The overarrows package[*]

Julien Labbé
https://github.com/julienlabbe/latex-packages

April 30, 2025

Abstract

A LaTeX package to create custom arrows over math expressions, mainly for vectors (but arrows can as well be drawn below). Arrows stretch with content, scale with math styles, and have a correct kerning when a subscript follows.

Short example:

```latex
\NewOverArrowCommand{\overrightharpoon}{%
  end=\rightharpoonup
}

\begin{align*}
  &\overrightharpoon{v} && \overrightharpoon{v}_{subscript} \\
  &\overrightharpoon{ABCD} && \overrightharpoon*{v}_{subscript}
\end{align*}
```

$$\begin{aligned} &\overset{\rightharpoonup}{v} && \overset{\rightharpoonup}{v}\ _{subscript} \\ &\overset{\rightharpoonup}{ABCD} && \overset{\rightharpoonup}{v}_{subscript} \end{aligned}$$

Predefined commands are also provided:

- to typeset vectors:
$$\vec{v} \qquad \overrightarrow{AB},$$
- to draw arrows of various shapes above math expressions:
$$\overrightarrow{AB} \qquad \overleftarrow{AB} \qquad \overleftrightarrow{AB} \qquad \overset{\rightharpoonup}{AB} \qquad \overset{\leftharpoonup}{AB} \qquad \overset{\rightharpoondown}{AB} \qquad \overset{\leftharpoondown}{AB} \qquad \overline{AB},$$
- to draw arrows of various shapes under math expressions:
$$\underrightarrow{AB} \qquad \underleftarrow{AB} \qquad \underleftrightarrow{AB} \qquad \underset{\rightharpoonup}{AB} \qquad \underset{\leftharpoonup}{AB} \qquad \underset{\rightharpoondown}{AB} \qquad \underset{\leftharpoondown}{AB} \qquad \underline{AB}.$$

[*]This document corresponds to overarrows v1.4, dated 2025/04/30.

Contents

1 Presentation of the package

The overarrows package allows to create commands for drawing arrows over math expressions. These arrows:

- are fully customisable, at command definition, through a key-value interface;
- stretch with the content and can cover many characters, like in $\overrightarrow{AB}$;
- scale with math styles[1], like in $\vec{v}_{\vec{u}_{\vec{w}}}$.

Commands created with the overarrows package are provided with a starred variant, that removes the extra end space generated by the arrow. This is particularly useful when the command is followed by a subscript. For example, the velocity of the center of mass can be written with exactly the same kerning when scalar v_{cm} or vector $\vec{v}_{\mathrm{cm}}$ (no extra space before the subscript, unlike the output of the unstarred variant: $\vec{v}\,_{\mathrm{cm}}$).

The overarrows package was primitively written for vectors, but in a highly customisable way. It can be used to define a large variety of arrows, using math symbols, or drawing commands from PGF/TikZ or PSTricks. It's also possible to create commands that draw the arrows under. Some predefined commands are provided, giving[2], for arrow over:

$$\overrightarrow{\alpha+\beta} \quad \overleftarrow{\alpha+\beta} \quad \overleftrightarrow{\alpha+\beta} \quad \overset{\rightharpoondown}{\alpha+\beta} \quad \overset{\leftharpoonup}{\alpha+\beta} \quad \overset{\rightharpoondown}{\alpha+\beta} \quad \overset{\leftharpoondown}{\alpha+\beta} \quad \overline{\alpha+\beta}$$

and for arrow under :

$$\underrightarrow{\alpha+\beta} \quad \underleftarrow{\alpha+\beta} \quad \underleftrightarrow{\alpha+\beta} \quad \underset{\rightharpoondown}{\alpha+\beta} \quad \underset{\leftharpoonup}{\alpha+\beta} \quad \underset{\rightharpoondown}{\alpha+\beta} \quad \underset{\leftharpoondown}{\alpha+\beta} \quad \underline{\alpha+\beta}.$$

2 Introduction

2.1 Vector arrows

Vectors are commonly typeset in bold face, or with an arrow above[3]. For this second convention, TEX/LATEXprovides the command `\vec`, which accents its content (using the `\mathaccent` command) with the character $\vec{\ }$ (`\mathchar"017E` in Computer Modern font). But $\vec{\ }$ isn't extensible[4], and gives: $\vec{v}$, $\vec{AB}$ or $\vec{\mathrm{grad}}$ (there's no command `\widevec` analogous to `\widehat`).

An extensible alternative is given by the command `\overrightarrow`, available in TEX/LATEX, and which is redefined by the commonly used amsmath package. But its arrow, built with the `\rightarrow` symbol $\rightarrow$, is too large, using the default *Computer Modern* font: $\overrightarrow{AB}$. Another alternative is the esvect package, which provides the `\vv` command and a set of custom arrows: $\overrightarrow{AB}$, $\overrightarrow{AB}$, $\overrightarrow{AB}$, $\overrightarrow{AB}$, $\overrightarrow{AB}$, $\overrightarrow{AB}$, $\overrightarrow{AB}$, $\overrightarrow{AB}$.

[1] `\displaystyle`, `\textstyle`, `\scriptstyle` and `\scriptscriptstyle`.

[2] Displayed here with the `old-arrows`$^{\to\mathrm{P.}\,16}$ option.

[3] See, for example: International Organization for Standardization. (2019). *Quantities and units – Part 2: Mathematics* (ISO Standard No. 80000-2:2019). `https://www.iso.org/standard/64973.html`.

[4] In fact, with the unicode engines LuaTEX and XƎTEX, the command `\Umathaccent` can now define extensible accents. This is used by the unicode-math package, which also set the arrows displayed by `\vec` and `\overrightarrow` in a coherent manner.

2.2 Stack and arrow macros

It's worth looking at the definition of amsmath `\overrightarrow` command:

```
\long macro:->\mathpalette {\overarrow@ \rightarrowfill@ }
```

Three macros are used here:

`\mathpalette` adapts the output to the current math style;

`\overarrow@` is the *stack macro*, that puts the arrow above the content;

`\rightarrowfill@` is the *arrow macro*, that holds the content of the arrow.

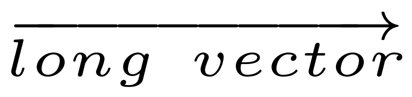
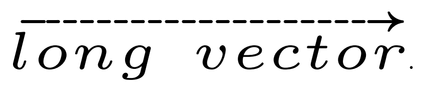
The command `\vv` from esvec is defined with a very similar way, using its own stack macro (`\overvect@`) and arrow macro (`\vectfill@`).

The overarrows package uses the same mechanism. Arrow and stack macros are set, at command creation, through a key-value interface provided by the pgfkeys package (after creation, however, the command definition is static and the key-value interface is not used).

2.3 Extensible arrows

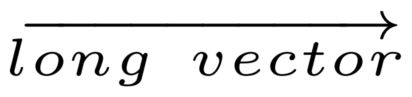
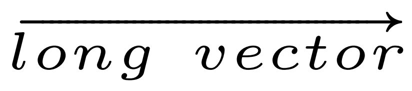
Arrows drawn by the commands `\overrightarrow` or `\vv` are built by joining math symbols, and made extensible by repetition of the central symbol[5]. Thus, the line of the macro `\overrightarrow` is made by repetition of command `\relbar` — (which simply corresponds to the minus sign), while `\vv` use its own command `\relbareda` -.

This method may generate some undesirable spacing issues, when symbols badly overlap. See, for example, the output of amsmath `\overrightarrow` (left) and esvect `\vv` (right) in `\scriptscriptstyle` math style (scaled by a factor 4):

While the arrow on the left lets guess where the symbols — overlap, the arrow on the right present unwanted spaces and show clearly its composition as association of the symbols -, - and →.

By default, the overarrows package uses the same mechanism to extend arrows according to their contents. Settings and tools are provided to perform fine tuning and avoid spacing issues. As example, see below the `\overrightarrow` and `\vv` commands, as redefined by overarrows (in `\scriptscriptstyle` and scaled by a factor 4):

The overarrows package also provides an alternative mechanism. When used, the length `\overarrowlength` is set, according to the arrow command content, and can be employed, for example, to draw arrows using PGF/TikZ, PSTricks or the LaTeX picture environment.

[5]Using the TeX `\cleaders` command.

3 Quick start

3.1 Loading the package overarrows

To load the overarrows, simply add in preamble, before the "`\begin{document}`":

```
\usepackage{overarrows}
```

Options can be given, in a comma-separated list. For example, to use the predefined commands shown in the section 1, page 4, write:

```
\usepackage[allcommands, old-arrows]{overarrows}
```

This define the commands (described in section 4.2.5, page 21):

- `\overrightarrow`$^{\rightarrow\text{P. 21}}$
- `\overleftarrow`$^{\rightarrow\text{P. 21}}$
- `\overleftrightarrow`$^{\rightarrow\text{P. 21}}$
- `\overrightharpoonup`$^{\rightarrow\text{P. 21}}$
- `\overrightharpoondown`$^{\rightarrow\text{P. 21}}$
- `\overleftharpoonup`$^{\rightarrow\text{P. 21}}$
- `\overleftharpoondown`$^{\rightarrow\text{P. 21}}$
- `\overbar`$^{\rightarrow\text{P. 21}}$
- `\underrightarrow`$^{\rightarrow\text{P. 22}}$
- `\underleftarrow`$^{\rightarrow\text{P. 22}}$
- `\underleftrightarrow`$^{\rightarrow\text{P. 22}}$
- `\underrightharpoonup`$^{\rightarrow\text{P. 22}}$
- `\underrightharpoondown`$^{\rightarrow\text{P. 22}}$
- `\underleftharpoonup`$^{\rightarrow\text{P. 22}}$
- `\underleftharpoondown`$^{\rightarrow\text{P. 22}}$
- `\underbar`$^{\rightarrow\text{P. 22}}$

Note that the `old-arrows`$^{\rightarrow\text{P. 16}}$ option may give bad results, if math fonts have been changed. Simply remove the option in this case.

Many other options are available. See the complete list, page 13.

3.2 Commands creation

Commands are created with `\NewOverArrowCommand`$^{\rightarrow\text{P. 17}}$. This macro take two mandatory arguments : the name of the command and the arrow configuration as comma-separated list of key-values. By default, a right arrow is set:

```
\NewOverArrowCommand{\myovercmd}{}
$\myovercmd{test}$
```

$$\overrightarrow{test}$$

Commands are defined with a starred variant, designed to handle subscripts:

```
$ v_{sub} \qquad \myovercmd{v}_{sub} \qquad \myovercmd*{v}_{sub} $
```

$$v_{sub} \qquad \overrightarrow{v}_{sub} \qquad \overrightarrow{v}_{\!sub}$$

3.3 Start and end of the arrow

Extremities of the arrow are set by the keys `start`$^{\rightarrow\text{P. 25}}$ and `end`$^{\rightarrow\text{P. 25}}$. For example, an arrow starting with a hook (symbols `\lhook` $\lhook$) and ending with two heads (symbol `\twoheadrightarrow` $\twoheadrightarrow$) is defined by:

```
\NewOverArrowCommand{\overhooktwoheadrightarrow}{%
  start=\lhook, end=\twoheadrightarrow,
}
```

Note that `\twoheadrightarrow` must be defined, as it is not in LaTeX. This can be done with the package amssymb, by adding in preamble:

```
\usepackage{amssymb}
```

But with the previous definition, the result of the command `\overhooktwoheadrightarrow` is faulty:

```
$ \overhooktwoheadrightarrow{v} \qquad \overhooktwoheadrightarrow{AB} $
```

$$\overhooktwoheadrightarrow{v} \qquad \overhooktwoheadrightarrow{AB}$$

The problem comes from symbols junction and the trimming used to obtain their overlap. It can be solved with the keys `trim start`$^{\rightarrow\text{P. 25}}$ and `trim end`$^{\rightarrow\text{P. 26}}$, which are numbers and set the corresponding trimming in math units (typically `1/18 em`). Appropriate values gives better results:

```
\NewOverArrowCommand{\overhooktwoheadrightarrow}{%
  start=\lhook, end=\twoheadrightarrow,
  trim start=1.5, trim end=2,
}
$ \overhooktwoheadrightarrow{v} \qquad \overhooktwoheadrightarrow{AB} $
```

$$\overhooktwoheadrightarrow{v} \qquad \overhooktwoheadrightarrow{AB}$$

If the math font differs from the default *Computer Modern*, the central part of the arrow may have inappropriate position or line width. This is because the default symbol used for the arrow line is `\relbareda` $\relbareda$ from the esvect package[6]. If needed, try to set the `middle`$^{\rightarrow\text{P. 25}}$ key with the symbol `\relbar` $\relbar$. The trimming should also be adapted:

```
\NewOverArrowCommand{\overhooktwoheadrightarrow}{%
  start=\lhook, end=\twoheadrightarrow, middle=\relbar,
  trim start=0, trim end=3, trim middle=5,
}
$ \overhooktwoheadrightarrow{v} \qquad \overhooktwoheadrightarrow{AB} $
```

$$\overhooktwoheadrightarrow{v} \qquad \overhooktwoheadrightarrow{AB}$$

Finding the correct values for `trim start`$^{\rightarrow\text{P. 25}}$, `trim end`$^{\rightarrow\text{P. 26}}$ and `trim middle`$^{\rightarrow\text{P. 25}}$ may need many trials. For this purpose, the macro `\TestOverArrow`$^{\rightarrow\text{P. 18}}$ displays the result of a command for different lengths and math styles:

[6]Except if the unicode-math package is used with a math font that provides the `\harrowextender` symbol (see the `middle config=auto` key).

```
\TestOverArrow{\overhooktwoheadrightarrow}
```

\displaystyle	\textstyle	\scriptstyle	\scriptscriptstyle
$\displaystyle v$	$\textstyle v$	$\scriptstyle v$	$\scriptscriptstyle v$
$\displaystyle AB$	$\textstyle AB$	$\scriptstyle AB$	$\scriptscriptstyle AB$
$\displaystyle \mathrm{grad}$	$\textstyle \mathrm{grad}$	$\scriptstyle \mathrm{grad}$	$\scriptscriptstyle \mathrm{grad}$
$\displaystyle my\ long\ vector$	$\textstyle my\ long\ vector$	$\scriptstyle my\ long\ vector$	$\scriptscriptstyle my\ long\ vector$

3.4 Size and position of the arrow

A command `\OverRightarrow`, built with the symbols `\Relbar` $=$ and `\Rightarrow` $\Rightarrow$, gives:

```
\NewOverArrowCommand{\OverRightarrow}{%
  start=\Relbar,
  middle=\Relbar,
  end=\Rightarrow,
  trim=4,
}
$ \OverRightarrow{v} \qquad \OverRightarrow{AB} $
```

$\overset{\Longrightarrow}{v} \qquad \overset{\Longrightarrow}{AB}$

The key `trim`$^{\rightarrow\text{P. 26}}$ sets `trim start`$^{\rightarrow\text{P. 25}}$, `trim middle`$^{\rightarrow\text{P. 25}}$ and `trim end`$^{\rightarrow\text{P. 26}}$ with the same value.

