

The `physics2` package

Zhang Tingxuan

2023/04/02 Version 0.2.2*

Abstract

This is the document for `physics2` package, which defines commands for typesetting math formulae faster and more simply. `physics2` is a modularized package, each module provides its own function.

This document describes the `physics2` package in more detail. But if you are a user of the legacy `physics` package, you can click [here](#) to see the documentation for `physics` users before you start. If you never used `physics` package before, just read *this* documentation.

Contents

1	Introduction	2
1.1	The purpose of this package	2
1.2	Packages required	2
1.3	Loading the <code>physics2</code> package	2
1.4	Loading a module of <code>physics2</code>	2
2	Modules of <code>physics2</code>	3
2.1	The automatically loaded <code>common</code> module	3
2.2	The <code>ab</code> module — automatic braces	4
2.3	The <code>ab.braket</code> module — Dirac bra-ket notation	5
2.4	The <code>braket</code> module — Dirac bra-ket notation	7
2.5	The <code>diagmat</code> module — simple diagonal matrices	8
2.6	The <code>doubleprod</code> module — tensors' double product operator	9
2.7	The <code>xmat</code> module — matrices with formatted entries	10

*<https://www.github.com/AlphaZTX/physics2>

3	The “legacy” modules	11
3.1	The <code>ab.legacy</code> module	12
3.2	The <code>bm-um.legacy</code> module	12
3.3	The <code>nabla.legacy</code> module	13
3.4	The <code>op.legacy</code> module	13
3.5	The <code>qtext.legacy</code> module	14

1 Introduction

1.1 The purpose of this package

This package aims to provide a bundle of commands for typesetting math faster in different modules. The commands provided by `physics2` and its different modules are designed to be short and easy to memorize.

1.2 Packages required

The `physics2` 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 `physics2` might require different packages. It will be explained in the following sections that which module requires which package.

The `physics2` package requires \LaTeX 2_ε kernel released after 2020/10. Please make sure that your \LaTeX distribution is not too old.

1.3 Loading the `physics2` package

Just like loading any package, write

```
\usepackage{physics2}
```

in the preamble to load the `physics2` package. In this version, `physics2` doesn’t provide a package option.

However, `physics2` itself only provides very few functions. Actually, it just provides a method to load modules. You need to load different modules of `physics2` to have different kinds of functions applied to your document.

1.4 Loading a module of `physics2`

You can load a module of `physics2` only *after* you write `\usepackage{physics2}` in the preamble. Load a `physics2` module like this:

`\usephysicsmodule{\module}`

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

`\usephysicsmodule{ab,ab.braket}`

This line loads the `ab` and `ab.braket` modules.

You can also load *one* module with options. The options of a `physics2` module can be a comma-separated key-value list. For example,

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

These two lines load the `ab` module with option `tightbraces=true` and load `ab.braket` and `doubleprod` modules.



The `common` module will be loaded automatically when you load the `physics2` package and *only* the `common` module will be loaded automatically. Any other module should be loaded manually by writing `\usephysicsmodule{\module}` after you loaded `physics2` in the preamble.

The following sections introduce all the user-level modules of `physics2`. View back to the table of contents to see the names of user-level modules.

2 Modules of `physics2`

2.1 The automatically loaded `common` module

The `common` module provides the following commands:

`\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,

[2.1.1]

`\[\left(\frac{1}{2}\right) 3 \]`
`\[\delopen(\frac{1}{2}\delclose) 3 \]`

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

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

`\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,

$$\left(\left(\left(\left(\left((0)\right)\right)\right)\right)\right)$$
$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \left\{ \frac{1}{2} \right\}$$
$$\left\langle \frac{1}{2} \right\rangle \quad \left| \frac{1}{2} \right| \quad \left\| \frac{1}{2} \right\|$$

[2.2.3] `\def\0{\frac12}`
`\[\pab{\0} \bab{\0} \Bab{\0} \]`
`\[\aab{\0} \vab{\0} \Vab{\0} \]`

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

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

These `\Xab` commands can take an optional star and an optional `[<biggg>]` argument. Star stands for using the default sizes. For example,

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

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

The options of `ab` module `tightbraces`, a bool type key, whose default value is `true`, influences whether thin skips are reserved around the paired delimiters. It only works with the automatically sized delimiters.

