingroup's in the definition of \phx@d@l@genxm.

```
\phx@AB@gen{mb}({mr()}%
  {\mathopen#1(#2\mathclose#1)\endgroup}
\phx@AB@gen{mb}[{mr[]}%
  {\mathopen#1[#2\mathclose#1]\endgroup}
\phx@AB@gen{mb}\{{mr\{\}}%
  {\mathopen#1\lbrace#2\mathclose#1\rbrace\endgroup}
\phx@AB@gen{mb}|{mr||}%
  {\mathopen#1\vert#2\mathclose#1\vert\endgroup}
\phx@AB@gen{mb}\|{mr\|\|}%
  {\mathopen#1\Vert#2\mathclose#1\Vert\endgroup}
\phx@AB@gen{mb}<{mr<>}%
  {\mathopen#1\langle#2\mathclose#1\rangle\endgroup}
\phx@AB@gen{mb}\lbrace{mr\lbrace\rbrace}%
  {\mathopen#1\lbrace#2\mathclose#1\rbrace\endgroup}
\phx@AB@gen{mb}\vert{mr\vert\vert}%
  {\mathopen#1\vert#2\mathclose#1\vert\endgroup}
\phx@AB@gen{mb}\Vert{mr\Vert\Vert}%
  {\mathopen#1\Vert#2\mathclose#1\Vert\endgroup}
\phx@AB@gen{mb}\langle{mr\langle\rangle}%
  {\mathopen#1\langle#2\mathclose#1\rangle\endgroup}
```

(*End of definition for* \phx@AB@gen.)

\phx@d@lx

\phx@d@lx {⟨*biggg or star branch name*⟩} {⟨*automatic branch name*⟩} {#3}

```
\def\phx@d@lx#1#2#3{%
  \ifcsname phx@del\string#3\endcsname
    \def\reserved@a{#1}% #3 is star or \biggg
  \else
    \def\reserved@a{#2}% #3 is delimiter
  \fi
  \csname phx@d@lx\reserved@a\endcsname#3}
```

(*End of definition for* \phx@d@lx.)

\phx@d@l@genxm

\phx@d@l@genxm {⟨*biggg or star branch name*⟩}

\phx@d@l@genxa

\phx@d@l@genxa {⟨*automatic branch name*⟩}

```
\def\phx@d@l@genxm#1{%
  \expandafter\def\csname phx@d@lx#1\endcsname##1##2{%
    \begingroup % \endgroup is at the end of #4 of \phx@AB@gen
```

```
    \ifx##1*\let\phx@tempa=\relax\else\let\phx@tempa=##1\fi
    \csname phx@#1@\string##2\endcsname\phx@tempa##2}}
\def\phx@d@l@genxa#1{%
  \expandafter\def\csname phx@d@lx#1\endcsname##1{%
    \csname phx@#1@\string##1\endcsname##1}}
```

(*End of definition for* `\phx@d@l@genxm` *and* `\phx@d@l@genxa`.)

`\phx@d@lxmb` `\phx@d@lxab`

`\phx@d@lxmb` ⟨*biggg or **⟩ ⟨*left delimiter*⟩ ⟨*subformula*⟩ ⟨*right delimiter*⟩
`\phx@d@lxab` ⟨*left delimiter*⟩ ⟨*subformula*⟩ ⟨*right delimiter*⟩

```
\phx@d@l@genxm{mb}
\phx@d@l@genxa{ab}
```

(*End of definition for* `\phx@d@lxmb` *and* `\phx@d@lxab`.)

`\ab` The users' command `\ab`.

```
\DeclareRobustCommand\ab{\phx@d@lx{mb}{ab}}
```

(*End of definition for* `\ab`. *This function is documented on page* 3.)

B.2.2 `\pab`-like commands

This is so simple. No need to comment a lot.

`\phx@d@l@geny` `\phx@d@l@geny` ⟨*command*⟩ ⟨*left delimiter*⟩ ⟨*right delimiter*⟩

This command used to define commands like `\pab`.

```
\def\phx@d@l@geny#1#2#3{%
```

`#1` : star; `#2` : bigg (csname); `#3` : subformula.

```
  \DeclareDocumentCommand#1{som}{
    \IfBooleanTF{##1}%
      {#2##3#3}%
      {\IfValueTF{##2}%
        {\csname##2l\endcsname#2##3\csname##2r\endcsname#3}%
        {\phx@abopen#2##3\phx@abclose#3}%
      }%
  }%
}
\phx@d@l@geny\pab()
```

```
\phx@d@l@geny\bab[]
\phx@d@l@geny\Bab\lbrace\rbrace
\phx@d@l@geny\vab\vert\vert
\phx@d@l@geny\aab\langle\rangle
\phx@d@l@geny\Vab\Vert\Vert
\protect \endinput
```

⟨/ab⟩

(*End of definition for* \phx@d@l@geny.)