The previous arrow is visually too big. The macro `\smallermathstyle`$^{\rightarrow\text{P. 19}}$ allows to obtain a better result:

```
\NewOverArrowCommand{\OverRightarrow}{%
  start={\smallermathstyle\Relbar},
  middle={\smallermathstyle\Relbar},
  end=\Rightarrow,
  trim=4,
}
$ \OverRightarrow{v} \qquad \OverRightarrow{AB} $
```

$\overset{\Rightarrow}{v} \qquad \overset{\Longrightarrow}{AB}$

Note that `\smallermathstyle`$^{\rightarrow\text{P. 19}}$ should not be used for `end`$^{\rightarrow\text{P. 25}}$, because this last is formatted with the same math style as `start`$^{\rightarrow\text{P. 25}}$.

It would be better to add an extra space between the arrow and the content of the command. This can be done with the key `space after arrow`$^{\rightarrow\text{P. 24}}$:

```
\NewOverArrowCommand{\OverRightarrow}{%
  start={\smallermathstyle\Relbar},
  middle={\smallermathstyle\Relbar},
  end=\Rightarrow,
  trim=4,
  space after arrow=0.25ex,
}
$ \OverRightarrow{v} \qquad \OverRightarrow{AB} $
```

$\overset{\Rightarrow}{v} \qquad \overset{\Longrightarrow}{AB}$

Default arrows are slightly shifted to the right. For a left arrow, this should be reversed, using the keys `shift left`$^{\to\text{P. 23}}$ and `shift right`$^{\to\text{P. 23}}$. These keys set the corresponding shifts, in math units. Example:

```
\NewOverArrowCommand{\OverLeftarrow}{%
  start={\smallermathstyle\Leftarrow},
  middle={\smallermathstyle\Relbar},
  end=\Relbar,
  trim=4,
  space after arrow=0.25ex,
  shift left=0, shift right=2,
}
$ \OverLeftarrow{v} \qquad \OverLeftarrow{AB} $
```

$$\overset{\Leftarrow}{v} \qquad \overset{\Longleftarrow}{AB}$$

Finally, the key `arrow under`$^{\to\text{P. 23}}$ places the arrow below the content, instead of above (and `space before arrow`$^{\to\text{P. 24}}$ sets the space upon it):

```
\NewOverArrowCommand{\UnderLeftRightarrow}{%
  start={\smallermathstyle\Leftarrow},
  middle={\smallermathstyle\Relbar},
  end=\Rightarrow,
  trim=4,
  arrow under,
  space before arrow=0.5ex,
  shift left=0, shift right=0,
}
$ \UnderLeftRightarrow{v} \qquad \UnderLeftRightarrow{AB} $
```

$$\underset{\Leftrightarrow}{v} \qquad \underset{\Leftrightarrow}{AB}$$

3.5 Symbols assemblage

Many LaTeX math symbols are built by assemblage, using the macro `\joinrel`[7] which remove 3 math units of horizontal space. The overarrows package provides a flexible version of `\joinrel`, called `\xjoinrel`$^{\to\text{P. 19}}$, which remove an arbitrary number of math units, given as optional argument.

Symbols association is then simple. As example, one can define a triple tail macro `\tttail` from the symbol `\succ` $\succ$:

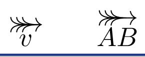
```
\newcommand*{\tttail}{\succ\xjoinrel[10]\succ\xjoinrel[10]\succ}
$ \tttail $
```

$$\succ\!\!\!\succ\!\!\!\succ$$

Thus defined, the macro `\tttail` can be used in arrow definition:

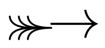
[7]For example, the symbol `\models` $\models$ is defined as `\mathrel{|}\joinrel\Relbar` and corresponds to the assemblage of a vertical line $|$ and the symbol `\Relbar` $=$. The command `\mathrel` modifies the spacing according to the math relation class ; `\Relbar` corresponds to the equal sign (it's definition is `\mathrel{=}`).

```
\NewOverArrowCommand{\overtttailrightarrow}{%
  start={\tttail},
  end={\rightarrow},
  trim start=12,
  shift left=0, shift right=0,
  space after arrow=.2ex,
  min length=24,
}
$ \overtttailrightarrow{v} \qquad \overtttailrightarrow{AB} $
```



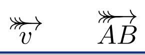
Here the `min length`$^{\to\text{P. 23}}$ key was added to ensure a minimum length (in math units) when the content of the command is small (as for a single character).

The previous arrow would be better with a smaller tail, and this can be done with the macro `\smallermathstyle`$^{\to\text{P. 19}}$. But a small tail and a normal sized head are not aligned; as `{\smallermathstyle\tttail}\xjoinrel[8]\rightarrow}` gives:



The solution comes from the command `\vcenter` which centers materials on math axis. The tail must then be wrapped in a `\hbox`:

```
\NewOverArrowCommand{\overtttailrightarrow}{%
  start={\vcenter{\hbox{$\smallermathstyle\tttail$}}},
  end={\rightarrow},
  trim start=12,
  shift left=0, shift right=0,
  space after arrow=.2ex,
  min length=24,
}
$ \overtttailrightarrow{v} \qquad \overtttailrightarrow{AB} $
```



Text symbols, namely symbols that are not defined in math mode, can also be used. They should yet be enclosed in the `\text` macro, from the amsmath package, to be correctly displayed and correctly scaled according to math style. With, for example, the arrow heads given by the symbols 40 and 41 of the *lasy* font:

```
\newcommand*{\leftarrowhead}{\usefont{U}{lasy}{m}{n}\symbol{40}}
\newcommand*{\rightarrowhead}{\usefont{U}{lasy}{m}{n}\symbol{41}}
\NewOverArrowCommand{\overrightleftarrow}{%
  start=\text{\rightarrowhead},
  end=\text{\leftarrowhead},
  trim start=0.7,  trim end=0.7,
  min length=20,
  shift leftright=-2,
}
$ \overrightleftarrow{AB} \qquad \scriptstyle\overrightleftarrow{AB} $
```

3.6 Drawing the arrow with TikZ

In addition to the default method presented previously (assemblage of symbols, as described in section 2.3, page 5), the overarrows package has an alternative method to draw the arrow. This one allows the use of graphic languages such as PGF/TikZ.

Drawing arrows with TikZ requires to load the `tikz` package and its library `arrows.meta`. This can be simply done by passing the `tikz`$^{\to\text{P. }16}$ option to the overarrows package[8]:

```
\usepackage[tikz]{overarrows}
```

To use PGF/TikZ language, the optional argument `tikz` must be passed to `\NewOverArrowCommand`$^{\to\text{P. }17}$. TikZ pictures are not extensible. That's why the overarrows package provides three lengths that can be used in TikZ commands:

- `\overarrowlength`$^{\to\text{P. }20}$ for the arrow length,
- `\overarrowthickness`$^{\to\text{P. }20}$ and `\overarrowsmallerthickness`$^{\to\text{P. }20}$ for the arrow thickness.

These lengths are computed at each utilisation of a command created with the `tikz` optional argument.

Without any other configuration, a right arrow is drawn:

```
\NewOverArrowCommand[tikz]{\overtikzarrow}{}
$ \overtikzarrow{v} \qquad \overtikzarrow{AB} $
```

$\overrightarrow{v} \qquad \overrightarrow{AB}$

Keys to use Tikz are described in section 4.3.4, page 27. Main keys are: `tikz options`$^{\to\text{P. }27}$, `path options`$^{\to\text{P. }27}$ and `path`$^{\to\text{P. }27}$. It's also possible to append settings with `add tikz options`$^{\to\text{P. }27}$ and `add path options`$^{\to\text{P. }27}$. The full TikZ command used to draw the arrow can as well be entirely redefined with the key `tikz command`$^{\to\text{P. }28}$

Here is a example of an arrow drawn with TikZ[9]:

```
\NewOverArrowCommand[tikz]{\overarchedleftrightarrow}{%
  add tikz options={y=\overarrowlength},
  add tikz options={line width={\overarrowsmallerthickness}},
  path options={arrows={<[scale=0.5]->[scale=0.5]}},
  path={(0,0) arc (-250:70:0.5 and 0.1)},
  center arrow,
  min length=25,
  space after arrow=0.4ex,
}
$ \overarchedleftrightarrow{v} \qquad \overarchedleftrightarrow{ABCD} $
```

[8]Note that the `tikz`$^{\to\text{P. }16}$ option isn't mandatory to use TikZ commands in overarrows. The `tikz` package and its library `arrows.meta` can be loaded independently.

[9]TikZ arrows are very powerfull, but much slower to draw than the default method using assemblage of math symbols.

3.7 Drawing the arrow with PSTricks

In addition to PGF/TikZ, the arrow can be drawn with PSTricks macros. For this, the optional argument `pstricks` must be passed to `\NewOverArrowCommand`$^{\to\text{P. 17}}$. Like with `tikz`, the three lengths `\overarrowlength`$^{\to\text{P. 20}}$, `\overarrowthickness`$^{\to\text{P. 20}}$ and `\overarrowsmallerthickness`$^{\to\text{P. 20}}$ can be used in PSTricks commands. By default, a right arrow is drawn:

```
\NewOverArrowCommand[pstricks]{\overpstarrow}{}
$ \overpstarrow{v} \qquad \overpstarrow{AB} $
```

$\overrightarrow{v} \qquad \overrightarrow{AB}$

The pstricks package has to be loaded (for example, using the `pstricks`$^{\to\text{P. 16}}$ option of the overarrows package)

Keys to use PSTricks commands are described in section 4.3.5, page 28. The main keys are `pstricks command`$^{\to\text{P. 28}}$, `psset`$^{\to\text{P. 28}}$, `arrow`$^{\to\text{P. 28}}$, `geometry`$^{\to\text{P. 28}}$ an `line thickness`$^{\to\text{P. 29}}$. Examples:

```
\NewOverArrowCommand[pstricks]{\overreddisks}{%
  psset={linecolor=red}, arrow=*-*, center arrow,
}
$ \overreddisks{v} \qquad \overreddisks{AB} $
```

$\overset{\bullet\!-\!\bullet}{v} \qquad \overset{\bullet\!-\!-\!-\!\bullet}{AB}$

```
\NewOverArrowCommand[pstricks]{\ellipticarrow}{%
  pstricks command={%
    \psellipticarcn{->}%^^A avoid space before coordinates
    (0.5\overarrowlength,0.2\overarrowlength)%^^A avoid space before coordinates
    (0.5\overarrowlength,0.2\overarrowlength)
    {170}{10}
  },
  geometry={(0,0.2\overarrowlength)(\overarrowlength,0.4\overarrowlength)},
  line thickness={\overarrowsmallerthickness},
  center arrow,
}
$ \ellipticarrow{v} \qquad \ellipticarrow{AB} $
```

$\overset{\curvearrowright}{v} \qquad \overset{\curvearrowright}{AB}$

3.8 Drawing the arrow with LaTeX picture environment

Without any other package, arrows can also be drawn with the LaTeX `picture` environment. In this case, the optional argument `picture` must be passed to `\NewOverArrowCommand`$^{\to\text{P. 17}}$. As with `tikz` or `pstricks`, the three lengths `\overarrowlength`$^{\to\text{P. 20}}$, `\overarrowthickness`$^{\to\text{P. 20}}$ and `\overarrowsmallerthickness`$^{\to\text{P. 20}}$ are available and can be used in `picture` drawing commands. By default, a right vector is drawn:

```
\NewOverArrowCommand[picture]{\overpictarrow}{}
$ \overpictarrow{v} \qquad \overpictarrow{AB} $
```

$\overrightarrow{v} \qquad \overrightarrow{AB}$

If overarrows is loaded with the option `pstarrows`$^{\to\text{P. 17}}$, the package pict2e is used and a PSTricks style vector arrows is set. This gives:

```
\NewOverArrowCommand[picture]{\overpictarrow}{}
$ \overpictarrow{v} \qquad \overpictarrow{AB} $
```

$\overrightarrow{v} \qquad \overrightarrow{AB}$

Keys to use LaTeX `picture` environment are described in section 4.3.6, page 29. The main keys are `picture command`$^{\to\text{P. 29}}$, `geometry`$^{\to\text{P. 29}}$ an `line thickness`$^{\to\text{P. 29}}$. Here is an example:

```
\NewOverArrowCommand[picture]{\overbandedarrow}{
  picture command={%
    \qbezier
    (0.0\overarrowlength,0)
    (0.5\overarrowlength,0)
    (0.9\overarrowlength,0.2\overarrowlength)
    \put(0.9\overarrowlength,0.2\overarrowlength)
    {\vector(2,1){0.2\overarrowlength}}
  },
  geometry={(\overarrowlength,0.4\overarrowlength)(0,0)},
  line thickness={\overarrowsmallerthickness},
  center arrow,
  space after arrow=0.4ex,
}
$ \overbandedarrow{v} \qquad \overbandedarrow{AB} $
```

$\overrightarrow{v} \qquad \overrightarrow{AB}$

4 User interface

4.1 Package options

The overarrows package accepts many options, given as a comma-separated list ⟨*options*⟩ at package loading: `\usepackage[`⟨*options*⟩`]{overarrows}`.

The option `esvect` is set by default. This can be overridden with `noesvect`.

4.1.1 esvect configuration

`esvect`

Loads the esvect package and redefines its vector commands `\vv`$^{\to\text{P. 20}}$ through the overarrows mechanism. Original esvect `\vv` macro is still available with `\esvectvv`$^{\to\text{P. 20}}$. The esvect font description is fixed to allow any font sizes.

The esvect package provides the symbol `\relbareda` - which is smaller and often more flexible than the classic one `\relbar` —. `\relbareda` fits with the standard *Computer Modern* math font, but can be unsuitable with other fonts.

The esvect package also provides the right arrow command `\fldr`. The shape of the arrow depends on the option passed to the esvect package: → (option `a`), → (option `b`), → (option `c`), → (option `d`), → (option `e`), → (option `f`), → (option `g`) or → (option `h`). Note that by default overarrows loads the esvect package with the option `f` (while esvect default is `d`). This can be changed with one of the eight options described bellow: `esvecta`, `esvectb`, `esvectc`, `esvectd`, `esvecte`, `esvectf`, `esvectg` and `esvecth`.

This option is set by default and can be unset with `noesvect`.

`noesvect`

Prevents the loading of the esvect package and the definition of the command `\vv`$^{\to\text{P. 20}}$.

`esvecta`

Loads the esvect package with the `a` option.
`\fldr` corresponds the to the symbol →. `\vv` command gives : $\vec{v}$ $\overrightarrow{AB}$ $\overrightarrow{\mathrm{grad}}$.

`esvectb`

Loads the esvect package with the `b` option.
`\fldr` corresponds the to the symbol →. `\vv` command gives : $\vec{v}$ $\overrightarrow{AB}$ $\overrightarrow{\mathrm{grad}}$.

`esvectc`

Loads the esvect package with the `c` option.
`\fldr` corresponds the to the symbol →. `\vv` command gives : $\vec{v}$ $\overrightarrow{AB}$ $\overrightarrow{\mathrm{grad}}$.

`esvectd`

Loads the esvect package with the `d` option.
`\fldr` corresponds the to the symbol →. `\vv` command gives : $\vec{v}$ $\overrightarrow{AB}$ $\overrightarrow{\mathrm{grad}}$.

`esvecte`

Loads the esvect package with the `e` option.
`\fldr` corresponds the to the symbol →. `\vv` command gives : $\vec{v}$ $\overrightarrow{AB}$ $\overrightarrow{\mathrm{grad}}$.

`esvectf`

Loads the esvect package with the `f` option.
`\fldr` corresponds the to the symbol →. `\vv` command gives : $\vec{v}$ $\overrightarrow{AB}$ $\overrightarrow{\mathrm{grad}}$.