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

This module provides four commands — `\bra`, `\ket`, `\braket` and `\ketbra`. After these commands can be a star (*) or a “biggg” command. These commands share similar syntaxes like `\ab`’s syntax. But, *the bra-ket commands from `ab.braket` module are completely different from `\ab`*. Their internal structures are different.

The argument of `\bra` should be delimited with `<` and `|`, that is,

$$\backslash\mathrm{bra} < \langle subformula \rangle |$$

For example,

[2.3.1] `\[\bra < \frac \phi 2 | \]`
`\[\bra* < \frac \phi 2 | \]`
`\[\bra\Big < \phi | \]`

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

$$\left\langle\phi\right|$$

$$\left\langle\phi\right|$$

The argument of `\ket` should be delimited with `|` and `>`, that is,

$$\backslash\mathrm{ket} | \langle subformula \rangle >$$

For example,

[2.3.2]

```
\[ \ket | \frac \psi 2 > \]
\[ \ket*| \frac \psi 2 > \]
\[ \ket\Big| \psi > \]
```

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

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

$$\left| \psi \right\rangle$$



If you want to write “>” and “<” for relations in the argument of `\bra` and `\ket`, you can write `\mathrel{>}` and `\mathrel{<}` (although there is almost no such need).

The argument of `\braket` should be delimited with `<` and `>`, that is,

`\braket < \langle subformula \rangle >`

In the `\langle subformula \rangle` argument, every “|” will be regarded as an extensible vertical bar. For example,

[2.3.3]

```
\[ \braket< \phi > \]
\[ \braket< \phi | \psi > \]
\[ \braket< \phi | A | \psi > \]
```

$$\langle \phi \rangle$$

$$\langle \phi | \psi \rangle$$

$$\langle \phi | A | \psi \rangle$$

[2.3.4]

```
\def\0{\frac\phi2}
\[ \braket < \0 | \psi > \]
\[ \braket* < \0 | \psi > \]
\[ \braket\Bigg< \0 | \psi > \]
```

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

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

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

The argument of `\ketbra` should be delimited with `|` and `|.` In the argument, `>` and `<` will be regarded as extensible `\rangle` and `\langle`. That is,

`\ketbra | \langle subformula1 \rangle > \langle optional \rangle < \langle subformula2 \rangle |`

For example,

[2.3.5]

```
\def\0{\frac\phi2}
\[ \ketbra      | \0 > \psi | \]
\[ \ketbra*     | \0 > \psi | \]
\[ \ketbra\Bigg| \0 > \psi | \]
```

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

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

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

[2.3.6]

```
\def\0{\frac\phi2}
\[ \ketbra | \0 >_x^y < \psi | \]
```

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



If you want to write “>” and “<” for relations in the argument of `\braket` and `\ketbra`, you can write `\>` and `\<` (although there is almost no such need). It is quite different from `\mathrel{>}` or `\mathrel{<}` because in these commands’ argument, `>` and `<` will be redefined.

Next, the `braket` module will be introduced. Please notice that `braket` is conflict with `ab.braket`, they cannot be used together.

2.4 The `braket` module — Dirac bra-ket notation

Please notice that this module is conflict with the `ab.braket` module. Don’t use them together.

This module contains four commands — `\bra`, `\ket`, `\braket` and `\ketbra`. After these commands can be a star (*) or a square-bracket-delimited size option, the size option can take the following values:

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

Star stands for “do not size the bra-ket automatically”.

The argument(s) of these four commands are braced with { and }. `\bra` and `\ket` take one mandatory argument. For example,

[2.4.1]

```
\def\0{\frac\phi2}
\[ \bra {\0} \quad \bra* {\0}
\quad \bra[Big] {\0} \]
\[ \ket {\0} \quad \ket* {\0}
\quad \ket[Big] {\0} \]
```

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

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

The `\braket` command, in default, can take two arguments.

