The physics2 package

Zhang Tingxuan

2023/02/07 Version 0.1.1*

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 documention for physics users before you start. If you never used physics package before, just read *this* documentation.

Contents

---

[*] https://www.github.com/AlphaZTX/physics2

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 more than one 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]

```
\[    0 \left(\frac12\right) 3    \]
\[ 0 \delopen(\frac12\delclose) 3 \]
```

$$0 \left(\frac12\right) 3$$

$$0 \mleft(\frac12\mright) 3$$

`\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,

[2.1.2]

```
\[\Biggg(\biggg(\Bigg(\bigg(\Big(\big((
)\big)\Big)\bigg)\Bigg)\biggg)\Biggg)\]
```

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

`\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:

```
 (,  )
 [,  ]
\{,  \}    or    \lbrace,  \rbrace
 <,  >     or    \langle,  \rangle
 |,  |     or      \vert,  \vert
\|,  \|    or      \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:

[2.2.1]

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

$$\left(\frac12\right)\quad\left[\frac12\right]\quad\left\{\frac12\right\}$$

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,

[2.2.2]

```
\[ \ab      <\frac12> \quad
   \ab\biggg|\frac12| \quad
   \ab*    \|\frac12\|   \]
```

$$\left\langle\frac12\right\rangle\quad\biggg|\frac12\biggg|\quad\|\frac12\|$$

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.

The ab module also provides `\`*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,

[2.2.3]

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

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

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

After `\`*X*`ab` can be a “biggg” command or a star. For example,

[2.2.4]

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

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

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,

`\bra <` ⟨*subformula*⟩ `|`

For example,

[2.3.1]

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

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

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

$$\Big\langle\phi\Big|$$

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

`\ket |` ⟨*subformula*⟩ `>`

For example,

[2.3.2]

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

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

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

$$\Big|\psi\Big\rangle$$

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` `<` ⟨*subformula*⟩ `>`

In the ⟨*subformula*⟩ 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}
\[ \braket     < \0 | \psi >  \]
\[ \braket*    < \0 | \psi >  \]
\[ \braket\Bigg< \0 | \psi >  \]
```

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

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

$$\Biggl\langle \frac{\phi}{2} \Biggm| \psi \Biggr\rangle$$

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

`\ketbra` `|` ⟨*subformula*$_1$⟩ `>` ⟨*optional*⟩ `<` ⟨*subformula*$_2$⟩ `|`

For example,

[2.3.5]

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

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

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

$$\Biggl| \frac{\phi}{2} \Biggr\rangle \Biggl\langle \psi \Biggr|$$

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

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

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

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

[2.4.2]

```
\def\0{\frac\phi2}
\[ \braket {\0} {\psi}   \quad
   \braket*{\0} {\psi}   \quad
   \braket[big] {\0} {\psi} \]
```

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

If you want `\braket` to take one or three arguments, you can write the number of arguments in the sqare 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\phi2}
\[ \braket [1] {\0} \quad
   \braket*[1] {\0} \]
\[ \braket [3] {\0}{A}{\psi}    \]
\[ \braket[3,big] {\0}{A}{\psi}
     \quad
   \braket[Big,3] {\0}{A}{\psi} \]
```

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

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

$$\bigl\langle\frac{\phi}{2}\big|A\big|\psi\bigr\rangle \quad \Bigl\langle\frac{\phi}{2}\Big|A\Big|\psi\Bigr\rangle$$

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\phi2}
\[ \ketbra  {\0} {\psi}    \quad
   \ketbra* {\0} {\psi}       \]
\[ \ketbra [Bigg] {\0} {\psi} \]
\[ \ketbra {\0} [_x^y] {\psi} \]
```

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

$$\Bigg|\frac{\phi}{2}\Bigg\rangle\!\Bigg\langle\psi\Bigg|$$

$$\left|\frac{\phi}{2}\right\rangle_{x}^{y}\!\left\langle\psi\right|$$

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

This module provides `\diagmat` command:

`\diagmat[empty=⟨empty entry⟩]{⟨diag⟩}`

where ⟨*diag*⟩ is the diagonal of the diagonal matrix. The entries should be separated by commas. The `empty` option is optional, with default value `0`. 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 $
```

$A \overset{\times}{\times} B : C$

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

The options of doubleprod module You can control the scale of "×" 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 "×"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:

`\xmat[`⟨*options*⟩`]{`⟨*entry*⟩`}{`⟨*rows shown*⟩`}{`⟨*cols shown*⟩`}`

If ⟨*rows shown*⟩ and ⟨*cols shown*⟩ 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]

```
\[
  \xmat{a}{2}{3}
\]
```

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

`\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]

```
\[
  \pxmat{M}{3}{3}
\]
```

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

If ⟨*rows shown*⟩ and ⟨*cols shown*⟩ contain non-digit characters, extra dots will be added. For example,

[2.7.3]

```
\[
  \bxmat[showleft=3,showtop=2]
    {X}{m}{n}
\]
```

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

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:

```
\usephysicsmodule[showtop=3,showleft=3]{xmat}
```

Then every `\xmat` with non-digital ⟨*rows shown*⟩ and ⟨*cols shown*⟩ will have 2 top-most rows and 3 left-most columns shown. This will also influence "`\xmat`"s with digital ⟨*rows shown*⟩ and ⟨*cols shown*⟩ when ⟨*rows shown*⟩ and ⟨*cols shown*⟩ are larger than the values corresponding to `showtop` and `showleft`. For example,

[2.7.4]

```
% \usephysicsmodule
%   [showtop=3,showleft=3]{xmat}
\[ \pxmat{A}{8}{8} \]
```

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

However, when ⟨*rows shown*⟩ and ⟨*cols shown*⟩ are 1 greater than ⟨*showtop*⟩ and ⟨*showleft*⟩, for example, ⟨*rows shown*⟩ $=4$ and ⟨*cols shown*⟩ $=4$ 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]

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

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

In the value of `format` key, `#1` stands for the common entry, or the first mandotary ⟨*entry*⟩ 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 ⟨*module*⟩`.legacy`. Most of them are designed to provide solutions to maintain documents written with the legacy physics package. It’s not suggest to use them in a new document.

3.1 The ab.legacy module

This module provides the following commands:

```
\abs    \norm    \eval    \order
```

They can take a normal argument. Between these commands and their argument can be a “biggg” command or a star. For example,

[3.1.1]

```
\def\0{\frac12}
\[ \abs{\0}            \quad
   \norm\Big{\0}       \quad
   \order*{\0}            \]
```

$$\left|\frac{1}{2}\right| \quad \Big\|\frac{1}{2}\Big\| \quad \mathcal{O}(\frac{1}{2})$$

[3.1.2]

```
\def\0{\frac12x}
\[ \eval{\0}_a^b         \quad
   \eval[(]{\0}_a^b      \quad
   \eval[[]\big{\0}_a^b    \]
```

$$\left.\frac{1}{2}x\right|_a^b \quad \left(\frac{1}{2}x\right|_a^b \quad \big[\frac{1}{2}x\big|_a^b$$

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

```
\usephysicsmodule[order=O]{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]

```
$\bm{0}\bm{A}\bm{z}
 \bm{\alpha}\bm{\Omega}$
```

$\mathbf{0}\boldsymbol{A}\boldsymbol{z}\boldsymbol{\alpha}\mathbf{\Omega}$

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

[3.4.2]

```
$\PV f(z)$ \quad $\pv f(z)$
```

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

`\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.