B.3 The ab.braket module

⟨*ab.braket⟩

```
\ProvidesFile {phx-ab.braket.sty} [\phx@date\ ‘ab.braket’ module of physics3]
```

This module requires \phx@abopen and \phx@abclose from ab. This module may have conflict with braket.

```
\RequirePhxmodu{ab}
\ifdefined\phx@bra@@
  \PackageWarning{physics3}{You cannot load ‘ab.braket’ and ‘braket’
    modules together.\MessageBreak Only ‘ab.braket’ module works now.}
\fi
```

\bra \bra < ⟨*subformula*⟩ |

(*End of definition for* \bra. *This function is documented on page 5.*)

```
\phx@AB@gen{br.m}<{mr<|}{\mathopen#1\langle#2\mathclose#1\vert\endgroup}
\phx@AB@gen{br.a}<{r<|}{\phx@abopen\langle#1\phx@abclose\vert}
\phx@d@l@genxm{br.m}
\phx@d@l@genxa{br.a}
\DeclareRobustCommand\bra{\phx@d@lx{br.m}{br.a}}
```

\ket \ket | ⟨*subformula*⟩ >

(*End of definition for* \ket. *This function is documented on page 5.*)

```
\phx@AB@gen{kt.m}|{mr|>}{\mathopen#1\vert#2\mathclose#1\rangle\endgroup}
\phx@AB@gen{kt.a}|{r|>}{\phx@abopen\vert#1\phx@abclose\rangle}
\phx@d@l@genxm{kt.m}
\phx@d@l@genxa{kt.a}
\DeclareRobustCommand\ket{\phx@d@lx{kt.m}{kt.a}}
```

\braket \braket <⟨*subformula*$_1$⟩ | ⟨*subformula*$_2$⟩ [| ⟨*subformula*$_3$⟩ ...] >

```
\begingroup
\catcode`\|=\active
\gdef\phx@@mb@bk#1#2{\begingroup
  \mathcode`\|="8000\def|{\egroup#1\vert\bgroup}%
  \def\<{\mathrel{<}}\def\>{\mathrel{>}}%
  \mathopen#1\langle\bgroup#2\egroup\mathclose#1\rangle\endgroup}
\gdef\phx@@ab@bk#1{\begingroup
  \mathcode`\|="8000\def|{\egroup\phx@abb@bkv\bgroup}%
  \def\<{\mathrel{<}}\def\>{\mathrel{>}}%
  \phx@abopen\langle\bgroup#1\egroup\phx@abclose\rangle\endgroup}
\endgroup
\def\phx@abb@bkv{\middle\vert}
\phx@AB@gen{bk.m}<{mr<>}{\phx@@mb@bk#1{#2}\endgroup}
\phx@AB@gen{bk.a}<{r<>}{\phx@@ab@bk{#1}}
\phx@d@l@genxm{bk.m}
\phx@d@l@genxa{bk.a}
\DeclareRobustCommand \braket {\phx@d@lx{bk.m}{bk.a}}
```

(*End of definition for* `\braket`*. This function is documented on page 6.*)

\ketbra \braket |⟨*subformula*$_1$⟩> ⟨*subformula*$_2$⟩ <⟨*subformula*$_3$⟩|

```
\begingroup
\catcode`\<=\active
\catcode`\>=\active
\gdef\phx@@mb@kb#1#2{\begingroup
  \mathcode`\<="8000 \mathcode`\>="8000%
  \def<{#1\langle}\def>{#1\rangle}%
  \def\<{\phx@abb@l}\def\>{\phx@abb@r}%
  \mathopen#1\vert#2\mathclose#1\vert\endgroup}
\endgroup
\gdef\phx@@ab@kb#1>#2<#3\phx@@end{\begingroup
  \def\<{\phx@abb@l}\def\>{\phx@abb@r}%
  \phx@abopen\vert\mathopen{\phx@mathvphantom{#3}}#1\phx@abclose\rangle#2%
  \phx@abopen\langle#3\mathclose{\phx@mathvphantom{#1}}\phx@abclose\vert
\endgroup}
\AtBeginDocument{\ifcsname symbf\endcsname
  \def\phx@abb@l{\Umathchar 3 \symoperators "003C }%
  \def\phx@abb@r{\Umathchar 3 \symoperators "003E }%
\fi}
\def\phx@abb@l{\mathchar"313C }
\def\phx@abb@r{\mathchar"313E }
```

```
\phx@AB@gen{kb.m}|{mr||}{\phx@@mb@kb#1{#2}\endgroup}
\phx@AB@gen{kb.a}|{r||}{\phx@@ab@kb#1\phx@@end}
\phx@d@l@genxm{kb.m}
\phx@d@l@genxa{kb.a}
\DeclareRobustCommand \ketbra {\phx@d@lx{kb.m}{kb.a}}
```

(*End of definition for* `\ketbra`. *This function is documented on page* 7.)