[2.4.2] `\def\0{\frac{\phi}{2}}`
`\[\braket {\0} {\psi} \quad`
`\braket*{\0} {\psi} \quad`
`\braket[big] {\0} {\psi} \]`

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

If you want `\braket` to take one or three arguments, you can write the number of arguments in the square bracket. If you need to specify the size of bra-ket simultaneously, you need to separate the number and the size with a comma:

[2.4.3] `\def\0{\frac{\phi}{2}}`
`\[\braket [1] {\0} \quad`
`\braket*[1] {\0} \]`
`\[\braket [3] {\0}{A}{\psi} \quad \]`
`\[\braket[3,big] {\0}{A}{\psi}`
`\quad`
`\braket[Big,3] {\0}{A}{\psi} \]`

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

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

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

The `\ketbra` command takes two mandatory arguments. It can also take an optional argument between the two mandatory arguments. The optional argument will be placed between \rangle and \langle :

[2.4.4] `\def\0{\frac{\phi}{2}}`
`\[\ketbra {\0} {\psi} \quad`
`\ketbra* {\0} {\psi} \quad \]`
`\[\ketbra [Bigg] {\0} {\psi} \]`
`\[\ketbra {\0} [_x^y] {\psi} \]`

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

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

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

2.5 The **diagmat** module — simple diagonal matrices

This module provides `\diagmat` command:

`\diagmat[empty = \langle empty entry \rangle] {\langle diag \rangle}`

where $\langle diag \rangle$ is the diagonal of the diagonal matrix. The entries should be separated by commas. The `empty` option is optional, with default value \emptyset . For example,

[2.5.1] `\[`
`\diagmat { 1, 2, 3 }`
`\]`

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{matrix}$$

`\pdiagmat`, `\bdiagmat`, `\Bdiagmat`, `\vdiagmat` and `\Vdiagmat` are also available. Prefixes like `p`, `b`, `B` have the same meaning as the `p`, `b`, `B` in `amsmath`'s `pmatrix`, `bmatrix` and `Bmatrix`. For example,

[2.5.2] `\[`
`\pdiagmat [empty = {}]`
`{ a, b, c, d }`
`\]`

$$\begin{pmatrix} a & & & \\ & b & & \\ & & c & \\ & & & d \end{pmatrix}$$

This module requires `amsmath`.

The options of `diagmat` module You can set the default value of `\diagmat`'s empty entries in the module option like this:

```
\usephysicsmodule[empty={\cdot}]{diagmat}
```

2.6 The `doubleprod` module — tensors' double product operator

Take an example of this module:

[2.6.1] `$ A \doublecross B \doubledot C $`

$$A \times B : C$$

`\doublecross` and `\doubledot` are regarded as binary operators by \TeX .

The options of `doubleprod` module You can control the scale of “ \times ” and “ $:$ ” in `\doublecross` and `\doubledot` in module option. For example,

```
\usephysicsmodule[crossscale=0.75,dotscale=1.2]{doubleprod}
```

The default values of `crossscale` and `dotscale` are 0.8 and 1. You can also control the distances between the two “ \times ”s and “ $:$ ”s through the `crossopenup` and `dotopenup` options. For example,

```
\usephysicsmodule[crossopenup=.05,dotopenup=.25]{doubleprod}
```

The default values of `crossopenup` and `dotopenup` are 0.02 and 0.2. The value stands for the multiple of current font size. Moreover, you can change the symbols produced by `\doublecross` and `\doubledot` by setting `crosssymbol` and `dotsymbol` in module option.

2.7 The **xmat** module — matrices with formatted entries

The **xmat** module provides `\xmat` command for matrices with formatted entries:

$$\backslash\mathrm{xmat}[\langle\mathrm{options}\rangle]{\langle\mathrm{entry}\rangle}{\langle\mathrm{rows\ shown}\rangle}{\langle\mathrm{cols\ shown}\rangle}$$

If $\langle\mathrm{rows\ shown}\rangle$ and $\langle\mathrm{cols\ shown}\rangle$ are digits, the value of them must be less at least 2 than the value of **amsmath**'s `MaxMatrixCols` counter. For example,

[2.7.1]
$$\begin{array}{l} \backslash[\\ \backslash\mathrm{xmat}\{a\}\{2\}\{3\} \\ \backslash] \end{array}$$

$$\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{array}$$

`\pxmat`, `\bxmat`, `\Bxmat`, `\vxmat` and `\Vxmat` are also available. The meaning of `p` and so on is the same as the `p` in `pmatrix` of **amsmath**. For example,

[2.7.2]
$$\begin{array}{l} \backslash[\\ \backslash\mathrm{pxmat}\{M\}\{3\}\{3\} \\ \backslash] \end{array}$$

$$\begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{pmatrix}$$

If $\langle\mathrm{rows\ shown}\rangle$ and $\langle\mathrm{cols\ shown}\rangle$ contain non-digit characters, extra dots will be added. For example,