`esvectg`

Loads the esvect package with the `g` option.
`\fldr` corresponds the to the symbol →. `\vv` command gives : $\vec{v}$ $\overrightarrow{AB}$ $\overrightarrow{\mathrm{grad}}$.

`esvecth`

Loads the esvect package with the `h` option.
`\fldr` corresponds the to the symbol →. `\vv` command gives : $\vec{v}$ $\overrightarrow{AB}$ $\overrightarrow{\mathrm{grad}}$.

4.1.2 Predefined commands

The overarrows package provides sixteen predefined commands, eight with the arrow over, and eight with the arrow under. By default, theses commands are not defined, and must be activated by the corresponding option. Beware that commands are created without checking if already defined by another package (`\overleftarrow`, `\overrightarrow`, `\overleftrightarrow`, `\underleftarrow`, `\underrightarrow` and `\underleftrightarrow` are, for example, part of the amsmath package).

Three options are also available to define set of commands.

Set of commands

`allcommands`

Defines all sixteen predefined commands.

`overcommands`

Defines all eight predefined commands with arrow over.

`undercommands`

Defines all eight predefined commands with arrow under.

Over arrows

`overrightarrow`

Defines the `\overrightarrow`→P. 21 command: $\overrightarrow{v}$, $\overrightarrow{AB}$, $\overrightarrow{\mathrm{grad}}$.

`overleftarrow`

Defines the `\overleftarrow`→P. 21 command: $\overleftarrow{v}$, $\overleftarrow{AB}$, $\overleftarrow{\mathrm{grad}}$.

`overleftrightarrow`

Defines the `\overleftrightarrow`→P. 21 command: $\overleftrightarrow{v}$, $\overleftrightarrow{AB}$, $\overleftrightarrow{\mathrm{grad}}$.

`overrightharpoonup`

Defines the `\overrightharpoonup`→P. 21 command: $\overrightharpoonup{v}$, $\overrightharpoonup{AB}$, $\overrightharpoonup{\mathrm{grad}}$.

`overrightharpoondown`

Defines the `\overrightharpoondown`→P. 21 command: $\overrightharpoondown{v}$, $\overrightharpoondown{AB}$, $\overrightharpoondown{\mathrm{grad}}$.

`overleftharpoonup`

Defines the `\overleftharpoonup`→P. 21 command: $\overleftharpoonup{v}$, $\overleftharpoonup{AB}$, $\overleftharpoonup{\mathrm{grad}}$.

`overleftharpoondown`

Defines the `\overleftharpoondown`→P. 21 command: $\overleftharpoondown{v}$, $\overleftharpoondown{AB}$, $\overleftharpoondown{\mathrm{grad}}$.

`overbar`

Defines the `\overbar`→P. 21 command: $\overline{v}$, $\overline{AB}$, $\overline{\mathrm{grad}}$.

Under arrows

`underrightarrow`

Defines the `\underrightarrow`→P. 22 command: $\underrightarrow{v}$, $\underrightarrow{AB}$, $\underrightarrow{\mathrm{grad}}$.

`underleftarrow`

Defines the `\underleftarrow`→P. 22 command: $\underleftarrow{v}$, $\underleftarrow{AB}$, $\underleftarrow{\mathrm{grad}}$.

underleftrightarrow

Defines the `\underleftrightarrow`$^{\rightarrow\text{P. }22}$ command: $\underleftrightarrow{v}$, $\underleftrightarrow{AB}$, $\underleftrightarrow{\mathrm{grad}}$.

underrightharpoonup

Defines the `\underrightharpoonup`$^{\rightarrow\text{P. }22}$ command: $\underset{\rightharpoonup}{v}$, $\underset{\rightharpoonup}{AB}$, $\underset{\rightharpoonup}{\mathrm{grad}}$.

underrightharpoondown

Defines the `\underrightharpoondown`$^{\rightarrow\text{P. }22}$ command: $\underset{\rightharpoondown}{v}$, $\underset{\rightharpoondown}{AB}$, $\underset{\rightharpoondown}{\mathrm{grad}}$.

underleftharpoonup

Defines the `\underleftharpoonup`$^{\rightarrow\text{P. }22}$ command: $\underset{\leftharpoonup}{v}$, $\underset{\leftharpoonup}{AB}$, $\underset{\leftharpoonup}{\mathrm{grad}}$.

underleftharpoondown

Defines the `\underleftharpoondown`$^{\rightarrow\text{P. }22}$ command: $\underset{\leftharpoondown}{v}$, $\underset{\leftharpoondown}{AB}$, $\underset{\leftharpoondown}{\mathrm{grad}}$.

underbar

Defines the `\underbar`$^{\rightarrow\text{P. }22}$ command: $\underline{v}$, $\underline{AB}$, $\underline{\mathrm{grad}}$.

4.1.3 Other options

old-arrows

Loads the old-arrows package with its option `old`. This provides the symbols `\varleftarrow` $\leftarrow$ and `\varrightarrow` $\rightarrow$, used then by default for predefined command.

When the `old-arrows` option is set, the commands `\overrightarrow`$^{\rightarrow\text{P. }21}$, `\overleftarrow`$^{\rightarrow\text{P. }21}$, `\overleftrightarrow`$^{\rightarrow\text{P. }21}$, `\underrightarrow`$^{\rightarrow\text{P. }22}$, `\underleftarrow`$^{\rightarrow\text{P. }22}$ and `\underleftrightarrow`$^{\rightarrow\text{P. }22}$ give respectively : $\overrightarrow{AB}$, $\overleftarrow{AB}$, $\overleftrightarrow{AB}$, $\underrightarrow{AB}$, $\underleftarrow{AB}$ and $\underleftrightarrow{AB}$

tikz

Loads the package tikz with its library `arrows.meta`.

Note that TikZ arrows, drawn with the `tikz` method, are always available, even if this option is not set, provided the tikz package and its library are loaded independently.

pstricks

Loads the package pstricks-add.

Note that, as it, this will compile with LaTeX, LuaLaTeX and XeLaTeX, but not with pdfLaTeX (see the PSTricks documentation). PSTricks arrows, drawn with the `pstricks` method, are always available, even if this option is not set, provided the pstricks package is loaded independently.

`pstarrows`

Loads the `pict2e` package, with its option `pstarrows`. Vectors using LaTeX `picture` environment gives then $\overrightarrow{AB}$ instead of $\overrightarrow{AB}$.

Note that this affect all vectors drawn in LaTeX `picture` environments, and that this setting can be changed on the fly with the commands `\pstarrows` and `\ltxarrows` from the `pict2e` package.

`subscripts`

Sets the default value of the key `detect subscripts`→P. 25 to `true`.

This option also impacts the command `\vv`→P. 20 and all predefined commands, so that they automatically use their starred variant when a subscript follows.

`subother`

New: v1.1 2023/02/15

Sets to 12 (*other* catcode category) the catcode of the "`_`" symbol used for subscript detection, when this is enabled by the key `detect subscripts`→P. 25 (see the section 5.1.2, page 31).

`subactive`

New: v1.1 2023/02/15

Sets to 13 (*active* catcode category) the catcode of the "`_`" symbol used for subscript detection, when this is enabled by the key `detect subscripts`→P. 25 (see the section 5.1.2, page 31).

`debug`

Writes the meaning of defined commands in LaTeX log.

4.2 Commands

4.2.1 Macro for commands creation

`\NewOverArrowCommand[⟨method⟩]{⟨command⟩}{⟨keys⟩}`
`\RenewOverArrowCommand[⟨method⟩]{⟨command⟩}{⟨keys⟩}`
`\ProvideOverArrowCommand[⟨method⟩]{⟨command⟩}{⟨keys⟩}`
`\DeclareOverArrowCommand[⟨method⟩]{⟨command⟩}{⟨keys⟩}`

Creates the command ⟨*command*⟩ and its starred variant ⟨*command*⟩`*`. The starred variant ⟨*command*⟩`*` removes the extra end space generated by the arrow, which is suitable, as example, when a subscript follows.

Updated: v1.2 2024/07/11

⟨*command*⟩ can be given with or without backslash (prior to the version 1.2, only the name, without backslash, was accepted).

`\NewOverArrowCommand` raises an error if ⟨*command*⟩ is already defined.

`\RenewOverArrowCommand` raises an error if ⟨*command*⟩ is undefined.

`\ProvideOverArrowCommand` sets ⟨*command*⟩ if the command is undefined and does nothing if it is already defined, without raising any error.

`\DeclareOverArrowCommand` sets ⟨*command*⟩, whether the command is already defined or not, without raising any error.

The ⟨*method*⟩ used to draw the arrow must be:

`symb` to draw the arrow by symbols assemblage (default);

`tikz` to draw the arrow with PGF/TikZ;

New: v1.2 2024/07/11

`pstricks` to draw the arrow with PSTricks;

`picture` to draw the arrow with the LaTeX `picture` environment.

With no ⟨*method*⟩ argument, the `symb` method is chosen.

⟨*keys*⟩ is a comma-separated list of keys-values. Available keys depends of the ⟨*method*⟩ chosen and are described in section 4.3, page 22.

```
\NewOverArrowCommand[tikz]{\myoverarrow}{arrows={Bar-Bar}, center arrow}
$ \myoverarrow{v} \qquad \myoverarrow{ABCD} $
```

$$\myoverarrow{v} \qquad \myoverarrow{ABCD}$$

`\TestOverArrow[⟨pattern⟩]{⟨command⟩}`
`\TestOverArrow*[⟨pattern⟩]{⟨command⟩}`

Displays the result of the command ⟨*command*⟩ for patterns of various lengths and for the four math styles. A custom ⟨*pattern*⟩ can be added to the predefined ones.

The starred variant `\TestOverArrow*` displays a full report, including kerning tests of the commands ⟨*command*⟩ and ⟨*command*⟩`*`.

Updated: v1.2 2024/07/11

⟨*command*⟩ can be given with or without backslash (prior to the version 1.2, only the name, without backslash, was accepted).

```
\TestOverArrow*[my~pattern]{vv}
```

Test of `\vv` and `\vv*` macros

`\vv` for different math styles

`\displaystyle`	`\textstyle`	`\scriptstyle`	`\scriptscriptstyle`
$\displaystyle\vv{v}$	$\textstyle\vv{v}$	$\scriptstyle\vv{v}$	$\scriptscriptstyle\vv{v}$
$\displaystyle\vv{AB}$	$\textstyle\vv{AB}$	$\scriptstyle\vv{AB}$	$\scriptscriptstyle\vv{AB}$
$\displaystyle\vv{\mathrm{grad}}$	$\textstyle\vv{\mathrm{grad}}$	$\scriptstyle\vv{\mathrm{grad}}$	$\scriptscriptstyle\vv{\mathrm{grad}}$
$\displaystyle\vv{my~long~vector}$	$\textstyle\vv{my~long~vector}$	$\scriptstyle\vv{my~long~vector}$	$\scriptscriptstyle\vv{my~long~vector}$
$\displaystyle\vv{my~pattern}$	$\textstyle\vv{my~pattern}$	$\scriptstyle\vv{my~pattern}$	$\scriptscriptstyle\vv{my~pattern}$

`\vv` kerning

$$\vv{t}_{\vv{u}_{\vv{v}}} \qquad \vv{\imath}_0 \qquad \vv{v} = \vv{v}_x + \vv{v}_y + \vv{v}_z = v_x\vv{\imath} + v_y\vv{\jmath} + v_z\vv{k}$$

`\vv*` kerning

$$\vv*{t}{\vv*{u}{\vv*{v}{}}} \qquad \vv*{\imath}{0} \qquad \vv{v} = \vv*{v}{x} + \vv*{v}{y} + \vv*{v}{z} = v_x\vv{\imath} + v_y\vv{\jmath} + v_z\vv{k}$$

4.2.2 Useful macros for symbols assemblage

Math symbols assemblage is the default method used to draw arrows. The macros `\xjoinrel` and `\smallermathstyle` are designed to help combine and format math symbols.

`\xjoinrel[⟨number⟩]`

Removes an horizontal space of ⟨*number*⟩ math units (`3.5 mu` by default). Must be used in math mode. Useful to assemble math symbols and create new ones.

```
\newcommand*{\triplebar}{\Relbar\xjoinrel[14]\relbar}
\newcommand*{\triplebararrow}{\Relbar\xjoinrel[15]\rightarrow}
\scalebox{2}{$ \triplebar \quad \triplebararrow $} \par
\scalebox{2}{$ \triplebar\xjoinrel\triplebararrow $}
```

$$\equiv \quad \Rrightarrow$$
$$\equiv\!\Rrightarrow$$

`\smallermathstyle`

Applies the next math style, smaller than the current. That is:

- uses `\scriptstyle` if the current math style is `\displaystyle` or `\textstyle`;
- uses `\scriptscriptstyle` if the current math style is `\scriptstyle`;
- does nothing if the current math style is `\scriptscriptstyle`.

```
$ \displaystyle AB \quad \textstyle AB
 \quad \scriptstyle AB \quad \scriptscriptstyle AB $\par
$ \displaystyle AB \quad \smallermathstyle AB
 \quad \smallermathstyle AB \quad \smallermathstyle AB $
```

$\displaystyle AB \quad \textstyle AB \quad \scriptstyle AB \quad \scriptscriptstyle AB$

$\displaystyle AB \quad \scriptstyle AB \quad \scriptscriptstyle AB \quad \scriptscriptstyle AB$

4.2.3 Useful lengths for TikZ, PSTricks or picture environment

Arrows drawn with graphic languages, like PGF/TikZ, PSTricks or the LaTeX `picture` environment, are not extensible. The three lengths `\overarrowlength`, `\overarrowthickness` and `\overarrowsmallerthickness` are computed at each utilisation of a command set with the `tikz`, `pstricks` or `picture` method, so they can be used in drawing commands.

```
\NewOverArrowCommand[tikz]{\overparabola}{%
  path options={x=\overarrowlength, line width=\overarrowsmallerthickness},
  path={(0,0) parabola[parabola height=0.2\overarrowlength] (1,0)},
  arrows={-}, center arrow, min length=30,
}
$\displaystyle \overparabola{v} \qquad \overparabola{ABCD} $ \par
$\scriptstyle \overparabola{v} \qquad \overparabola{ABCD} $ \par
```

$\displaystyle \overparen{v} \qquad \overparen{ABCD}$

$\scriptstyle \overparen{v} \qquad \overparen{ABCD}$

`\overarrowlength`

Is set to the width of the arrow command content, or, if larger, to the minimal arrow length set through the key `min length`$^{\to\text{P. }23}$.

`\overarrowthickness`

Is set to the default rule thickness of the current math style. That is:

- `\fontdimen 8 \textfont 3` in `\displaystyle` or `\textstyle`;
- `\fontdimen 8 \scriptfont 3` in `\scriptstyle`;
- `\fontdimen 8 \scriptscriptfont 3` in `\scriptscriptstyle`.

Updated: v1.2 2024/07/11

Theses settings are adapted when the package unicode-math is loaded (using `\Umathoverbarrule` with LuaLaTeX or `\fontdimen 54`, family 2 with XeLaTeX — see the manual of unicode-math).

`\overarrowsmallerthickness`

Is set to the default rule thickness of the next smaller math style. That is:

- `\fontdimen 8 \scriptfont 3` in `\displaystyle` or `\textstyle`;
- `\fontdimen 8 \scriptscriptfont 3` in `\scriptstyle` or `\scriptscriptstyle`.