`\bknorm` `\kbproj` `\expval`

Define some high-level commands for inner product, outer product of basis and expectation value.

```
\DeclareDocumentCommand \bknorm { s o m }
  {%
    \IfBooleanTF{#1} {\mathopen\langle#3\vert #3\mathclose\rangle}
      {%
        \IfValueTF{#2}
          {\expanded{\noexpand\phx@d@lx{bk.m}{bk.a}%
            \expandafter\noexpand\csname #2\endcsname}<#3|#3>}
          {\phx@d@lx{bk.m}{bk.a}<#3|#3>}%
      }%
  }
\DeclareDocumentCommand \kbproj { s o m }
  {%
    \IfBooleanTF{#1}
      {%
        \mathopen\vert#3\mathclose\rangle
        \mathopen\langle#3\mathclose\vert
      }
      {%
        \IfValueTF{#2}
          {\expanded{\noexpand\phx@d@lx{kb.m}{kb.a}%
            \expandafter\noexpand\csname #2\endcsname}|#3><#3|}
          {\phx@d@lx{kb.m}{kb.a}|#3><#3|}%
      }%
  }
\DeclareDocumentCommand \expval { s o m }
  {%
    \phx@ev@doopt{#2}%
    \IfValueTF {#2}
      {%
        \ifx\@phx@ev\@empty
          \ifx \@phx@ev@basis\@empty \else
          \IfBooleanTF{#1}
```

```
            {\phx@d@lx{bk.m}{bk.a}*<\@phx@ev@basis|#3|\@phx@ev@basis>}
            {\phx@d@lx{bk.m}{bk.a}<\@phx@ev@basis |#3|\@phx@ev@basis>}%
          \fi
        \else
          \ifx\@phx@ev@basis\@empty
            \IfBooleanTF{#1} {\phx@d@lx{bk.m}{bk.a}*<#3>}
              {%
                \expanded{%
                  \noexpand \phx@d@lx{bk.m}{bk.a}\expandafter \noexpand
                  \csname \@phx@ev\endcsname}<#3>%
              }%
            \else
              \IfBooleanTF{#1}
              {%
                \phx@d@lx{bk.m}{bk.a}*<\@phx@ev@basis|#3|\@phx@ev@basis>%
              }
              {%
                \expanded{%
                  \noexpand \phx@d@lx{bk.m}{bk.a}\expandafter
                  \noexpand \csname \@phx@ev\endcsname}%
                <\@phx@ev@basis|#3|\@phx@ev@basis>%
              }%
            \fi
          \fi
      }
      {%
        \IfBooleanTF{#1}
          {\phx@d@lx{bk.m}{bk.a}*<#3>} {\phx@d@lx{bk.m}{bk.a}<#3>}%
      }%
  }
```

(*End of definition for* `\bknorm`, `\kbproj`, *and* `\expval`. *These functions are documented on page* 7.)

```
\protect \endinput
```

⟨/ab.braket⟩

B.4 The `braket` module

⟨*braket⟩

```
\ProvidesFile {phx-braket.sty} [\phx@date\ 'braket' module of physics3]
```

This module requires `\phx@abopen` and `\phx@abclose` from `ab`. This module may have conflict with `ab.braket`.

```
\RequirePhxmodu{ab}
\ifdefined\phx@abb@bkv
  \PackageWarning{physics3}{You cannot load ‘ab.braket’ and ‘braket’
    modules together.\MessageBreak Only ‘braket’ module works now.}
\fi
```

\bra \bra * [⟨*biggg*⟩] {⟨*subformula*⟩}

```
\DeclareDocumentCommand\bra{ s o m }{%
  \IfBooleanTF{#1}
    {\mathopen\langle#3\mathclose\vert}
    {\IfValueTF{#2}
      {\csname#2l\endcsname\langle#3\csname#2r\endcsname\vert}
      {\phx@abopen\langle#3\phx@abclose\vert}%
    }%
}
```

(*End of definition for* \bra*. This function is documented on page 5.*)

\ket \ket * [⟨*biggg*⟩] {⟨*subformula*⟩}

```
\DeclareDocumentCommand\ket{ s o m }{%
  \IfBooleanTF{#1}
    {\mathopen\vert#3\mathclose\rangle}
    {\IfValueTF{#2}
      {\csname#2l\endcsname\vert#3\csname#2r\endcsname\rangle}
      {\phx@abopen\vert#3\phx@abclose\rangle}%
    }%
}
```

(*End of definition for* \ket*. This function is documented on page 5.*)

\braket \ket * [⟨*biggg*⟩, $n \in \{1, 2, 3\}$] {⟨*subformula 1*⟩} ...{⟨*subformula n*⟩}

```
\DeclareDocumentCommand \braket { s O{} }
  {%
    \IfBooleanTF{#1}
      {%
        \gdef\@phx@bk@argnum{ii}%
        \phx@bk@doopt{#2}%
        \gdef\@phx@bk@l{mathopen}%
        \gdef\@phx@bk@m{mathord}%
        \gdef\@phx@bk@r{mathclose}%
      }
      {%
```

```
        \gdef\@phx@bk@argnum{ii}%
        \gdef\@phx@bk@l{phx@abopen}%
        \gdef\@phx@bk@m{middle}%
        \gdef\@phx@bk@r{phx@abclose}%
        \phx@bk@doopt{#2}%
      }%
    \csname phx@bk@in@\@phx@bk@argnum\endcsname
  }
```

(*End of definition for* `\braket`. *This function is documented on page 6.*)

`\phx@bk@in@i` `\phx@bk@in@ii` `\phx@bk@in@iii`

`\phx@bk@in@`⟨*n.roman*⟩ `{`⟨*subformula 1*⟩`}` … `{`⟨*subformula n*⟩`}`

⟨*n.roman*⟩ is n in roman lowercase, where $n \in \{1, 2, 3\}$.

```
\def\phx@bk@in@i#1{%
  \csname\@phx@bk@l\endcsname\langle{#1}%
  \csname\@phx@bk@r\endcsname\rangle}
\def\phx@bk@in@ii#1#2{%
  \csname\@phx@bk@l\endcsname\langle{#1}%
  \csname\@phx@bk@m\endcsname\vert{#2}%
  \csname\@phx@bk@r\endcsname\rangle}
\def\phx@bk@in@iii#1#2#3{%
  \csname\@phx@bk@l\endcsname\langle{#1}%
  \csname\@phx@bk@m\endcsname\vert{#2}%
  \csname\@phx@bk@m\endcsname\vert{#3}%
  \csname\@phx@bk@r\endcsname\rangle}
```

(*End of definition for* `\phx@bk@in@i`, `\phx@bk@in@ii`, *and* `\phx@bk@in@iii`.)