[2.7.3]
$$\begin{array}{l} \backslash[\\ \backslash\mathrm{bxmat}[\mathrm{showleft}=3,\mathrm{showtop}=2] \\ \{X\}\{m\}\{n\} \\ \backslash] \end{array}$$

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} & \cdots & X_{1n} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & X_{m3} & \cdots & X_{mn} \end{bmatrix}$$

In this example we used the `showleft` and `showtop` options. The default value of them is the value of `MaxMatrixCols` minus 2. You can also set them in the module option like this:

$$\backslash\mathrm{usephysicsmodule}[\mathrm{showtop}=3,\mathrm{showleft}=3]{\mathrm{xmat}}$$

Then every `\xmat` with non-digital $\langle\mathrm{rows\ shown}\rangle$ and $\langle\mathrm{cols\ shown}\rangle$ will have 2 top-most rows and 3 left-most columns shown. This will also influence “`\xmat`”s with digital $\langle\mathrm{rows\ shown}\rangle$ and $\langle\mathrm{cols\ shown}\rangle$ when $\langle\mathrm{rows\ shown}\rangle$ and $\langle\mathrm{cols\ shown}\rangle$ are larger than the values corresponding to `showtop` and `showleft`. For example,

[2.7.4]
$$\begin{array}{l} \% \backslash\mathrm{usephysicsmodule} \\ \% [\mathrm{showtop}=3,\mathrm{showleft}=3]{\mathrm{xmat}} \\ \backslash[\backslash\mathrm{pxmat}\{A\}\{8\}\{8\} \backslash] \end{array}$$

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{18} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{28} \\ A_{31} & A_{32} & A_{33} & \cdots & A_{38} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{81} & A_{82} & A_{83} & \cdots & A_{88} \end{pmatrix}$$

However, when $\langle rows\ shown \rangle$ and $\langle cols\ shown \rangle$ are 1 greater than $\langle showtop \rangle$ and $\langle showleft \rangle$, for example, $\langle rows\ shown \rangle = 4$ and $\langle cols\ shown \rangle = 4$ in last example's settings, `\xmat` will still add the extra dots:

```
[2.7.5] % \usephysicsmodule
% [showtop=3,showleft=3]{xmat}
\[ \pxmat{A}{4}{4} \]
```

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{14} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{24} \\ A_{31} & A_{32} & A_{33} & \cdots & A_{34} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{41} & A_{42} & A_{43} & \cdots & A_{44} \end{pmatrix}$$

In such situations, we need to specify `showtop` and `showleft` manually. For example,

```
[2.7.6] % \usephysicsmodule
% [showtop=3,showleft=3]{xmat}
\[ \pxmat[showtop=4,showleft=4]
    {A}{4}{4} \]
```

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix}$$

The `\xmat` command provides the `format` option, which allows users to use a new entry format. For example,

```
[2.7.7] \[
    \xmat [showleft=2,showtop=2,
        format=\texttt{\#1[\#2][\#3]}}
    {x}{m}{n}
\]
```

$$\begin{pmatrix} x[1][1] & x[1][2] & \cdots & x[1][n] \\ x[2][1] & x[2][2] & \cdots & x[2][n] \\ \vdots & \vdots & \ddots & \vdots \\ x[m][1] & x[m][2] & \cdots & x[m][n] \end{pmatrix}$$

In the value of `format` key, `#1` stands for the common entry, or the first mandatory $\langle entry \rangle$ argument of `\xmat`; `#2` stands for the row index and `#3` stands for the column index.

This module requires [amsmath](#).

The options of `xmat` module Only `showtop` and `showleft` can be used as module options. `format` should be only used in the optional argument of the `\xmat` command.

3 The “legacy” modules

The legacy modules have similar names like $\langle module \rangle$.`legacy`. Most of them 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.

3.1 The `ab.legacy` module

This module provides the following commands:

<code>\abs</code>	<code>\norm</code>	<code>\eval</code>	<code>(\peval</code>	<code>\beval)</code>	<code>\order</code>
-------------------	--------------------	--------------------	----------------------	----------------------	---------------------

They share the same syntax as $\langle cmd \rangle^* [\langle biggg \rangle] \{ \langle subformula \rangle \}$. Star and $\langle biggg \rangle$ are optional. Star stands for “use the default size”. For example,