Updated: v1.2 2024/07/11

Theses settings are adapted when the package unicode-math is loaded (using `\Umathoverbarrule` with LuaLaTeX or `\fontdimen 54`, family 2 with XeLaTeX — see the manual of unicode-math).

4.2.4 Vectors macros

The macro `\vv`, dedicated to vectors, is automatically defined when the option `esvect`$^{\to\text{P. }13}$ is set (which is the default). It is a clone of the `\vv` command provided by the esvect package, but its starred variant has a correct kerning when followed by a subscript.

`\vv{`⟨*content*⟩`}`
`\vv*{`⟨*content*⟩`}`

Draws a vector arrow upon math ⟨*content*⟩. The shape of the arrow depends on the corresponding options described in section 4.1.1, page 13 : `esvecta`$^{\to\text{P. }14}$, `esvectb`$^{\to\text{P. }14}$, `esvectc`$^{\to\text{P. }14}$, `esvectd`$^{\to\text{P. }14}$, `esvecte`$^{\to\text{P. }14}$, `esvectf`$^{\to\text{P. }14}$, `esvectg`$^{\to\text{P. }14}$, `esvecth`$^{\to\text{P. }14}$.

The starred variant `\vv*` suppresses the end space created by the arrow.

```
$ \vv{\imath}_{0}  \quad \vv{e}_r \quad \vv{L}_\Delta $\par
$ \vv*{\imath}_{0}  \quad \vv*{e}_r \quad \vv*{L}_\Delta $
```

$$\vec{\imath}\,_0 \quad \vec{e}\,_r \quad \vec{L}\,_\Delta$$
$$\vec{\imath}_0 \quad \vec{e}_r \quad \vec{L}_\Delta$$

`\esvectvv`

Is simply the backup of the original esvect `\vv` command.

```
$ \esvectvv{\imath}_{0}  \quad \esvectvv{e}_{r} \quad \esvectvv{L}_\Delta $\par
$ \esvectvv*{\imath}{0}  \quad \esvectvv*{e}{r} \quad \esvectvv*{L}{\Delta} $
```

$$\vec{\imath}_0 \quad \vec{e}_r \quad \vec{L}_\Delta$$
$$\vec{\imath}_0 \quad \vec{e}_r \quad \vec{L}_\Delta$$

4.2.5 Predefined commands

Predefined commands are defined if the corresponding option is set (see section 4.1.2, page 14). The commands `\overrightarrow`, `\overleftarrow`, `\overleftrightarrow`, `\underrightarrow`, `\underleftarrow` and `\underleftrightarrow` are affected by the option `old-arrows`$^{\to \text{P. }16}$.

Over arrows

`\overrightarrow`

$$\overrightarrow{v} \qquad \overrightarrow{AB} \qquad \overrightarrow{\mathrm{grad}}$$

The shape of the arrow is smaller if the option `old-arrows`$^{\to \text{P. }16}$ is set.

`\overleftarrow`

$$\overleftarrow{v} \qquad \overleftarrow{AB} \qquad \overleftarrow{\mathrm{grad}}$$

The shape of the arrow is smaller if the option `old-arrows`$^{\to \text{P. }16}$ is set.

`\overleftrightarrow`

$$\overleftrightarrow{v} \qquad \overleftrightarrow{AB} \qquad \overleftrightarrow{\mathrm{grad}}$$

The shape of the arrows is smaller if the option `old-arrows`$^{\to \text{P. }16}$ is set.

`\overrightharpoonup`

$$\overrightharpoonup{v} \qquad \overrightharpoonup{AB} \qquad \overrightharpoonup{\mathrm{grad}}$$

`\overrightharpoondown`

$$\overrightharpoondown{v} \qquad \overrightharpoondown{AB} \qquad \overrightharpoondown{\mathrm{grad}}$$

`\overleftharpoonup`

$$\overleftharpoonup{v} \qquad \overleftharpoonup{AB} \qquad \overleftharpoonup{\mathrm{grad}}$$

`\overleftharpoondown`

$$\overleftharpoondown{v} \qquad \overleftharpoondown{AB} \qquad \overleftharpoondown{\mathrm{grad}}$$

`\overbar`

$$\overline{v} \qquad \overline{AB} \qquad \overline{\mathrm{grad}}$$

Under arrows

`\underrightarrow`

$$\underrightarrow{v} \qquad \underrightarrow{AB} \qquad \underrightarrow{\mathrm{grad}}$$

The shape of the arrow is smaller if the option `old-arrows`$^{\rightarrow \mathrm{P.}\,16}$ is set.

`\underleftarrow`

$$\underleftarrow{v} \qquad \underleftarrow{AB} \qquad \underleftarrow{\mathrm{grad}}$$

The shape of the arrow is smaller if the option `old-arrows`$^{\rightarrow \mathrm{P.}\,16}$ is set.

`\underleftrightarrow`

$$\underleftrightarrow{v} \qquad \underleftrightarrow{AB} \qquad \underleftrightarrow{\mathrm{grad}}$$

The shape of the arrows is smaller if the option `old-arrows`$^{\rightarrow \mathrm{P.}\,16}$ is set.

`\underrightharpoonup`

$$\underrightharpoonup{v} \qquad \underrightharpoonup{AB} \qquad \underrightharpoonup{\mathrm{grad}}$$

`\underrightharpoondown`

$$\underrightharpoondown{v} \qquad \underrightharpoondown{AB} \qquad \underrightharpoondown{\mathrm{grad}}$$

`\underleftharpoonup`

$$\underleftharpoonup{v} \qquad \underleftharpoonup{AB} \qquad \underleftharpoonup{\mathrm{grad}}$$

`\underleftharpoondown`

$$\underleftharpoondown{v} \qquad \underleftharpoondown{AB} \qquad \underleftharpoondown{\mathrm{grad}}$$

`\underbar`

$$\underline{v} \qquad \underline{AB} \qquad \underline{\mathrm{grad}}$$

4.3 Keys

The customisation of arrows is done at command creation through a key-value interface provided by the pgfkeys package (with `/overarrows/` as key path).

4.3.1 Arrow position and length settings

These keys are available whatever the method chosen at command creation (see section 4.2.1, page 17 for the documentation of commands creation).

Length

`min length={⟨number⟩}` (no default, see below for the initial value)

Sets the minimal arrow length to ⟨*number*⟩ math units. The arrow length is set from content width, or, if larger, to this value.

The initial value of `min length` depends on the ⟨*method*⟩ chosen at command creation (see section 4.2.1, page 17 for the documentation of commands creation):

- ⟨*number*⟩ `= 0` for the `symb` method (method by default);
- ⟨*number*⟩ `= 12` for the `tikz` method;
- ⟨*number*⟩ `= 12` for the `pstricks` method;
- ⟨*number*⟩ `= 18` for the `picture` method.

```
\NewOverArrowCommand{\overlongarrow}{min length=50}
$ \overlongarrow{v} \qquad \overlongarrow{ABCDEF} $
```

$\overrightarrow{\quad v \quad} \qquad \overrightarrow{ABCDEF}$

Placement

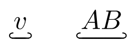
`arrow under` (default `autoconfig`, initially unset)
`arrow under=autoconfig|noconfig`

Places the arrow under, instead of over.

`arrow under` or `arrow under=autoconfig` also configures suitably the key `detect subscripts`$^{\rightarrow\text{P. 25}}$ to `false` and the key `before arrow`$^{\rightarrow\text{P. 24}}$ to get an additional space over the arrow.

`arrow under=noconfig` does not do any additional configuration.

```
\NewOverArrowCommand{\underhooks}{%
  start={\lhook}, end={\rhook},  trim=1,
  arrow under, shift leftright=-4,
}
$ \underhooks{v} \qquad \underhooks{AB} $
```



Horizontal shifts

`shift left={⟨number⟩}` (no default, initially 2)

Shifts the left side of the arrow by ⟨*number*⟩ math units (positive number means a shift to the right).

`shift right={⟨number⟩}` (no default, see below for the initial value)

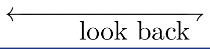
Shifts the right side of the arrow by ⟨*number*⟩ math units (positive number means a shift to the left).

The initial value of `shift right` depends on the ⟨*method*⟩ chosen at command creation (see section 4.2.1, page 17 for the documentation of commands creation):

- ⟨*number*⟩ `= 0` for the `symb` method (method by default);

- ⟨*number*⟩ = -2 for the `tikz`, `pstricks` and `picture` methods.

```
\NewOverArrowCommand{\lookback}{%
  start={\leftarrow}, end={\rightharpoondown},
  shift left=-50, shift right=-10,
}
$ \lookback{\text{look back}} $
```



`shift leftright=[`⟨*number*⟩`]` (no default)

Sets `shift left`$^{\to\text{P. 23}}$ and `shift right`$^{\to\text{P. 23}}$ to the same ⟨*number*⟩ value.

`center arrow`

Sets `shift left`$^{\to\text{P. 23}}$ and `shift right`$^{\to\text{P. 23}}$ to zero.

`left arrow` (default 2)

Sets `shift left`$^{\to\text{P. 23}}$ to zero and `shift right`$^{\to\text{P. 23}}$ to ⟨*number*⟩.

`right arrow` (default 2)

Sets `shift right`$^{\to\text{P. 23}}$ to zero and `shift left`$^{\to\text{P. 23}}$ to ⟨*number*⟩.

Vertical adjunct

`before arrow={`⟨*vertical material*⟩`}` (initially empty)
`after arrow={`⟨*vertical material*⟩`}` (initially empty)

Adds the ⟨*vertical material*⟩ before or after the arrow.

Over and under arrow commands are typeset through the TEX `\ialign` command, which aligns contents, like a tabular. The ⟨*vertical material*⟩ is inserted *between* the rows, with TEX `\noalign` command.

These keys are essentially used to add some extra space between the arrow and the content of the command. They can be set in a handier way with the keys `space before arrow` and `space after arrow`.

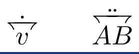
`space before arrow={`⟨*length*⟩`}` (no default)

Adds a space of ⟨*length*⟩ before the arrow. This sets the keys `before arrow`.

`space after arrow={`⟨*length*⟩`}` (no default)

Adds a space of ⟨*length*⟩ after the arrow. This sets the keys `after arrow`.

```
\NewOverArrowCommand{\overharpoonsdown}{%
  start=\leftharpoondown, end=\rightharpoondown, center arrow,
  space before arrow=-0.2ex, space after arrow=0.3ex,
}
$ \dot{\overharpoonsdown{v}} \qquad \ddot{\overharpoonsdown{AB}}$
```



4.3.2 Subscripts detection setting

This key is available whatever the method chosen at command creation (see section 4.2.1, page 17 for the documentation of commands creation).

`detect subscripts=true|false` (default `true`, see below for the initial value)

Removes automatically the extra end space created by the arrow, if a subscript immediately follows the command.

By default, the initial value of `detect subscripts` is `false`. When the option `subscripts`$^{\rightarrow \text{P. 17}}$ is set, the initial value of `detect subscripts` is `true`.

Note that the detection may fail when the standard subscript command is changed or altered (see the section 5.1.2, page 31).

```
\NewOverArrowCommand{\autosub}{detect subscripts}
$ \imath_0 \qquad \autosub{\imath}_0 \qquad
 {\autosub{\imath}}_0 \qquad {\autosub*{\imath}}_0 $
```

$\imath_0 \qquad \overrightarrow{\imath}_0 \qquad {\overrightarrow{\imath}}\,_0 \qquad {\overrightarrow{\imath}}_0$

4.3.3 Symbols assemblage settings

The following keys are available for arrows drawn with the default `symb` method (see section 4.2.1, page 17 for the documentation of commands creation).

`start={⟨command⟩}` (no default, initially `\relbar`)
`middle={⟨command⟩}` (no default, initially set by `middle config=auto`)
`end={⟨command⟩}` (no default, see below for the initial value)

Sets the ⟨*command*⟩ used to draw the start (left), middle (center) or end (right) part of the arrow. The `middle` one is repeated, if necessary, to extend the arrow. It is set, initially by `middle config=auto`. By default, the `end` symbols is initially `\rightarrow` $\rightarrow$. When the option `old-arrows`$^{\rightarrow \text{P. 16}}$ is set, the initial value of `end` is `\varrightarrow` $\rightarrow$.

`start` and `end` symbols are typeset in the same group. `middle` is typeset alone. This means that, if a command, like `\smallermathstyle`$^{\rightarrow \text{P. 19}}$, is used to alter the symbols, it should be applied both to `start` and `middle` (but not to `end`).

```
\NewOverArrowCommand{\smalleroverrightarrow}{%
  start={\smallermathstyle\relbar},
  middle={\smallermathstyle\relbareda},
  end={\rightarrow},
  space after arrow={0.2ex},
}
$ \smalleroverrightarrow{v} \qquad \smalleroverrightarrow{AB} $
```

$\overrightarrow{v} \qquad \overrightarrow{AB}$

`trim start={⟨number⟩}` (no default, initially 7)

Trims ⟨*number*⟩ math units from the right side of the `start` symbol.

`trim middle={⟨number⟩}` (no default, initially set by `middle config=auto`)

Trims ⟨*number*⟩ math units from both left and right sides of the `middle` symbol.

`trim end={⟨number⟩}` (no default, initially 7)

Trims ⟨*number*⟩ math units from the left side of the `end` symbol.

`trim={⟨number⟩}` (no default)

Sets `trim start`→P. 25, `trim middle`→P. 25 and `trim end` to the same ⟨*number*⟩ value.

`no trimming`

Clears `trim start`→P. 25, `trim middle`→P. 25 and `trim end`.

`middle config=auto|relbar|relbareda|harrowextender` (no default)

Sets a suitable configuration for the keys `middle`→P. 25 and `trim middle`→P. 25:

For `middle config = relbar`, `middle`→P. 25 is set to `\relbar` — and `trim middle`→P. 25 to `2.5`.

For `middle config = relbareda`, `middle`→P. 25 is set to `\relbareda` - and `trim middle`→P. 25 to `1`.

New: v1.2 2024/07/11

For `middle config = harrowextender`, `middle`→P. 25 is set to `\harrowextender` and `trim middle`→P. 25 to `0`.

Updated: v1.2 2024/07/11

For `middle config = auto`, if `\harrowextender` is provided by the math font[10], `middle`→P. 25 is set with `middle config = harrowextender`. If `\harrowextender` isn't availlable, `middle`→P. 25 is set with `middle config = relbareda` if the option `esvect`→P. 13 is set (which is the default) and `middle config = relabar` if not.

`amsmath` (default `mimic`)
`amsmath=mimic|strict`

Loads a configuration coherent with amsmath `\overrightarrow` command.

`amsmath` or `amsmath=mimic` sets the corresponding keys suitably:

```
start={\relbar}        middle={\relbar}     end={\rightarrow}
trim start=7           trim middle=2        trim end=7
shift leftright=0      after arrow={}       before arrow={}
```

`amsmath=strict` makes, in addition, the command uses the internal macros of amsmath `\overrightarrow` (`no trimming`, `fill macro={\arrowfill@}`, `stack macro={\overarrow@}`). Note that many configuration keys becomes ineffective.

`esvect` (default `mimic`)
`esvect=mimic|strict`

Loads a configuration coherent with amsmath `\vv` command.