`\phx@bk@doopt` `\phx@bk@do@pt`

`\phx@bk@doopt {`⟨*clist*⟩`}`

Parse the optional argument of `\braket`. This will add 3 entries to hash.

```
\def\@phx@bk@do@pt#1,{\ifx#1\relax\@empty\else
  \edef\reserved@a{\zap@space#1 \@empty}%
  \ifx\reserved@a\@empty\else
    \ifcsname phx@del\expandafter\string\csname\reserved@a\endcsname\endcsname
      \xdef\@phx@bk@l{\reserved@a l}%
      \xdef\@phx@bk@m{\reserved@a}% but not m (m stands for \mathrel)
      \xdef\@phx@bk@r{\reserved@a r}%
    \else
      \ifnum\reserved@a>3%
        \PackageError{physics3}{\string\braket\space can only take 3
          mandatory arguments at most}{Check if you had written a number
          more than 3 in the [optional] argument.}%
```

```
      \fi
      \xdef\@phx@bk@argnum{\romannumeral\reserved@a}%
    \fi
  \fi
  \expandafter\@phx@bk@do@pt\fi}
\def\phx@bk@doopt#1{\@phx@bk@do@pt#1,\relax,}
```

(*End of definition for* `\phx@bk@doopt` *and* `\phx@bk@do@pt`.)

`\ketbra` `\ketbra * [⟨biggg⟩] {⟨subformula 1⟩} [⟨between 1 and 2⟩] {⟨subformula 2⟩}`

```
\DeclareDocumentCommand \ketbra { s o m O{} m }
  {%
    \IfBooleanTF{#1}
      {%
        \mathopen\vert#3\mathclose\rangle#4%
        \mathopen\langle#5\mathclose\vert
      }
      {%
        \IfValueTF{#2}
          {\csname#2l\endcsname\vert#3\csname#2r\endcsname\rangle#4%
            \csname#2l\endcsname\langle#5\csname#2r\endcsname\vert}
          {\begingroup
            \phx@abopen\vert
              \mathopen{\phx@mathvphantom{#5}}#3\phx@abclose\rangle#4%
            \phx@abopen\langle#5
              \mathclose{\phx@mathvphantom{#3}}\phx@abclose\vert
          \endgroup}%
      }%
  }
```

(*End of definition for* `\ketbra`. *This function is documented on page* 7.)

`\bknorm` `\kbproj` `\expval` Define some high-level commands for inner product, outer product of basis and expectation value.

```
\DeclareDocumentCommand \bknorm { s o m }
  {%
    \IfBooleanTF{#1}
      {%
        \gdef\@phx@bk@l{mathopen}%
        \gdef\@phx@bk@m{mathord}%
        \gdef\@phx@bk@r{mathclose}%
```

```
      }
      {%
        \gdef\@phx@bk@l{phx@abopen}%
        \gdef\@phx@bk@m{middle}%
        \gdef\@phx@bk@r{phx@abclose}%
        \IfValueT{#2}
          {%
            \ifcsname phx@del\expandafter\string\csname#2\endcsname\endcsname
              \xdef\@phx@bk@l{#2l}%
              \xdef\@phx@bk@m{#2}%
              \xdef\@phx@bk@r{#2r}%
            \fi
          }%
      }%
    \phx@bk@in@ii {#3} {#3}%
  }
\DeclareDocumentCommand \kbproj { s o m }
  {%
    \IfBooleanTF{#1}
      {%
        \mathopen\vert#3\mathclose\rangle
        \mathopen\langle#3\mathclose\vert
      }
      {%
        \IfValueTF{#2}
          {\csname#2l\endcsname\vert#3\csname#2r\endcsname\rangle
            \csname#2l\endcsname\langle#3\csname#2r\endcsname\vert}
          {\begingroup
            \phx@abopen\vert
              \mathopen{\phx@mathvphantom{#3}}#3\phx@abclose\rangle
            \phx@abopen\langle#3
              \mathclose{\phx@mathvphantom{#3}}\phx@abclose\vert
          \endgroup}%
      }%
  }
\DeclareDocumentCommand \expval { s O{} m }
  {
    \phx@ev@doopt{#2}%
    \IfBooleanTF{#1}
      {%
        \gdef\@phx@bk@l{mathopen}%
        \gdef\@phx@bk@m{mathord}%
```

```
        \gdef\@phx@bk@r{mathclose}%
      }
      {%
        \gdef\@phx@bk@l{phx@abopen}%
        \gdef\@phx@bk@m{middle}%
        \gdef\@phx@bk@r{phx@abclose}%
        \ifx \@phx@ev\@empty \else
          \xdef\@phx@bk@l{\@phx@ev l}%
          \xdef\@phx@bk@m{\@phx@ev}%
          \xdef\@phx@bk@r{\@phx@ev r}%
        \fi
      }%
      \ifx \@phx@ev@basis\@empty
        \phx@bk@in@i {#3}%
      \else
        \phx@bk@in@iii {\@phx@ev@basis} {#3} {\@phx@ev@basis}%
      \fi
  }
```

(*End of definition for* `\bknorm`, `\kbproj`, *and* `\expval`. *These functions are documented on page* 7.)

```
\protect \endinput
```

⟨/braket⟩

Restore the namespace.