[3.1.1] $\backslash\mathrm{def}\{1+\frac{1}{2}\}$ $\backslash\mathrm{quad}$ $\left|1+\frac{1}{2}\right|$ $\left\|1+\frac{1}{2}\right\|$ $\mathcal{O}(1+\frac{1}{2})$
 $\backslash[\ \backslash\mathrm{abs}\{\backslash 0\}$ $\backslash\mathrm{quad}$
 $\backslash\mathrm{norm}[\mathrm{Big}]\{\backslash 0\}$ $\backslash\mathrm{quad}$
 $\backslash\mathrm{order}^*\{\backslash 0\}$ $\backslash]$

$$[3.1.2] \quad \begin{array}{l} \backslash \text{def}\{1+\frac{1}{2}x\} \\ \backslash [\quad \backslash \text{eval}\{\backslash 0\}_a^b \quad \quad \quad \backslash \text{quad} \\ \quad \quad \backslash \text{peval}*\{\backslash 0\}_a^b \quad \quad \quad \backslash \text{quad} \\ \quad \quad \backslash \text{beval}[\text{bigl}\{\backslash 0\} \ a^b \quad \quad \quad \backslash] \end{array} \quad \boxed{1 + \frac{1}{2}x \Big|_a^b \quad \left(1 + \frac{1}{2}x\right)_a^b \quad \left[1 + \frac{1}{2}x\right]_a^b}$$

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

```
\usephysicsmodule[order=0]{ab.legacy}
```

For further information of this module, see §2.1 of [physics2-legacy](#).

3.2 The `bm-um.legacy` module

If you are maintaining a document with plenty of “`\bm`”s or “`\boldsymbol`”s in it but want to use `unicode-math` package simultaneously, you could take a look at this module.

The `\bm` command from `bm` package uses `\mathversion` to support its function, but there are few OpenType math fonts who released with a bold version. The `bm-um.legacy` module provides a `\bm` command too, but this `\bm` can only take *one* math character or a series of math characters sharing the same category code as its argument. If the argument was Latin letters or Greek letters, `\bm` would switch to the bold italic glyphs corresponding to them (if there exists bold italic glyphs); else `\bm` would switch to the bold upright glyphs. For example,

[3.2.1] $\$ \backslash \mathrm{bm}\{0\} \backslash \mathrm{bm}\{A\} \backslash \mathrm{bm}\{z\}$
 $\backslash \mathrm{bm}\{\backslash \mathrm{alpha}\} \backslash \mathrm{bm}\{\backslash \Omega\} \$$ **0AzαΩ**

3.3 The **nabla.legacy** module

This module provides some commands related to nabla (∇). Notice that this module requires the **fixdif** package with file date 2023/01/31 at minimum.

This module defines `\grad` and `\curl` and redefines `\div`. For example,

[3.3.1] `\[\grad V \]`
`\[\div (x,y,z) \]`
`\[\curl(x,y,z) \]`

$$\nabla V$$

$$\nabla \cdot (x, y, z)$$

$$\nabla \times (x, y, z)$$

The “÷” symbol was redefined as `\divsymbol`.

3.4 The **op.legacy** module

This module provides a series of commands for log-like operators. They are

`\asin` `\acos` `\atan`
`\acsc` `\asec` `\acot`
`\Tr` `\tr` `\rank`
`\erf` `\Res` `\res`
`\PV` `\pv`
`\Re` `\Im`

where `\Re` and `\Im` are redefined. The first four lines of commands yield what they look like in math mode. For example,

[3.4.1] `\$ \asin x \$ \quad \$ \rank A \$`

$$\operatorname{asin} x \quad \operatorname{rank} A$$

`\PV` yields “ \mathcal{P} ” as an ordinary symbol and `\pv` yields “p.v.”. For example,

[3.4.2] `\$ \PV f(z) \$ \quad \$ \pv f(z) \$`

$$\mathcal{P}f(z) \quad \text{p.v. } f(z)$$

`\Re` and `\Im` are redefined as “Re” and “Im”. \Re and \Im are redefined as `\Resymbol` and `\Imsymbol`, in default.

This module *does not* require **amsmath**.

The options of **op.legacy module** `ReIm`, a bool key with default value `true`, determines whether to redefine `\Re` and `\Im`. If you want to reserve the definition of `\Re` and `\Im`, you can write like this:

`\usephysicsmodule[ReIm=false]{op.legacy}`

3.5 The `qtext.legacy` module

This module was written just to offer a method to maintain documents written with the legacy `physics` package. See §2.4 of `physics2-legacy` for more information.