`esvect` or `esvect=mimic` sets the corresponding keys suitably:

```
start={\relbaredd}          middle={\relbareda}        end={\fldr}
trim start=1.5              trim middle=0              trim end=1.5
space before arrow=-.7pt    space after arrow=-.3pt    right arrow=2
```

`esvect=strict` makes, in addition, the command uses the internal macros of esvect `\vv` (`no trimming`, `fill macro={\traitfill@}`, `stack macro={\overvect@}`). Note that many configuration keys becomes ineffective.

[10]See the documentation of the package unicode-math.

4.3.4 TikZ settings

If, at command creation (see section 4.2.1, page 17 for the documentation of commands creation), the `tikz` method is chosen, then the arrow is drawn by the command:

```
\tikz[<tikz options>]{<tikz command>}
```

where `tikz options` and `tikz command`$^{\to \text{P. }28}$ are two keys described below. When `tikz command` is let unset, the drawing command turns into:

```
\tikz[<tikz options>]{\draw[<path options>] <path>;}
```

The best way to customise `tikz` arrows is then to set the keys `tikz options`, `path options` and `path`, preferably through the handy alternatives: `add tikz options`, `add path options`, `arrows`, `line thickness` or `thinner`$^{\to \text{P. }28}$.

```
\NewOverArrowCommand[tikz]{\overdotteddoublearrow}{%
  add tikz options={blue}, add path options={densely dotted},
  arrows={->[scale=0.5]>[scale=0.5]}, thinner,
  min length=20, space after arrow={0.3ex},
}
$ \overdotteddoublearrow{v} \qquad \overdotteddoublearrow{AB} $
```

$\overset{\twoheadrightarrow}{v} \qquad \overset{\twoheadrightarrow}{AB}$

The following keys are available when the `tikz` method is chosen.

`tikz options={`⟨*TikZ options*⟩`}`
(no default, initially `x=\overarrowlength, line width=\overarrowthickness`)

Sets TikZ options to ⟨*TikZ options*⟩.

`path options={`⟨*path options*⟩`}`
(no default, initially `arrows=-Classical TikZ Rightarrow, cap=round`)

Sets TikZ path options to ⟨*path options*⟩.

`path={`⟨*path specification*⟩`}` (no default, initially `(0,0)--(1,0)`)

Sets TikZ path specification to ⟨*path*⟩ (the ending semicolon is automatically appended).

`add tikz options={`⟨*TikZ options*⟩`}` (no default)

Appends the options ⟨*TikZ options*⟩ to the key `tikz options`.

`add path options={`⟨*path options*⟩`}` (no default)

Appends the options ⟨*path options*⟩ to the key `path options`.

`arrows={`⟨*arrow specification*⟩`}` (no default)

Appends the option `arrows={`⟨*arrow specification*⟩`}` to the key `path options`.

`line thickness={`⟨*length*⟩`}` (no default)

Appends the option `line width={`⟨*length*⟩`}` to the key `path options`.

thinner

Sets the keys `line thickness` with `\overarrowsmallerthickness`.

tikz command=`{`⟨*TikZ command*⟩`}` (initially unset)

Sets the ⟨*TikZ command*⟩ used to draw the arrow. If left unset, the value `\draw[path options] path;` is used.

4.3.5 PSTricks settings

New: v1.2 2024/07/11 If, at command creation (see section 4.2.1, page 17 for the documentation of commands creation), the `pstricks` method is chosen, then the arrow is drawn by:

```
\begin{pspicture}<geometry>%
  \psset{linewidth=<line thickness>}%
  \psset{<psset>}%
  <pstricks command>%
\end{pspicture}%
```

where `geometry`, `line thickness`→P. 29 `psset` and `pstricks command` are four keys described below.

```
\NewOverArrowCommand[pstricks]{\overloopandarrow}{
  pstricks command={%
    \pscurve{->}(0,0)
    (0.6\overarrowlength,0.05\overarrowlength)
    (0.5\overarrowlength,0.1\overarrowlength)
    (0.4\overarrowlength,0.05\overarrowlength)
    (\overarrowlength,0)
  },
  geometry={(0,0)(\overarrowlength,0.2\overarrowlength)},
  space after arrow=2pt, min length=20,
  geometry={(0,0)(\overarrowlength,0.2\overarrowlength)},
}
$ \overloopandarrow{v} \qquad \overloopandarrow{AB} $
```

The following keys are available when the `pstricks` method is chosen.

pstricks command=`{`⟨*pstricks command*⟩`}`
(no default, initially `\psline{->}(0,0)(\overarrowlength,0)`)

Sets the `pspicture` command to ⟨*pstricks command*⟩.

arrow=`{`⟨*arrow*⟩`}` (no default, initially `->`)

Sets `pstricks command` with `\psline{`⟨*arrow*⟩`}(0,0)(\overarrowlength,0)`.

psset=`{`⟨*pstricks setting*⟩`}` (no default, initially empty)

Sets ⟨*pstricks setting*⟩ with `\psset`.

geometry=`{`⟨*pstricks geometry specification*⟩`}`
(no default, initially `(0,-0.5ex)(\overarrowlength,1ex)`)

Sets the `pspicture` geometry to ⟨*pstricks geometry specification*⟩.

`line thickness={⟨length⟩}` (no default)

Sets the line thickness to ⟨*length*⟩.

`thinner`

Sets the keys `line thickness` with `\overarrowsmallerthickness`.

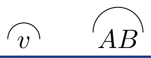
4.3.6 Picture environment settings

If, at command creation (see section 4.2.1, page 17 for the documentation of commands creation), the `picture` method is chosen, then the arrow is drawn by:

```
\begin{picture}<geometry>%
  \linethickness{<line thickness>}%
  <picture command>%
\end{picture}%
```

where `geometry`, `line thickness` and `picture command` are three keys described below.

```
% ^^A \arc and \roundcap commands are from the pict2e package
% ^^A this example needs \usepackage{pict2e} in the preamble
\NewOverArrowCommand[picture]{\overarc}{%
  picture command={%
    \roundcap
    \put(0.5\overarrowlength,0){\arc[180,0]{0.6\overarrowlength}}
  },
  geometry={%
    (1.2\overarrowlength,0.5\overarrowlength)(-0.1\overarrowlength,0.2ex)
  },
  thinner, center arrow,
}
$ \overarc{v} \qquad \overarc{AB} $
```



The following keys are available when the `picture` method is chosen.

`picture command={⟨picture command⟩}`
(no default, initially `\put(0,0){\vector(1,0){\overarrowlength}}`)

Sets picture command to ⟨*picture command*⟩.

`geometry={⟨picture geometry specification⟩}`
(no default, initially `(\overarrowlength,1ex)(0,-0.5ex)`)

Sets picture geometry to ⟨*picture geometry specification*⟩.

`line thickness={⟨length⟩}` (no default)

Sets the picture line thickness to ⟨*length*⟩.

`thinner` (no default)

Sets the keys `line thickness` with `\overarrowsmallerthickness`.

4.4 Advanced commands and keys

The following commands and keys are used in the implementation of the overarrows package. They can also be employed for an advanced configuration of the commands created, although unnecessary in the vast majority of cases.

4.4.1 Advanced commands

`\SetOverArrowsSubscriptCommand{⟨command⟩}`

New: v1.1 2023/02/15

Sets to ⟨*command*⟩ the command used for subscript detection, when this is enabled by the key `detect subscripts`→P. 25 (see the section 5.1.2, page 31).

`\SetOverArrowsMethod[⟨stack mechanism⟩]{⟨name⟩}[⟨pre code⟩]{⟨keys def⟩}`
`\SetOverArrowsMethod*{⟨name⟩}[⟨pre code⟩]{⟨keys def⟩}`

Defines the method ⟨*name*⟩, to be used with `\NewOverArrowCommand`→P. 17, with `\RenewOverArrowCommand`→P. 17, with `\ProvideOverArrowCommand`→P. 17 or with `\DeclareOverArrowCommand`→P. 17. When the ⟨*name*⟩ method is chosen, corresponding keys are defined by ⟨*keys def*⟩. This must set, in particular, the keys `no stack macro hook`→P. 31 and `no arrow macro hook`→P. 31. Optional code ⟨*pre code*⟩ is evaluated before the keys definition.

The unstarred variant automatically defines the key `no stack macro hook`→P. 31, according to the value of the optional ⟨*stack mechanism*⟩. This one must be:

`fill` if `arrow macro` creates extensible arrows (typically with `\cleaders`). In this case, the arrow macro (defined by `no arrow macro hook`→P. 31) is called with the math style, passed as argument (it can be, for example, the macro `\rightarrowfill@` used by amsmath `\overrightarrow`). `fill` is the mechanism used by the `symb` method.

`lens` if `arrow macro` creates fixed-length arrows, and needs the computation of lengths `\overarrowlength`→P. 20, `\overarrowthickness`→P. 20 and `\overarrowsmallerthickness`→P. 20. In this case, the arrow macro (defined by `no arrow macro hook`→P. 31) is called without argument. `lens` is the mechanism used by the `tikz` and `picture` methods.

Without optional ⟨*stack mechanism*⟩, `fill` is used. The starred variant does not set the key `no stack macro hook`→P. 31.

4.4.2 Advanced keys

`stack macro={⟨stack definition⟩}` (no default, initially unset)

Defines the stack macro to be ⟨*stack definition*⟩. Stack macro is a command which takes three arguments: the arrow macro set by `arrow macro`, the math style, and the command content (under or over the arrow). ⟨*stack definition*⟩ can be, for example, the macro `\overarrow@` used by amsmath `\overrightarrow`.

`arrow macro={⟨arrow definition⟩}` (no default, initially unset)

Defines the arrow macro (used in the stack macro) by to be ⟨*arrow definition*⟩.

`no stack macro hook={⟨code⟩}` (no default)

Sets the ⟨*code*⟩ executed if `stack macro` is left unset, after user evaluation of ⟨*keys*⟩ in `\NewOverArrowCommand`→P. 17, `\RenewOverArrowCommand`→P. 17, `\ProvideOverArrowCommand`→P. 17 or `\DeclareOverArrowCommand`→P. 17.

⟨*code*⟩ must configure `stack macro`→P. 30 accordingly to the user keys setting.

`no arrow macro hook={⟨code⟩}` (no default)

Sets the ⟨*code*⟩ executed if `arrow macro`→P. 30 is left unset, after user evaluation of ⟨*keys*⟩ in `\NewOverArrowCommand`→P. 17, `\RenewOverArrowCommand`→P. 17, `\ProvideOverArrowCommand`→P. 17 or `\DeclareOverArrowCommand`→P. 17.

⟨*code*⟩ must configure `arrow macro`→P. 30 accordingly to the user keys setting.

`fill macro={⟨definition⟩}` (no default, initially unset)

Defines the fill macro to be ⟨*definition*⟩. The fill macro is used by arrows created with the `symb` method, to set `arrow macro`→P. 30 in `no arrow macro hook`. It is called with fours arguments: start, middle and end symbols used to draw the arrow, and the math style. ⟨*definition*⟩ can be, for example, the macro `\arrowfill@` used by amsmath `\overrightarrow`.

5 Complements

5.1 Know issues

5.1.1 Math font change

If the math font differs from the default *Computer Modern*, arrow drawn with the `symb` method may have a central part of the arrow with inappropriate position or line width. This is because the default symbol used for the arrow line is `\relbareda` - from the esvect package. This can be fixed with the `noesvect`→P. 14 option.

Depending of the math font, predefined commands may be faulty. For example, at the time of writing, hooks vertical position is incorrect with *Asana Math* or `\harrowextender` is badly positioned with *Stix two Math* (for the smallest math styles), *Libertinus Math* and *GFSNeohellenicMath*.

5.1.2 Detection of non standard subscripts

The subscript detection enabled by the key `detect subscripts`→P. 25 is based on the LaTeX macro `\@ifnextchar`. The detection may fail if the standard subscript command is modified of altered. This is the case, as example:

- with the spbmark package (`https://www.ctan.org/pkg/spbmark`), by Qu Yi, which allows a complete customisation of subscripts, through the `\sub` command;
- with the altsubsup package (`https://www.ctan.org/pkg/altsubsup`), by Julien Labbé, which provides an alternative subscript format, and changes, for this purpose, the catcode of the underscore symbol “`_`” from 8 (*subscript* catcode category) to 12 (*other* catcode category).

To handle theses cases, the command used for subscript detection can be redefined with `\SetOverArrowsSubscriptCommand`→P. 30. Compatibility with the spbmark package is then obtained by:

```
\SetOverArrowsSubscriptCommand{\sub}
```

In the same way, with the altsubsup package, add:

```
\SetOverArrowsSubscriptCommand{_}
```

after the `\begin{document}` (namely, after the catcode redefinition done by altsubsup).

Alternatively, two package options handle the cases where the catcode of the underscore "`_`" symbol is changed: `subother`→P. 17 (for catcode 12, or *other*) and `subactive`→P. 17 (for catcode 13, or *active*). Hence, setting the `subother`→P. 17 option is sufficient for compatibility with the altsubsup package (no need of `\SetOverArrowsSubscriptCommand`→P. 30). Note, that with options `subother`→P. 17 and `subactive`→P. 17, the command `\TestOverArrow*`→P. 18 may give bad results for kerning test, as defined before the catcode redefinition.

5.2 Package dependencies

The following packages are used by overarrows:

- amsmath
- etoolbox
- pgfkeys
- esvect (unless the option `noesvect`→P. 14 is used)
- old-arrows (when the option `old-arrows`→P. 16 is used)
- tikz (when the `tikz` method or the option `tikz`→P. 16 is used)
- pict2e (when the option `pstarrows`→P. 17 is used)

LaTeX distributions prior to 2020/10/01 must load the xparse package before overarrows.

5.3 Alternatives

esvect package (`https://www.ctan.org/pkg/esvect`), by Eddie Saudrais, provides the fine vector macro `\vv`. This package is loaded by default by overarrows.

letterswitharrows package (`https://www.ctan.org/pkg/letterswitharrows`), by Max Teegen, provides left and right over arrows commands, which can extend to multiple characters.

overrightarrow package (`https://www.ctan.org/pkg/overrightarrow`), by Robin Fairbairns, provides the `\Overrightarrow` which is an amalgam of `\overrightarrow` and `\Rightarrow`.

harpoon package (`https://ctan.org/pkg/harpoon`), by Tobias Kuipers, provides over- and under-harpoon symbol commands.

5.4 Changelog

v1.3 Bug fix for `esvect` options (see `https://github.com/julienlabbe/latex-packages/issues/2`).

v1.2
- Fix compatibility issues with unicode-math.
- Allow to draw the arrow with PSTricks.
- Make esvect handle all font sizes.
- Allow backslash in command name for `\NewOverArrowCommand` and variants.
- Rewrite starred variant for better performances.

v1.1 Support for non-standard subscripts.

v1.0.1 Bug fix for under* options.

v1.0 Initial version.