⟨@@=phx⟩

B.5 The `doubleprod` module

⟨*doubleprod⟩

```
\ProvidesExplFile {phx-doubleprod.sty} {\phx@date} {\phx@version}
  {'doubleprod' (vertically stacked binary operators) module of physics3}
```

Boolean options.

```
\__phx_define_key:nnnn { doubleprod } { crosssymbol } { } { \def\@phx@dbl@c {#1} }
\__phx_define_key:nnnn { doubleprod } { dotsymbol   } { } { \def\@phx@dbl@d {#1} }
\__phx_define_key:nnnn { doubleprod } { crossscale  } { } { \def\@phx@dbl@sc{#1} }
\__phx_define_key:nnnn { doubleprod } { dotscale    } { } { \def\@phx@dbl@sd{#1} }
\__phx_define_key:nnnn { doubleprod } { crossopenup } { } { \def\@phx@dbl@oc{#1} }
\__phx_define_key:nnnn { doubleprod } { dotopenup   } { } { \def\@phx@dbl@od{#1} }
\__phx_setkeys:nn      { doubleprod }
  {
    crosssymbol = \times, crossscale = 0.8, crossopenup = .02,
```

```
    dotsymbol   = \ldotp, dotscale   = 1,   dotopenup   = .2
  }
\__phx_processkeyopt:n { doubleprod }
\def\phx@dbl@gen#1#2#3#4{%
  \DeclareRobustCommand#1{\mathbin{\vcenter{\baselineskip\z@skip%
    \lineskip#4\phx@dblcurrf@size%
    \hbox_set:Nn \l_tmpa_box {\fontsize{#2\phx@dblcurrf@size}\z@$#3$}
    \box_use:N \l_tmpa_box \box_use_drop:N \l_tmpa_box}}}
\def\phx@dblcurrf@size{\dimexpr\f@size pt\relax}
\phx@dbl@gen\doublecross\@phx@dbl@sc\@phx@dbl@c\@phx@dbl@oc
\phx@dbl@gen\doubledot\@phx@dbl@sd\@phx@dbl@d\@phx@dbl@od
\file_input_stop:
```

⟨/doubleprod⟩

B.6 The `diagmat` module

```
⟨*diagmat⟩
\ProvidesExplFile {phx-diagmat.sty} {\phx@date} {\phx@version}
  {‘diagmat’ module of physics3}
```

This module requires mathtools and xpatch.

```
\RequirePackage { mathtools, etoolbox }
```

Do expansion of column specification argument (thanks @egreg on TEX Stack**Exchange**).

```
\patchcmd \MT_matrix_begin:N { \exp_args:Ne \array } { \exp_args:Ne \array } {}
  { \patchcmd \MT_matrix_begin:N { \array } { \exp_args:Ne \array } {} {} }
\__phx_define_key:nnnn { diagmat } { empty } {   }
  { \tl_gset:Nn \l__phx_mat_empty_tl { #1 } }
\__phx_define_key:nnnn { diagmat } { align } { r }
  { \tl_gset:Nn \l__phx_mat_align_tl { #1 } }
```

This module requires some new variables.

```
\clist_new:N \l__phx_mat_diag_clist
\int_new:N   \l__phx_mat_dim_int
\tl_new:N    \l__phx_mat_line_tl
\tl_new:N    \l__phx_diagmat_tl
\tl_new:N    \l__phx_mat_empty_tl
\tl_new:N    \l__phx_mat_align_tl
\tl_const:Nn \l__phx_mat_align_c_tl { c }
\__phx_processkeyopt:n { diagmat }
\keys_define:nn { phy / diagmat }
  {
    empty   .tl_set:N  = \l__phx_mat_empty_tl,
    align   .tl_set:N  = \l__phx_mat_align_tl,
```

```
  }
```

(*End of definition for* `\l__phx_mat_diag_clist` *and others.*)

\diagmat
\pdiagmat
\bdiagmat
\Bdiagmat
\vdiagmat
\Vdiagmat

`\`⟨*delimiter type*⟩`diagmat [`⟨*key-val list*⟩`] {`⟨*diagonal*⟩`}`

```
\clist_map_inline:nn { {}, p, b, B, v, V }
  {
    \exp_args:Nc \DeclareDocumentCommand { #1diagmat } { s O{} m }
      {
        \IfBooleanTF { ##1 }
          { \__phx_diagmat_type:nnn { #1small } { ##2 } { ##3 } }
          { \__phx_diagmat_type:nnn { #1      } { ##2 } { ##3 } }
      }
  }
```

(*End of definition for* `\diagmat` *and others. These functions are documented on page 8.*)

__phx_diagmat_type:nnn

`\__phx_diagmat_type:nnn {`⟨*delimiter type*⟩`} {`⟨*key-val list*⟩`} {`⟨*diagonal*⟩`}`

```
\cs_new:Npn \__phx_diagmat_type:nnn #1#2#3
  {
    \group_begin:
    \clist_set:Nn \l__phx_mat_diag_clist {#3}
    \int_set:Nn \l__phx_mat_dim_int { \clist_count:N \l__phx_mat_diag_clist }
    \int_compare:nNnT { \l__phx_mat_dim_int } > { \value { MaxMatrixCols } }
      { \setcounter { MaxMatrixCols } { \l__phx_mat_dim_int } }
    \keys_set:nn { phy / diagmat } {#2}
    \tl_gclear:N \l__phx_diagmat_tl
    \int_step_inline:nnn { 1 } { \l__phx_mat_dim_int }
      {
        \int_step_inline:nnn { 1 } { \l__phx_mat_dim_int }
          {
            \int_compare:nNnTF { ##1 } = { ####1 }
              {
                \clist_gpop:NN \l__phx_mat_diag_clist \l__phx_tmpa_tl
                \tl_if_empty:NTF \l__phx_tmpa_tl
                  { \tl_gput_right:Nn \l__phx_mat_line_tl { \l__phx_mat_empty_tl } }
```