6 Implementation

```
\RequirePackage{etoolbox}
```

Management of options

Declaration of conditionals

```
\newif\ifovar@option@oldarrows@
\newif\ifovar@option@tikz@
\newif\ifovar@option@pstricks@
\newif\ifovar@option@pstarrows@
\newif\ifovar@detectsubscripts@
\newif\ifovar@option@subother@
\newif\ifovar@option@subactive@
\newif\ifovar@option@debug@
```

Following conditionals are for predefined commands.

```
\newif\ifovar@option@overrightarrow@
\newif\ifovar@option@underrightarrow@
\newif\ifovar@option@overleftarrow@
\newif\ifovar@option@underleftarrow@
\newif\ifovar@option@overleftrightarrow@
\newif\ifovar@option@underleftrightarrow@
\newif\ifovar@option@overrightharpoonup@
\newif\ifovar@option@underrightharpoonup@
\newif\ifovar@option@overrightharpoondown@
\newif\ifovar@option@underrightharpoondown@
\newif\ifovar@option@overleftharpoonup@
\newif\ifovar@option@underleftharpoonup@
\newif\ifovar@option@overleftharpoondown@
\newif\ifovar@option@underleftharpoondown@
\newif\ifovar@option@overbar@
\newif\ifovar@option@underbar@
```

Declaration of options

```
\def\ovar@option@esvect{f}
\DeclareOption{esvect}{\gdef\ovar@option@esvect{f}}
\DeclareOption{noesvect}{\gundef\ovar@option@esvect}
\DeclareOption{esvecta}{\gdef\ovar@option@esvect{a}}
\DeclareOption{esvectb}{\gdef\ovar@option@esvect{b}}
\DeclareOption{esvectc}{\gdef\ovar@option@esvect{c}}
\DeclareOption{esvectd}{\gdef\ovar@option@esvect{d}}
\DeclareOption{esvecte}{\gdef\ovar@option@esvect{e}}
\DeclareOption{esvectf}{\gdef\ovar@option@esvect{f}}
\DeclareOption{esvectg}{\gdef\ovar@option@esvect{g}}
\DeclareOption{esvecth}{\gdef\ovar@option@esvect{h}}
\DeclareOption{old-arrows}{\ovar@option@oldarrows@true}
\DeclareOption{tikz}{\ovar@option@tikz@true}
\DeclareOption{pstricks}{\ovar@option@pstricks@true}
\DeclareOption{pstarrows}{\ovar@option@pstarrows@true}
\DeclareOption{subscripts}{\ovar@detectsubscripts@true}
\DeclareOption{subother}{\ovar@option@subother@true}
\DeclareOption{subactive}{\ovar@option@subactive@true}
\DeclareOption{debug}{\ovar@option@debug@true}
```

Following options are for predefined commands.

```
\DeclareOption{overrightarrow}{\ovar@option@overrightarrow@true}
\DeclareOption{underrightarrow}{\ovar@option@underrightarrow@true}
\DeclareOption{overleftarrow}{\ovar@option@overleftarrow@true}
\DeclareOption{underleftarrow}{\ovar@option@underleftarrow@true}
\DeclareOption{overleftrightarrow}{\ovar@option@overleftrightarrow@true}
\DeclareOption{underleftrightarrow}{\ovar@option@underleftrightarrow@true}
\DeclareOption{overrightharpoonup}{\ovar@option@overrightharpoonup@true}
\DeclareOption{underrightharpoonup}{\ovar@option@underrightharpoonup@true}
\DeclareOption{overrightharpoondown}{\ovar@option@overrightharpoondown@true}
\DeclareOption{underrightharpoondown}{\ovar@option@underrightharpoondown@true}
\DeclareOption{overleftharpoonup}{\ovar@option@overleftharpoonup@true}
\DeclareOption{underleftharpoonup}{\ovar@option@underleftharpoonup@true}
\DeclareOption{overleftharpoondown}{\ovar@option@overleftharpoondown@true}
\DeclareOption{underleftharpoondown}{\ovar@option@underleftharpoondown@true}
\DeclareOption{overbar}{\ovar@option@overbar@true}
\DeclareOption{underbar}{\ovar@option@underbar@true}
```

Following options are for sets of predefined commands.

```
\DeclareOption{overcommands}{%
  \ovar@option@overrightarrow@true
  \ovar@option@overleftarrow@true
  \ovar@option@overleftrightarrow@true
  \ovar@option@overrightharpoonup@true
  \ovar@option@overrightharpoondown@true
  \ovar@option@overleftharpoonup@true
  \ovar@option@overleftharpoondown@true
  \ovar@option@overbar@true
}
\DeclareOption{undercommands}{%
  \ovar@option@underrightarrow@true
  \ovar@option@underleftarrow@true
  \ovar@option@underleftrightarrow@true
  \ovar@option@underrightharpoonup@true
  \ovar@option@underrightharpoondown@true
  \ovar@option@underleftharpoonup@true
  \ovar@option@underleftharpoondown@true
  \ovar@option@underbar@true
}
\DeclareOption{allcommands}{%
```

```
  \ovar@option@overrightarrow@true
  \ovar@option@underrightarrow@true
  \ovar@option@overleftarrow@true
  \ovar@option@underleftarrow@true
  \ovar@option@overleftrightarrow@true
  \ovar@option@underleftrightarrow@true
  \ovar@option@overrightharpoonup@true
  \ovar@option@underrightharpoonup@true
  \ovar@option@overrightharpoondown@true
  \ovar@option@underrightharpoondown@true
  \ovar@option@overleftharpoonup@true
  \ovar@option@underleftharpoonup@true
  \ovar@option@overleftharpoondown@true
  \ovar@option@underleftharpoondown@true
  \ovar@option@overbar@true
  \ovar@option@underbar@true
}
```

Options processing

```
\DeclareOption*{\PackageWarning{overarrows}{Unknown option: '\CurrentOption'}}
\ProcessOptions*
```

Package dependencies

LaTeX distributions prior to 2020/10/01 must add the xparse package.

etoolbox is loaded at the very start of the package, as `\gundef` is used at options processing.

```
\RequirePackage{amsmath}
```

Option `old-arrows`$^{\to\text{P. }16}$. Configuration of arrows used for predefined commands.

```
\def\ovar@rightarrow{\rightarrow}
\def\ovar@leftarrow{\leftarrow}
\ifovar@option@oldarrows@
  \RequirePackage[old]{old-arrows}
  \def\ovar@rightarrow{\varrightarrow}
  \def\ovar@leftarrow{\varleftarrow}
\fi
```

Option `esvect`$^{\to\text{P. }13}$.

```
\ifdefined\ovar@option@esvect
  \PassOptionsToPackage{\ovar@option@esvect}{esvect}
  \RequirePackage{esvect}
```

Fix font description in `uesvect.fd` to allow any sizes (taken from Enrico Gregorio, `https://tex.stackexchange.com/a/689863/`)

```
  \DeclareFontFamily{U}{esvect}{}
  \DeclareFontShape{U}{esvect}{m}{n}{
    <-5.5> vect5
    <5.5-6.5> vect6
    <6.5-7.5> vect7
    <7.5-8.5> vect8
    <8.5-9.5> vect9
    <9.5-> vect10
  }{}
\fi
```

Option `tikz`$^{\to\text{P. }16}$.

```
\ifovar@option@tikz@
  \RequirePackage{tikz}
  \usetikzlibrary{arrows.meta}
\fi
```

Option `pstricks`$^{\to\text{P. 16}}$.

```
\ifovar@option@pstricks@
  \RequirePackage{pstricks-add}
\fi
```

Option `pstarrows`$^{\to\text{P. 17}}$.

```
\ifovar@option@pstarrows@
  \RequirePackage[pstarrows]{pict2e}
\fi
```

Add hook rules to apply settings after unicode-math.

```
\DeclareHookRule{begindocument}{overarrows}{after}{unicode-math-luatex}
\DeclareHookRule{begindocument}{overarrows}{after}{unicode-math-xetex}
```

Set `\ovar@auto@middle` and `\ovar@auto@trim@middle`, used by configurations made with `middle config=auto`.

```
\AddToHook{begindocument}[overarrows]
  {%
    \ifdef{\relbareda}
      {%
        \gdef\ovar@auto@middle{\relbareda}
        \gdef\ovar@auto@trim@middle{1}
      }
      {%
        \gdef\ovar@auto@middle{\relbar}
        \gdef\ovar@auto@trim@middle{2.5}
      }%
    \@ifpackageloaded{unicode-math}
      {%
```

Test of `\harrowextender` availability taken from Enrico Gregorio, (`https://tex.stackexchange.com/a/218407/`).

```
        \check@mathfonts
        \iffontchar\textfont\tw@\string"23AF
          \gdef\ovar@auto@middle{\mathrel\harrowextender}
          \gdef\ovar@auto@trim@middle{0}
        \fi
      }
      {}%
  }
```

Configuration of subscripts detection

`\SetOverArrowsSubscriptCommand`

Sets the subscript command.

```
\newcommand{\SetOverArrowsSubscriptCommand}[1]{\global\let\ovar@subcmd=#1}
```

Initial configuration.

```
\SetOverArrowsSubscriptCommand{_}
```

Option `subother`$^{\to\text{P. 17}}$ for *other* (catcode 12) subscript commands.

```
\ifovar@option@subother@
  \begingroup
    \catcode `\_=12
    \SetOverArrowsSubscriptCommand{_}%
  \endgroup
\fi
```

Option `subactive`$^{\rightarrow\text{P. 17}}$ for *active* (catcode 13) subscript commands.

```
\ifovar@option@subactive@
  \begingroup
    \catcode `_=13
    \SetOverArrowsSubscriptCommand{_}%
  \endgroup
\fi
```

Management of keys

Family declaration and setters

```
\RequirePackage{pgfkeys}
\pgfkeys{overarrows/.is family}
```

`\ovar@set`

```
\newcommand{\ovar@set}[1]{\pgfqkeys{/overarrows}{#1}}
```

`\SetOverArrowsMethod`

```
\NewDocumentCommand{\SetOverArrowsMethod}{ s O{fill} m O{} m }{%
  \IfBooleanTF{#1}{%
    \csgdef{ovar@set@#3}{#4\ovar@set{#5}}%
  }{%
    \csgdef{ovar@set@#3}{#4\ovar@set{%
        no stack macro hook/.code={%
          \ovar@set{stack macro/.expanded={%
              \expandafter\expandonce\csname ovar@stack@#2\endcsname%
              {\expandonce\ovar@length@min}%
              {\expandonce\ovar@before@arrow}{\expandonce\ovar@after@arrow}%
            }}%
        },#5}}%
  }%
}
```

Common keys

```
\SetOverArrowsMethod*{common}[\undef{\ovar@macro@stack}\undef{\ovar@macro@arrow}]{%
```

`detect subscripts`$^{\rightarrow\text{P. 25}}$.

```
  detect subscripts/.is if=ovar@detectsubscripts@,
```

`stack macro`$^{\rightarrow\text{P. 30}}$ and `arrow macro`$^{\rightarrow\text{P. 30}}$.

```
  stack macro/.store in=\ovar@macro@stack,
  arrow macro/.store in=\ovar@macro@arrow,
  stack macro/.value required,
  arrow macro/.value required,
```

`no stack macro hook`$^{\rightarrow\text{P. 31}}$, `no arrow macro hook`$^{\rightarrow\text{P. 31}}$. These two keys must be redefined by the command `\ovar@set@`⟨*method*⟩.

```
  no stack macro hook/.code={%
    \PackageError{overarrows}{Undefined stack macro}
    {The requested method is perhaps mispelled}
  },
  no arrow macro hook/.code={%
    \PackageError{overarrows}{Undefined arrow macro}
    {The requested method is perhaps mispelled}
  },
```

`min length`$^{\rightarrow\text{P. 23}}$.

```
  min length/.store in=\ovar@length@min,
  min length/.value required,
  min length=0,
```

`before arrow`$^{\to \mathrm{P.}\,24}$, `after arrow`$^{\to \mathrm{P.}\,24}$, `space before arrow`$^{\to \mathrm{P.}\,24}$, `space after arrow`$^{\to \mathrm{P.}\,24}$.

```
  before arrow/.store in=\ovar@before@arrow,
  after arrow/.store in=\ovar@after@arrow,
  before arrow/.value required,
  after arrow/.value required,
  before arrow=\empty,
  after arrow=\empty,
  space before arrow/.code=\pgfkeysalso{before arrow={\kern ##1}},
  space after arrow/.code=\pgfkeysalso{after arrow={\kern ##1}},
```

`shift left`$^{\to \mathrm{P.}\,23}$, `shift right`$^{\to \mathrm{P.}\,23}$, `shift leftright`$^{\to \mathrm{P.}\,24}$, `center arrow`$^{\to \mathrm{P.}\,24}$, `left arrow`$^{\to \mathrm{P.}\,24}$, `right arrow`$^{\to \mathrm{P.}\,24}$.

```
  shift left/.store in=\ovar@shift@left,
  shift right/.store in=\ovar@shift@right,
  shift left/.value required,
  shift right/.value required,
  shift leftright/.code=\pgfkeysalso{%
    shift left=##1, shift right=##1,
  },
  center arrow/.code=\pgfkeysalso{shift leftright=0},
  shift leftright/.value required,
  center arrow/.value forbidden,
  left arrow/.code=\pgfkeysalso{%
    shift left=0, shift right=##1,
  },
  right arrow/.code=\pgfkeysalso{%
    shift left=##1, shift right=0,
  },
  left arrow/.default=2,
  right arrow/.default=2,
  right arrow,
```

`arrow under`$^{\to \mathrm{P.}\,23}$.

```
  arrow under/.is choice,
  arrow under/noconfig/.code={
    \def\ovar@stack@fill{\ovar@stackunder@fill}
    \def\ovar@stack@lens{\ovar@stackunder@lens}
  },
  arrow under/autoconfig/.code={
    \pgfkeysalso{%
      arrow under=noconfig,
      detect subscripts=false,
      before arrow={\kern 1.3\ex@\relax},% like underarrow@ from amsmath
    }
  },
  arrow under/.default=autoconfig,
}
```

Keys for the symb method

```
\SetOverArrowsMethod{symb}[\undef{\ovar@macro@arrowfill}]{%
```

Fill macro.

```
  fill macro/.store in=\ovar@macro@arrowfill,
  fill macro/.value required,
```

Arrow macro.

```
  no arrow macro hook/.code={%
    \ifdef{\ovar@macro@arrowfill}{}{%
      \ovar@set{%
        fill macro/.expanded={%
          \noexpand\ovar@arrow@fill%
          {\expandonce\ovar@shift@left}{\expandonce\ovar@shift@right}%
        }
      }
    }
    \ovar@set{%
      arrow macro/.expanded={%
        \expandonce{\ovar@macro@arrowfill}%
        {\expandonce{\ovar@arrow@start}\expandonce{\ovar@trim@start}}%
        {\expandonce{\ovar@trim@middle}\expandonce{\ovar@arrow@middle}%
          \expandonce{\ovar@trim@middle}}%
        {\expandonce{\ovar@trim@end}\expandonce{\ovar@arrow@end}}%
      }
    }
  },
```

start→P. 25, middle→P. 25, end→P. 25.

```
  start/.store in=\ovar@arrow@start,
  middle/.store in=\ovar@arrow@middle,
  end/.store in=\ovar@arrow@end,
  start/.value required,
  middle/.value required,
  end/.value required,
```

trim start→P. 25, trim middle→P. 25, trim end→P. 26, trim→P. 26, no trimming→P. 26.

```
  trim start/.code={\def\ovar@trim@start{\xjoinrel[##1]}},
  trim middle/.code={\def\ovar@trim@middle{\xjoinrel[##1]}},
  trim end/.code={\def\ovar@trim@end{\xjoinrel[##1]}},
  trim start/.value required,
  trim middle/.value required,
  trim end/.value required,
  trim/.code={\pgfkeysalso{trim start={##1}, trim middle={##1}, trim end={##1}}},
  trim/.value required,
  no trimming/.code={%
    \let\ovar@trim@start\empty
    \let\ovar@trim@middle\empty
    \let\ovar@trim@end\empty
  },
  no trimming/.value forbidden,
```

middle config→P. 26.

```
  middle config/.is choice,
  middle config/.value required,
  middle config/relbar/.code=\pgfkeysalso{%
    middle={\relbar},
    trim middle={2.5},
  },
  middle config/relbareda/.code={%
    \ifundef{\relbareda}{%
      \PackageWarning{overarrows}{Key 'middle config=relbareda' used,
        \MessageBreak%
        but \protect\relbareda\space is undefined; ignored.
        \MessageBreak%
        Load 'esvect' package, or use 'esvect' option \MessageBreak%
        to remove this warning}
    }{%
      \pgfkeysalso{%
```

```
        middle={\relbareda},
        trim middle={1},
      }
    }
  },
```

```
  middle config/harrowextender/.code={%
    \pgfkeysalso{%
      middle={\harrowextender},
      trim middle={0},
    }
  },
```

Set `middle config` with (hopefully) a good configuration. It would be better to reuse the previous `middle config` settings, but we have to wait for the `begindocument` hook to know which one to use.

```
  middle config/auto/.code={%
    \pgfkeysalso{%
      middle={\ovar@auto@middle},
      trim middle={\ovar@auto@trim@middle},
    }
  },
```

`amsmath`→P. 26.

```
  amsmath/.is choice,%
  amsmath/mimic/.code=\pgfkeysalso{%
    start={\relbar}, middle={\relbar}, end={\rightarrow},
    trim start=7,
    trim middle=2,
    trim end=7,
    shift leftright=0,
    after arrow={}, before arrow={},
  },
  amsmath/strict/.code=\pgfkeysalso{%
    amsmath=mimic,
    no trimming,
    fill macro={\arrowfill@}, stack macro={\overarrow@},
  },
  amsmath/.default=mimic,
```

`esvect`→P. 26.

```
  esvect/.is choice,%
  esvect/mimic/.code=\pgfkeysalso{%
    start={\relbaredd}, middle={\relbareda}, end={\fldr},
    trim start=1.5,
    trim end=1.5,
    trim middle=0,
    right arrow=2,
    space before arrow=-.7pt,
    space after arrow=-.3pt,
  },
  esvect/strict/.code=\pgfkeysalso{%
    esvect=mimic,
    no trimming,
    fill macro={\traitfill@}, stack macro={\overvect@},
  },
  esvect/.default=mimic,
```

Initial configuration.

```
  amsmath, middle config=auto, end=\ovar@rightarrow, right arrow,
}
```

Keys for the tikz method

```
\SetOverArrowsMethod[lens]{tikz}[\undef{\ovar@tikz@command}]{%
```

Arrow macro.

```
  no arrow macro hook/.code={%
    \ifdef{\ovar@tikz@command}{}{%
      \pgfkeysgetvalue{/overarrows/path options}{\ovar@tikz@pathoptions}
      \ovar@set{%
        tikz command/.expanded={%
          \noexpand\draw[\expandonce\ovar@tikz@pathoptions]\expandonce\ovar@tikz@path;
        }
      }
    }
    \pgfkeysgetvalue{/overarrows/tikz options}{\ovar@tikz@options}
    \ovar@set{%
      arrow macro/.expanded={%
        $\noexpand\mkern \expandonce{\ovar@shift@left} mu\noexpand\relax$%
        \noexpand\tikz[\expandonce{\ovar@tikz@options}]{\expandonce{\ovar@tikz@command}}%
        $\noexpand\mkern \expandonce{\ovar@shift@right} mu\noexpand\relax$%
      }
    }
  },
```

TikZ parts: `tikz command`$^{\rightarrow \text{P. } 28}$, `tikz options`$^{\rightarrow \text{P. } 27}$, `path options`$^{\rightarrow \text{P. } 27}$, `path`$^{\rightarrow \text{P. } 27}$.

```
  tikz command/.store in=\ovar@tikz@command,
  tikz options/.initial={x=\overarrowlength, line width=\overarrowthickness},
  path options/.initial={arrows={-Classical TikZ Rightarrow}, cap=round},
  path/.store in=\ovar@tikz@path,
  path={(0,0)--(1,0)},
  tikz command/.value required,
  tikz options/.value required,
  path options/.value required,
  path/.value required,
```

TikZ handy keys: `add path options`$^{\rightarrow \text{P. } 27}$, `add tikz options`$^{\rightarrow \text{P. } 27}$, `arrows`$^{\rightarrow \text{P. } 27}$, `line thickness`$^{\rightarrow \text{P. } 27}$, `thinner`$^{\rightarrow \text{P. } 28}$.

```
  add path options/.code=\pgfkeysalso{%
    path options/.append={, ##1}},%
  add tikz options/.code=\pgfkeysalso{%
    tikz options/.append={, ##1}},%
  arrows/.code=\pgfkeysalso{add path options={arrows={##1}}},%
  line thickness/.code=\pgfkeysalso{add path options={line width=##1}},%
  thinner/.code=\pgfkeysalso{line thickness={\overarrowsmallerthickness}},%
  add path options/.value required,%
  add tikz options/.value required,%
  arrows/.value required,%
  line thickness/.value required,%
  thinner/.value forbidden,%
```

Initial configuration.

```
  shift right=-2,
  min length=12,
}
```

Keys for the pstricks method

```
\SetOverArrowsMethod[lens]{pstricks}{%
```

Arrow macro.

```
  no arrow macro hook/.code={%
    \ovar@set{%
      arrow macro/.expanded={%
        $\noexpand\mkern \expandonce{\ovar@shift@left} mu\noexpand\relax$%
        \noexpand\begin{pspicture}\expandonce{\ovar@pstricks@geometry}%
          \noexpand\psset{linewidth=\expandonce{\ovar@pstricks@linethickness}}%
          \noexpand\psset{\expandonce{\ovar@pstricks@psset}}%
          \expandonce{\ovar@pstricks@command}%
        \noexpand\end{pspicture}%
        $\noexpand\mkern \expandonce{\ovar@shift@right} mu\noexpand\relax$%
      }
    }
  },
```

Pstricks parts: `pstricks command`→P. 28, `psset`→P. 28, `geometry`→P. 28, `line thickness`→P. 29.

```
  pstricks command/.store in=\ovar@pstricks@command,
  psset/.store in=\ovar@pstricks@psset,
  geometry/.store in=\ovar@pstricks@geometry,
  line thickness/.store in=\ovar@pstricks@linethickness,
  pstricks command/.value required,
  psset/.value required,
  geometry/.value required,
  line thickness/.value required,
```

Pstricks handy key: `arrow`→P. 28, `thinner`→P. 29.

```
  arrow/.style={pstricks command={\psline{##1}(0,0)(\overarrowlength,0)}},%
  arrow/.value required,%
  thinner/.style={line thickness={\overarrowsmallerthickness}},%
  thinner/.value forbidden,%
```

Initial configuration.

```
  shift right=-2,
  min length=12,
  geometry={(0,-0.5ex)(\overarrowlength,0.5ex)},%
  line thickness={\overarrowthickness},%
  arrow={->},%
  psset={},%
}
```

Keys for the picture method

```
\SetOverArrowsMethod[lens]{picture}{%
```

Arrow macro.

```
  no arrow macro hook/.code={%
    \ovar@set{%
      arrow macro/.expanded={%
        $\noexpand\mkern \expandonce{\ovar@shift@left} mu\noexpand\relax$%
        \noexpand\begin{picture}\expandonce{\ovar@picture@geometry}%
          \noexpand\linethickness{\expandonce{\ovar@picture@linethickness}}%
          \expandonce{\ovar@picture@command}%
          \noexpand\end{picture}%
        $\noexpand\mkern \expandonce{\ovar@shift@right} mu\noexpand\relax$%
      }
    }
  },
```

Picture parts: `picture command`→P. 29, `geometry`→P. 29, `line thickness`→P. 29.

```
  picture command/.store in=\ovar@picture@command,
  geometry/.store in=\ovar@picture@geometry,
  line thickness/.store in=\ovar@picture@linethickness,
  picture command/.value required,
  geometry/.value required,
  line thickness/.value required,
```

Picture handy key: `thinner`→ P. 29.

```
  thinner/.code=\pgfkeysalso{line thickness={\overarrowsmallerthickness}},
```

Initial configuration.

```
  shift right=-2,
  min length=18,
  geometry={(\overarrowlength,1ex)(0,-0.5ex)},%
  line thickness={\overarrowthickness},%
  picture command={\put(0,0){\vector(1,0){\overarrowlength}}},%
}
```

Commands

Macros for symbols assemblage

`\xjoinrel`

```
\ifdef{\xjoinrel}{%
  \PackageWarning{overarrows}{Command \protect\xjoinrel\space already defined.
    \MessageBreak%
    Previous definition will be overridden}
}{}
```

Use a default value of `3.5 mu`, as recommended by Enrico Gregorio (see `https://tex.stackexchange.com/a/471736`). `\joinrel` uses a value of 3 mu.

```
\DeclareRobustCommand{\xjoinrel}[1][3.5]{\mathrel{\mkern-#1mu}}
```

`\smallermathstyle`

```
\newcommand*{\smallermathstyle}{%
  \mathchoice{\scriptstyle}{\scriptstyle}{\scriptscriptstyle}{}
}
```

`\ovar@arrow@fill`

Macro used for default `fill macro`→ P. 31.
#1: left shift
#2: right shift
#3: arrow start
#4: arrow middle
#5: arrow end
#6: math style

```
\def\ovar@arrow@fill#1#2#3#4#5#6{%
  $\m@th\thickmuskip0mu\medmuskip\thickmuskip\thinmuskip\thickmuskip\relax%
  \mkern #1 mu\relax#6#3%
  \cleaders\hbox{$#6#4$}\hfill%
  #5\mkern #2 mu\relax$%
}
```

Macros for fixed length arrows

Lengths declaration.

```
\newlength{\overarrowlength}
\newlength{\overarrowthickness}
\newlength{\overarrowsmallerthickness}
\newlength{\ovar@tempdim}
```

`\ovar@set@arrowlength`

Sets `\overarrowlength`→ P. 20.
#1: min length, in math units
#2: math style
#3: content

```
\def\ovar@set@arrowlength#1#2#3{%
  \settowidth{\ovar@tempdim}{$\m@th#2\mskip #1 mu\relax$}%
  \settowidth{\overarrowlength}{$\m@th#2#3$}%
  \ifdim \overarrowlength < \ovar@tempdim \overarrowlength=\ovar@tempdim\fi%
}
```

`\ovar@set@arrowthickness`

`\ovar@set@arrowthickness@UM@lua`

Sets `\overarrowthickness`$^{\rightarrow \text{P. } 20}$ and `\overarrowsmallerthickness`$^{\rightarrow \text{P. } 20}$.

`#1`: math style

Set to the default rule thickness of the current math style, normaly given by `\fontdimen` 8 family 3. With unicode-math, use instead:

- `\fontdimen` 54 family 2 with XeTeX,
- `\Umathoverbarrule` with LuaTex.

```
\def\ovar@rulethickness@fontdimen{8}
\def\ovar@rulethickness@family{3}
\def\ovar@set@arrowthickness#1{%
  \ifx#1\displaystyle%
    \overarrowthickness =
      \fontdimen \ovar@rulethickness@fontdimen  \textfont \ovar@rulethickness@family%
    \overarrowsmallerthickness =
      \fontdimen \ovar@rulethickness@fontdimen \scriptfont \ovar@rulethickness@family%
  \else\ifx#1\textstyle%
    \overarrowthickness =
      \fontdimen \ovar@rulethickness@fontdimen \textfont \ovar@rulethickness@family%
    \overarrowsmallerthickness =
      \fontdimen \ovar@rulethickness@fontdimen \scriptfont \ovar@rulethickness@family%
  \else\ifx#1\scriptstyle%
    \overarrowthickness =
      \fontdimen \ovar@rulethickness@fontdimen \scriptfont \ovar@rulethickness@family%
    \overarrowsmallerthickness =
      \fontdimen \ovar@rulethickness@fontdimen \scriptscriptfont \ovar@rulethickness@family%
  \else%
    \overarrowthickness =
      \fontdimen \ovar@rulethickness@fontdimen \scriptscriptfont \ovar@rulethickness@family%
    \overarrowsmallerthickness = \overarrowthickness%
  \fi\fi\fi%
}
```

unicode-math with LuaTeX version.

```
\def\ovar@set@arrowthickness@UM@lua#1{%
  \overarrowthickness = \Umathoverbarrule #1
  \ifx#1\displaystyle%
    \overarrowsmallerthickness = \Umathoverbarrule \textstyle%
  \else\ifx#1\textstyle%
    \overarrowsmallerthickness = \Umathoverbarrule \scriptstyle%
  \else%
    \overarrowsmallerthickness = \Umathoverbarrule \scriptscriptstyle%
  \fi\fi%
}
```

Test which version to use.

```
\AddToHook{begindocument}[overarrows]
  {%
    \@ifpackageloaded{unicode-math-luatex}
      {%
        \global\let\ovar@set@arrowthickness\ovar@set@arrowthickness@UM@lua
      }
      {%
        \@ifpackageloaded{unicode-math-xetex}
        {%
          \gdef\ovar@rulethickness@fontdimen{54}
          \gdef\ovar@rulethickness@family{2}
        }
        {}
    }
  }
```

Stack macros

\ovar@stackover@@

\ovar@stackunder@@

Bases of all stack macros.
#1: min length, in math units
#2: vertical mode material before arrow
#3: vertical mode material after arrow
#4: arrow
#5: math style
#6: content

```
\def\ovar@stackover@@#1#2#3#4#5#6{\vbox{\ialign{##\crcr%
      $#5\mskip #1 mu\relax$\crcr%
      \noalign{#2\nointerlineskip}#4\crcr%
      \noalign{#3\nointerlineskip}%
      $\m@th\hfil#5#6\hfil$\crcr%
    }%
  }%
}
\def\ovar@stackunder@@#1#2#3#4#5#6{\vtop{\ialign{##\crcr%
      $\m@th\hfil#5#6\hfil$\crcr%
      \noalign{#2\nointerlineskip}#4\crcr%
      \noalign{#3\nointerlineskip}%
      $#5\mskip #1 mu\relax$\crcr%
    }%
  }%
}
```

\ovar@stackover@

\ovar@stackunder@

Stack macros without min arrow length.
#1: vertical mode material before arrow
#2: vertical mode material after arrow
#3: arrow macro
#4: math style
#5: content

```
\def\ovar@stackover@#1#2#3#4#5{\ovar@stackover@@{0}{#1}{#2}{#3}{#4}{#5}}
\def\ovar@stackunder@#1#2#3#4#5{\ovar@stackunder@@{0}{#1}{#2}{#3}{#4}{#5}}
```

\ovar@stackover@fill

\ovar@stackunder@fill

\ovar@stack@fill

Stack macros for extensible arrows.
#1: min length, in math units
#2: vertical mode material before arrow
#3: vertical mode material after arrow
#4: arrow filler macro
#5: math style
#6: content

```
\def\ovar@stackover@fill#1#2#3#4#5#6{\ovar@stackover@@{#1}{#2}{#3}{#4#5}{#5}{#6}}
\def\ovar@stackunder@fill#1#2#3#4#5#6{\ovar@stackunder@@{#1}{#2}{#3}{#4#5}{#5}{#6}}
```

`\ovar@stack@fill` matches the macro `\ovar@stackover@fill` by default, or `\ovar@stackunder@fill` with `arrow under`→ P. 23.

```
\def\ovar@stack@fill{\ovar@stackover@fill}
```