Maybe it's better to use `\expandafter\scantokens\expandafter{\l__phx_tmpa_tl}` in the next line.

```
                  { \tl_gput_right:No \l__phx_mat_line_tl { \l__phx_tmpa_tl } }
              }
```

```
              { \tl_gput_right:Nn \l__phx_mat_line_tl { \l__phx_mat_empty_tl } }
            \int_compare:nNnTF { ####1 } = { \l__phx_mat_dim_int }
              {
                \tl_gput_right:Nn \l__phx_mat_line_tl { \\ }
              }
              {
                \tl_gput_right:Nn \l__phx_mat_line_tl { & }
              }
          }
        \tl_gput_right:No \l__phx_diagmat_tl { \l__phx_mat_line_tl }
        \tl_gclear:N \l__phx_mat_line_tl
      }
    \bool_lazy_or:nnTF
      { \tl_if_empty_p:N \l__phx_mat_align_tl                          }
      { \tl_if_eq_p:NN   \l__phx_mat_align_tl \l__phx_mat_align_c_tl }
      {
        \begin { #1 matrix }
          \tl_use:N \l__phx_diagmat_tl
        \end { #1 matrix }
      }
      {
        \begin { #1 matrix* } [ \l__phx_mat_align_tl ]
          \tl_use:N \l__phx_diagmat_tl
        \end { #1 matrix* }
      }
    \group_end:
  }
```

(*End of definition for* `\__phx_diagmat_type:nnn`.)

```
\file_input_stop:
```

⟨/diagmat⟩

B.7 The `xmat` module

⟨*xmat⟩

```
\ProvidesExplFile {phx-xmat.sty} {\phx@date} {\phx@version}
  {‘xmat’ module of physics3}
```

This module requires mathtools and xpatch.

```
\RequirePackage { mathtools, etoolbox }
```

Do expansion of column specification argument (thanks @egreg on TEX StackExchange).

```
\patchcmd \MT_matrix_begin:N { \exp_args:Ne \array } { \exp_args:Ne \array } {}
```

```
  { \patchcmd \MT_matrix_begin:N { \array } { \exp_args:Ne \array } {} {} }
\__phx_define_key:nnnn { xmat } { showtop } { }
  { \int_gset:Nn \l__phx_xmat_showtop_int { #1 } }
\__phx_define_key:nnnn { xmat } { showleft } { }
  { \int_gset:Nn \l__phx_xmat_showleft_int { #1 } }
\__phx_define_key:nnnn { diagmat } { align } { r }
  { \tl_gset:Nn \l__phx_xmat_align_tl { #1 } }
```

This module requires some new variables.

```
\bool_new:N  \l__phx_xmat_extra_vdots_bool
\bool_new:N  \l__phx_xmat_extra_cdots_bool
\int_new:N   \l__phx_xmat_showtop_int
\int_new:N   \l__phx_xmat_showleft_int
\tl_new:N    \l__phx_xmat_tl
\tl_new:N    \l__phx_xmat_align_tl
\tl_const:Nn \l__phx_xmat_align_c_tl { c }
\int_gset:Nn \l__phx_xmat_showtop_int  { \value { MaxMatrixCols } - 2 }
\int_gset:Nn \l__phx_xmat_showleft_int { \value { MaxMatrixCols } - 2 }
```

(*End of definition for* `\l__phx_xmat_extra_vdots_bool` *and others.*)

```
\cs_new:Npn \__phx_xmat_entry_format:nnn #1#2#3
  {
    #1 \c_math_subscript_token { #2 #3 }
  }
\__phx_processkeyopt:n { xmat }
\clist_map_inline:nn { {}, p, b, B, v, V }
  {
    \exp_args:Nc \DeclareDocumentCommand { #1xmat } { s O{} m m m }
      {
        \IfBooleanTF { ##1 }
          { \__phx_xmat_type:nnnnn { #1small } { ##2 } { ##3 } { ##4 } { ##5 } }
          { \__phx_xmat_type:nnnnn { #1      } { ##2 } { ##3 } { ##4 } { ##5 } }
      }
  }
\keys_define:nn { phx / xmat }
  {
    format  .cs_set:Np = \__phx_xmat_entry_format:nnn #1#2#3 ,
    showtop .int_set:N = \l__phx_xmat_showtop_int ,
    showleft.int_set:N = \l__phx_xmat_showleft_int ,
    align   .tl_set:N  = \l__phx_xmat_align_tl ,
  }
```

__phx_if_digits_only_p:n
__phx_if_digits_only:nTF

__phx_if_digits_only:nTF {⟨*token list*⟩} {⟨*true code*⟩} {⟨*flase code*⟩}

Use LaTeX3 regular expression to tell if ⟨`token list`⟩ (the numbers of rows or columns) contain digits only.