\ovar@stackover@lens
\ovar@stackunder@lens
\ovar@stack@lens

Stack macros for fixed-length arrows (these call `\ovar@set@arrowlength` and `\ovar@set@arrowthickness`).
`#1`: min length, in math units
`#2`: vertical mode material before arrow
`#3`: vertical mode material after arrow
`#4`: arrow content macro
`#5`: math style
`#6`: content

```
\def\ovar@stackover@lens#1#2#3#4#5#6{%
  \ovar@set@arrowlength{#1}{#5}{#6}%
  \ovar@set@arrowthickness{#5}%
  \ovar@stackover@{#2}{#3}{#4}{#5}{#6}%
}
\def\ovar@stackunder@lens#1#2#3#4#5#6{%
  \ovar@set@arrowlength{#1}{#5}{#6}%
  \ovar@set@arrowthickness{#5}%
  \ovar@stackunder@{#2}{#3}{#4}{#5}{#6}%
}
```

`\ovar@stack@lens` matches the macro `\ovar@stackover@lens` by default, or `\ovar@stackunder@lens` with `arrow under`$^{\to \text{P. } 23}$.

```
\def\ovar@stack@lens{\ovar@stackover@lens}
```

Macro for commands creation

In the initial version, the commands names must be given as csname (without backslash). To harmonize the syntax with standard `\NewDocumentCommand`, define an argument processor so that both `\NewOverArrowCommand{\myarrow}` and `\NewOverArrowCommand{myarrow}` are accepted.

```
\ExplSyntaxOn
\cs_new_protected:Npn \__overarrows_processor_strip_escape_char:n #1
  {
     \regex_match:nnTF { ^\cC. } { #1 }
     { \tl_set:Nx \ProcessedArgument { \cs_to_str:N #1 } }
     { \tl_set:Nx \ProcessedArgument { #1 } }
  }
\cs_new_eq:NN \ovar@cmdname@processor \__overarrows_processor_strip_escape_char:n
\ExplSyntaxOff
```

\DeclareOverArrowCommand

```
\NewDocumentCommand{\DeclareOverArrowCommand}{
 O{symb} >{\ovar@cmdname@processor} m m
}{%
  \begingroup
  \ovar@set@common
  \ifcsdef{ovar@set@#1}{%
    \csuse{ovar@set@#1}
  }{%
    \PackageError{overarrows}{Unknown method #1}
    {Try with 'symb', 'tikz', 'pstriks' or 'picture'}
  }
  \ovar@set{#3}
  \ifdef{\ovar@macro@arrow}{}{%
    \ovar@set{no arrow macro hook}
  }
  \ifdef{\ovar@macro@stack}{}{%
    \ovar@set{no stack macro hook}
  }
  \csxdef{ovar@#2@normal}{%
    \noexpand\mathpalette{%
      \expandonce{\ovar@macro@stack}{\expandonce{\ovar@macro@arrow}}%
    }
  }
  \csxdef{ovar@#2@starred}{%
    \noexpand\mathpalette{%
      \noexpand\ovar@starversion{%
        \expandonce{\ovar@macro@stack}{\expandonce{\ovar@macro@arrow}}%
      }
    }
  }
  \ifovar@option@debug@
  \PackageInfo{overarrows}{%
    Meaning of \protect\ovar@#2@normal\MessageBreak
    used for \@backslashchar#2:\MessageBreak%
      \expandafter\meaning\csname ovar@#2@normal\endcsname}
  \fi
```

Expand `\ifovar@detectsubscripts@` before closing the group, then define the command.

```
  \expandafter%
  \endgroup
  \ifovar@detectsubscripts@%
  \csgdef{ovar@#2@auto}##1{%
    \@ifnextchar \ovar@subcmd {%
      \csuse{ovar@#2@starred}{##1}%
    }{%
      \csuse{ovar@#2@normal}{##1}%
    }%
  }
  \expandafter\DeclareDocumentCommand\csname #2\endcsname { s }{%
    \IfBooleanTF{##1}{\csuse{ovar@#2@starred}}{\csuse{ovar@#2@auto}}%
  }%
  \else
  \expandafter\DeclareDocumentCommand\csname #2\endcsname { s }{%
    \IfBooleanTF{##1}{\csuse{ovar@#2@starred}}{\csuse{ovar@#2@normal}}%
  }%
  \fi
}
```

`\ProvideOverArrowCommand`

```
\NewDocumentCommand{\ProvideOverArrowCommand}{
 O{symb} >{\ovar@cmdname@processor} m m
}{%
  \ifcsdef{#2}{}{
    \DeclareOverArrowCommand[#1]{#2}{#3}
  }
}
```

`\NewOverArrowCommand`

```
\NewDocumentCommand{\NewOverArrowCommand}{
 O{symb} >{\ovar@cmdname@processor} m m
}{%
  \ifcsdef{#2}{%
    \PackageError{overarrows}{Command \csname #2\endcsname already defined}%
    {You have used \protect\NewOverArrowCommand\space with a command that
      already has a definition. \MessageBreak%
      Choose another name, or use instead \protect\DeclareOverArrowCommand.}
  }{%
    \DeclareOverArrowCommand[#1]{#2}{#3}
  }
}
```

`\RenewOverArrowCommand`

```
\NewDocumentCommand{\RenewOverArrowCommand}{
 O{symb} >{\ovar@cmdname@processor} m m
}{%
  \ifcsundef{#2}{%
    \PackageError{overarrows}{Command  \csname #2\endcsname undefined}%
    {You have used \protect\RenewOverArrowCommand\space with a command that was
      never defined. \MessageBreak%
      Check the requested name, or use instead \protect\NewOverArrowCommand.}
  }{%
    \DeclareOverArrowCommand[#1]{#2}{#3}
  }
}
```

Starred variant

\ovar@starversion

#1: definition (stack macro + arrow macro)
#2: math style
#3: content

```
\newsavebox\ovar@tempbox
\def\ovar@starversion#1#2#3{%
  \sbox{\ovar@tempbox}{$\m@th #1#2{#3}$}%
  \usebox{\ovar@tempbox}%
```

Remove the extra space added by the arrow.

```
  \settowidth{\ovar@tempdim}{$\m@th #2{#3}$}%
  \kern\dimeval{0.5\ovar@tempdim - 0.5\wd\ovar@tempbox}%
}
```

\vv vector command

\vv

\esvectvv

Backup and redefinition of esvect \vv→P. 20 vector command.

```
\ifdefined\ovar@option@esvect
  \let\esvectvv\vv
  \undef\vv
```

```
  \NewOverArrowCommand{\vv}{esvect, middle config = relbareda}
\fi
```

Predefined commands

Declare predefined commands after unicode-math settings.

```
\AddToHook{begindocument}[overarrows]
  {
```

\overrightarrow

```
    \ifovar@option@overrightarrow@
      \DeclareOverArrowCommand{\overrightarrow}{%
        amsmath, middle config=relbar,
        end=\ovar@rightarrow,
        right arrow,
      }
    \fi
```

\underrightarrow

```
    \ifovar@option@underrightarrow@
      \DeclareOverArrowCommand{\underrightarrow}{%
        amsmath, middle config=relbar,
        end=\ovar@rightarrow,
        right arrow,
        arrow under,
      }
    \fi
```

\overleftarrow

```
    \ifovar@option@overleftarrow@
      \DeclareOverArrowCommand{\overleftarrow}{%
        amsmath, middle config=relbar,
        start=\ovar@leftarrow,
        end=\relbar,
        left arrow,
      }
    \fi
```

\underleftarrow

```
    \ifovar@option@underleftarrow@
      \DeclareOverArrowCommand{\underleftarrow}{%
        amsmath, middle config=relbar,
        start=\ovar@leftarrow,
        end=\relbar,
        left arrow,
        arrow under,
      }
    \fi
```

\overleftrightarrow

```
    \ifovar@option@overleftrightarrow@
      \DeclareOverArrowCommand{\overleftrightarrow}{%
        amsmath, middle config=relbar,
        start=\ovar@leftarrow,
        end=\ovar@rightarrow,
        center arrow,
      }
    \fi
```

\underleftrightarrow

```
    \ifovar@option@underleftrightarrow@
      \DeclareOverArrowCommand{\underleftrightarrow}{%
        amsmath, middle config=relbar,
        start=\ovar@leftarrow,
        end=\ovar@rightarrow,
        center arrow,
        arrow under,
      }
    \fi
```

\overrightharpoonup

```
    \ifovar@option@overrightharpoonup@
      \DeclareOverArrowCommand{\overrightharpoonup}{%
        amsmath, middle config=relbar,
        end=\rightharpoonup,
        right arrow,
      }
    \fi
```

\underrightharpoonup

```
    \ifovar@option@underrightharpoonup@
      \DeclareOverArrowCommand{\underrightharpoonup}{%
        amsmath, middle config=relbar,
        end=\rightharpoonup,
        right arrow,
        arrow under,
      }
    \fi
```

\overrightharpoondown

```
    \ifovar@option@overrightharpoondown@
      \DeclareOverArrowCommand{\overrightharpoondown}{%
        amsmath, middle config=relbar,
        end=\rightharpoondown,
        right arrow,
      }
    \fi
```

\underrightharpoondown

```
    \ifovar@option@underrightharpoondown@
    \DeclareOverArrowCommand{\underrightharpoondown}{%
      amsmath, middle config=relbar,
      end=\rightharpoondown,
      right arrow,
      arrow under,
    }
  \fi
```

\overleftharpoonup

```
  \ifovar@option@overleftharpoonup@
    \DeclareOverArrowCommand{\overleftharpoonup}{%
      amsmath, middle config=relbar,
      start=\leftharpoonup,
      end=\relbar,
      left arrow,
    }
  \fi
```

\underleftharpoonup

```
  \ifovar@option@underleftharpoonup@
    \DeclareOverArrowCommand{\underleftharpoonup}{%
      amsmath, middle config=relbar,
      start=\leftharpoonup,
      end=\relbar,
      left arrow,
      arrow under,
    }
  \fi
```

\overleftharpoondown

```
  \ifovar@option@overleftharpoondown@
    \DeclareOverArrowCommand{\overleftharpoondown}{%
      amsmath, middle config=relbar,
      start=\leftharpoondown,
      end=\relbar,
      left arrow,
    }
  \fi
```

\underleftharpoondown

```
  \ifovar@option@underleftharpoondown@
    \DeclareOverArrowCommand{\underleftharpoondown}{%
      amsmath, middle config=relbar,
      start=\leftharpoondown,
      end=\relbar,
      left arrow,
      arrow under,
    }
  \fi
```

\overbar

```
  \ifovar@option@overbar@
    \DeclareOverArrowCommand{\overbar}{%
      amsmath, middle config=relbar,
      start={\std@minus}, end={\std@minus},% \relbar is defined with \mathsm@sh
      shift leftright=0,
      space after arrow=-0.3ex,
    }
  \fi
```

With unicode-math, add `\vphantom{+}` to get the correct position.

\underbar

```
  \ifovar@option@underbar@
    \DeclareOverArrowCommand{\underbar}{%
      amsmath, middle config=relbar,
      start={\vphantom{+}\std@minus}, end={\std@minus},% \relbar is defined with \mathsm@sh
      shift leftright=0,
      arrow under,
      space before arrow=-0.3ex,
    }
  \fi
```

End of `\AddToHook{begindocument}` hook.

```
  }
```

Test macros

`\ovar@testmathstyles`

Tabular containing the output of a command for the four math styles and different patterns.

```
\newcommand{\ovar@testmathstyles}[2][]{
  \begingroup
  \newcommand*{\ovar@row@teststyle}[1]{%
    $\displaystyle ##1$
    & $\textstyle ##1$
    & $\scriptstyle ##1$
    & $\scriptscriptstyle ##1$
    \\
  }
  \renewcommand*{\arraystretch}{1.5}
  \begin{tabular*}{0.95\linewidth}{@{\extracolsep{\fill}} cccc}
    \hline
    \footnotesize\texttt{\texttt{\textbackslash displaystyle}}
    & \footnotesize\texttt{\texttt{\textbackslash textstyle}}
    & \footnotesize\texttt{\texttt{\textbackslash scriptstyle}}
    & \footnotesize\texttt{\texttt{\textbackslash scriptscriptstyle}}
    \\
    \hline
    \ovar@row@teststyle{\csuse{#2}{v}}
    \ovar@row@teststyle{\csuse{#2}{AB}}
    \ovar@row@teststyle{\csuse{#2}{\mathrm{grad}}}
    \ovar@row@teststyle{\csuse{#2}{my~long~vector}}
    \IfValueT{#1}{\ovar@row@teststyle{\csuse{#2}{#1}}}
    \hline
  \end{tabular*}
  \endgroup
}
```

`\ovar@testkerning`

```
\begingroup
\ifovar@option@subother@  \catcode `\_=12 \fi
\ifovar@option@subactive@ \catcode `\_=13 \fi
\gdef\ovar@testkerning#1{%
  \begin{displaymath}
    #1{t}_{#1{u}_{#1{v}}}
    \qquad
    #1{\imath}_0
    \qquad
    #1{v}
    = #1{v}_x + #1{v}_y + #1{v}_z
    = v_x #1{\imath} + v_y #1{\jmath} + v_z #1{k}
  \end{displaymath}
}
\endgroup
```

`\TestOverArrow`

```
\NewDocumentCommand{\TestOverArrow}{
  s o >{\ovar@cmdname@processor} m
}{%
  \ifcsdef{#3}{}{%
    \PackageWarning{overarrows}{Unknown name '#3' passed to
      \protect\TestOverArrow}
  }
  \IfBooleanTF{#1}{%
    \noindent\framebox{%
      \begin{minipage}{0.95\linewidth}
```

```
        \centering
        \noindent\textbf{\large%
          Test of \texttt{\textbackslash#3} and \texttt{\textbackslash#3*} macros}
        \bigskip\par
        \textbf{\texttt{\textbackslash#3} for different math styles}
        \smallskip\par
        \ovar@testmathstyles[#2]{#3}%
        \bigskip\par
        \textbf{\texttt{\textbackslash#3} kerning}
        \ovar@testkerning{\csuse{#3}}
        \textbf{\texttt{\textbackslash#3*} kerning}
        \ovar@testkerning{\csuse{#3}*}
      \end{minipage}%
    }\bigskip\par
  }{%
    \ovar@testmathstyles[#2]{#3}%
  }
}
```

Index

Entries listed in the categories “commands”, “lengths”, and “internal macros” also include references to package implementation.

Change History

v1.0

General: Initial version 1

v1.0.1

General: Bug fix for under* options 34

v1.1

General: Support for non-standard subscripts 36, 47, 51

v1.2

General: Add option pstricks 34

Add the method pstricks 41

Allow backslash in command name 46

Declare predefined commands after unicode-math settings . . . 48

Fix esvect font sizes 35

Get the correct rule thickness with unicode-math 44

Remove useless `\AtBeginDocument` 36

Use `\def` instead of `\let` for `\ovar@rightarrow` and `\ovar@leftarrow` 35

Use `\harrowextender`, if availlable. 36

Use boxes in starred variant for better performances 48

v1.3

General: Process options in the order specified by the user . . . 35

Replace esvect conditionnal by the definition of a control sequence 48

Use a control sequence to store the esvect option 34

Use only one `\PassOptionsToPackage` with esvect 35

v1.4

General: Make commmands robust 47