(*End of definition for* `\__phx_if_digits_only:nTF`.)

```
\prg_new_conditional:Npnn \__phx_if_digits_only:n #1 { TF }
  {
    \regex_match:nnTF { \A [[:digit:]]* \Z } { #1 }
      { \prg_return_true: } { \prg_return_false: }
  }
```

__phx_xmat_type:nnnnn

__phx_xmat_type:nnnnn {⟨*delimiter type*⟩} {⟨*key-val list*⟩} {⟨*common entry*⟩} {⟨*row number*⟩} {⟨*column number*⟩}

(*End of definition for* `\__phx_xmat_type:nnnnn`.)

```
\cs_new:Npn \__phx_xmat_type:nnnnn #1#2#3#4#5
  {
    \group_begin:
    \tl_gclear:N \l__phx_xmat_tl
    \keys_set:nn { phx / xmat } { #2 } %
    \__phx_if_digits_only:nTF { #4 }
      {
        \int_compare:nNnTF { #4 } < { \l__phx_xmat_showtop_int + 1 }
          {
            \int_set:Nn \l__phx_xmat_showtop_int { #4 }
            \bool_set_false:N \l__phx_xmat_extra_vdots_bool
          }
          {
            \bool_set_true:N \l__phx_xmat_extra_vdots_bool
          }
      }
      {
        \bool_set_true:N \l__phx_xmat_extra_vdots_bool
      }
    \__phx_if_digits_only:nTF { #5 }
      {
        \int_compare:nNnTF { #5 } < { \l__phx_xmat_showleft_int + 1 }
          {
            \int_set:Nn \l__phx_xmat_showleft_int { #5 }
            \bool_set_false:N \l__phx_xmat_extra_cdots_bool
          }
          {
```

```
            \bool_set_true:N \l__phx_xmat_extra_cdots_bool
          }
      }
      {
        \bool_set_true:N \l__phx_xmat_extra_cdots_bool
      }
    \int_step_inline:nn { \l__phx_xmat_showtop_int }
      {
        \tl_put_right:Nn \l__phx_xmat_tl
          { \__phx_xmat_entry_format:nnn { #3 } { ##1 } { 1 } }
        \int_step_inline:nnn { 2 } { \l__phx_xmat_showleft_int }
          {
            \tl_put_right:Nn \l__phx_xmat_tl
              { & \__phx_xmat_entry_format:nnn { #3 } { ##1 } { ####1 } }
          }
        \bool_if:NT \l__phx_xmat_extra_cdots_bool
          {
            \tl_put_right:Nn \l__phx_xmat_tl
              { & \cdots & \__phx_xmat_entry_format:nnn { #3 } { ##1 } { #5 } }
          }
        \tl_put_right:Nn \l__phx_xmat_tl { \\ }
      }
    \bool_if:NT \l__phx_xmat_extra_vdots_bool
      {
        \tl_put_right:Nn \l__phx_xmat_tl { \vdots }
        \prg_replicate:nn { \l__phx_xmat_showleft_int - 1 }
          {
            \tl_put_right:Nn \l__phx_xmat_tl { & \vdots }
          }
        % Add \ddots if vdots_bool and cdots_bool be true simultaneously.
        \bool_if:NT \l__phx_xmat_extra_cdots_bool
          {
            \tl_put_right:Nn \l__phx_xmat_tl { & \ddots & \vdots }
          } % else relax
        \tl_put_right:Nn \l__phx_xmat_tl { \\ }
        % The last row.
        \tl_put_right:Nn \l__phx_xmat_tl
          { \__phx_xmat_entry_format:nnn { #3 } { #4 } { 1 } }
        \int_step_inline:nnn { 2 } { \l__phx_xmat_showleft_int }
          {
            \tl_put_right:Nn \l__phx_xmat_tl
              { & \__phx_xmat_entry_format:nnn { #3 } { #4 } { ##1 } }
```

```
      }
    \bool_if:NT \l__phx_xmat_extra_cdots_bool
      {
        \tl_put_right:Nn \l__phx_xmat_tl
          { & \cdots & \__phx_xmat_entry_format:nnn { #3 } { #4 } { #5 } }
      }
  } % else relax
  \bool_lazy_or:nnTF
    { \tl_if_empty_p:N \l__phx_xmat_align_tl                        }
    { \tl_if_eq_p:NN   \l__phx_xmat_align_tl \l__phx_xmat_align_c_tl }
    {
      \begin { #1 matrix }
        \tl_use:N \l__phx_xmat_tl
      \end { #1 matrix }
    }
    {
      \begin { #1 matrix* } [ \l__phx_xmat_align_tl ]
        \tl_use:N \l__phx_xmat_tl
      \end { #1 matrix* }
    }
  \group_end:
}
\file_input_stop:
```

⟨/xmat⟩

B.8 The operator module

⟨*operator⟩

```
\ProvidesExplFile {phx-operator.sty} {\phx@date} {\phx@version}
  {‘operator’ module of physics3}
\__phx_define_key:nnnn { operator } { ReIm     } { true  }
  { \def\phx@reserveda {#1} }
\__phx_define_key:nnnn { operator } { identity } { } { \def\phx@oplega@id {#1} }
\__phx_define_key:nnnn { operator } { PV       } { } { \def\@phx@oplega@PV{#1} }
\__phx_define_key:nnnn { operator } { pv       } { } { \def\@phx@oplega@pv{#1} }
\__phx_setkeys:nn      { operator }
  { PV = \mathcal{P}, pv = {p.v.}, ReIm = true, identity = bbold }
\__phx_processkeyopt:n { operator }
```

\asin
\acos
\atan
\acsc
\asec
\acot
\erf
\erfc
\Tr
\tr
\rank

```
\clist_map_inline:nn
  { asin, acos, atan, acsc, asec, acot, erf, erfc, Tr, tr, rank, Res, res }
```

```
  {
    \exp_args:Nc \DeclareRobustCommand {#1}
      { \mathop { \operator@font #1 } \nolimits }
  }
\DeclareRobustCommand\PV  {\mathord{\@phx@oplega@PV}}
\DeclareRobustCommand\pv  {\mathop{\operator@font\@phx@oplega@pv{}}\nolimits}
\DeclareRobustCommand\upe {\mathrm e}
\DeclareRobustCommand\iu  {\mathrm i\mkern1mu}
\clist_map_inline:nn { i, I }
  {
    \exp_args:Nc \DeclareDocumentCommand {#1dentity} { o }
      {
        \group_begin:
        \mode_if_math:TF
          {
            \mbox
              {
                \fontencoding{U} \fontseries{m} \fontshape{n}
                \IfValueTF { ##1 }
                  { \fontfamily { ##1 } } { \fontfamily { \phx@oplega@id } }
                \selectfont
                \tl_if_eq:nnT {#1} { i } { 1 }
                \tl_if_eq:nnT {#1} { I } { I }
              }
          }
          {
            \exp_args:Nnx \msg_error:nn { physics3 }
              {
                The ~ \expandafter\string\csname #1dentity\endcsname\space
                command ~ requires ~ math ~ mode.
              }
          }
        \group_end:
      }
  }
```

(*End of definition for* `\asin` *and others.*)

`\Re` `\Im` Restore `\Re` and `\Im` in `\Resymbol` and `\Imsymbol`. The `\AtBeginDocument` hook is used for the compatibility of unicode-math.

```
\ifx \phx@reserveda \phx@true
  \AtBeginDocument
```

```
    {
      \let\Resymbol\Re
      \let\Imsymbol\Im
      \DeclareRobustCommand\Re{\mathop{\operator@font Re}\nolimits}%
      \DeclareRobustCommand\Im{\mathop{\operator@font Im}\nolimits}%
    }
\fi
```

(*End of definition for* `\Re` *and* `\Im`.)

Requires fixdif version 2.x.

```
\RequirePackage{fixdif}[2023/01/31]
\letdif\phx@nl@nabla{nabla}
```

`\grad` `\div` `\curl` `\laplacian` ∇-related operators.

```
\RequirePhxmodu{ab}
\AtBeginDocument
  {
    \ifcsname div\endcsname\let\divsymbol\div\fi
    \DeclareRobustCommand \grad     {\phx@nl@nabla\ab}
    \DeclareRobustCommand \div      {\phx@nl@nabla\cdot\ab}
    \DeclareRobustCommand \curl     {\phx@nl@nabla\times\ab}
    \DeclareRobustCommand \laplacian{\phx@nl@nabla^2\ab}
  }
```

(*End of definition for* `\grad` *and others.*)

```
\file_input_stop:
```

⟨/operator⟩

B.9 The `ab.legacy` module

⟨∗ab.legacy⟩

```
\ProvidesFile {phx-ab.legacy.sty} [\phx@date\ 'ab.legacy' module of physics3]
```

Requires `ab`'s `tight` option.

```
\RequirePhxmodu{ab}
\phx@define@key{ab.legacy}{order}{\mathcal O}{\def\phx@ab@ordersym{#1}}
\phx@setkeys{ab.legacy}{order}
\phx@processkeyopt{ab.legacy}
\phx@d@l@geny\abs\vert\vert
\phx@d@l@geny\norm\Vert\Vert
\DeclareDocumentCommand\order{som}{%
  \phx@ab@ordersym
```

```
  \IfBooleanTF{#1} {(#3)}
    {\IfValueTF{#2}
      {\csname#2l\endcsname(#3\csname#2r\endcsname)}
      {\phx@abopen(#3\phx@abclose)}%
    }%
}
\phx@d@l@geny\eval.\vert
\phx@d@l@geny\peval(\vert
\phx@d@l@geny\beval[\vert
\protect \endinput
```

⟨/ab.legacy⟩

B.10 The `qtext.legacy` module

This module is written for the compatibility with the bad commands provided by physics only. The commands in this module should NEVER be used!

```
⟨*qtext.legacy⟩
\ProvidesExplFile {phx-qtext.legacy.sty} {\phx@date} {\phx@version}
  {‘qtext.legacy’ module of physics3.sty}
\RequirePackage{amstext}
\def\phx@qtext@#1#2{#1\text{#2}\quad}
\DeclareRobustCommand\qqtext{\@ifstar{\phx@qtext@{}}{\phx@qtext@\quad}}
\DeclareRobustCommand\qq    {\qqtext}
\DeclareRobustCommand\qcomma{,\quad}
\DeclareRobustCommand\qc    {\qcomma}
\DeclareRobustCommand\qcc
  {\@ifstar{\phx@qtext@{}{c.c}}{\phx@qtext@\quad{c.c}}}
\clist_map_inline:nn
  {
    if,       then,   else,   otherwise,  unless,
    give,     using,  unless, assume,     since,
    let,      for,    all,    even,       odd,
    integer,  and,    or,     as,         in
  }
  {
    \exp_args:Nc \DeclareRobustCommand { q#1 }
      { \@ifstar{ \phx@qtext@ {} {#1} } { \phx@qtext@ \quad {#1} } }
  }
\file_input_stop:
```

⟨/qtext.legacy⟩

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

C

D

E

F

G

H

I

K

Q

R

S

T

TEX and LATEX 2ε commands:

U

V

X

Z