

# The `amsbsy` package

Frank Mittelbach      Rainer Schöpf      Michael Downes

Version v1.2d, 1999/11/29

This file is maintained by the L<sup>A</sup>T<sub>E</sub>X Project team.  
Bug reports can be opened (category `amslatex`) at  
<https://latex-project.org/bugs/>.

## 1 Introduction

The package `amsbsy`, first written in 1989, implements a few commands for producing **bold** characters in the ‘normal’ *math version*.

*Note: It is recommended nowadays to use the `bm` package, which became available in 1997.*

If we have bold fonts which contain the character in question then we will use these fonts to produce the wanted character. But sometimes math fonts are only available in a certain weight (e.g. the AMS symbol fonts). For these cases we provide a command which is called `\pmb` (an acronym for **p**oor **m**an’s **b**old) with one argument. The bolder weight is achieved by copying the argument three times in slightly different positions which is better than nothing but no match for a real bold font.

There also exists the `\boldsymbol` command which is better in all cases where bold fonts exists. This command will internally switch to the corresponding ‘bold’ *math version* typeset its argument in this version.

Both commands will preserve the nature of their arguments, i.e. if they get a relational atom their result will again be a relation as far as T<sub>E</sub>X’s mathspacing is concerned.

Since it is good policy to make at least a small test we try to typeset the infinity sign ( $\infty$ ) first with `\pmb` and then with `\boldsymbol`.

$$\infty = \infty \quad ?$$

Standard package info.

```
\NeedsTeXFormat{LaTeX2e}% LaTeX 2.09 can't be used (nor non-LaTeX)
[1994/12/01]% LaTeX date must December 1994 or later
\ProvidesPackage{amsbsy}[1999/11/29 v1.2d Bold Symbols]
```

## 2 The implementation

We need some functions from the `amsgen` package.

```
\RequirePackage{amsgen}
```

`\boldsymbol` In implementing `\boldsymbol`, we must take into account  $\TeX$ 's limitation of only 16 mathgroups (math families, in Knuth's terminology). If we wanted to maintain mathgroups for both the bold and non-bold version of each math font, it would not take long to run out of mathgroups. Therefore what we do instead for a bold symbol is embed it in an `\hbox`; inside that `\hbox`, when we start another math formula, we can change all the mathgroups to their bold equivalents.

However, to get the correct math style inside the `\hbox` (display, text, script or scriptscript) we have to use `\mathchoice`. Since `\mathversion{bold}` has a lot of overhead, and `\mathchoice` typesets the argument text four times, we would rather not put the `\mathversion` command inside each `\hbox` in the `\mathchoice`; on the other hand, `\mathversion` gives an error message if it's used in math mode. Therefore if we want to execute `\mathversion{bold}` before starting the `\mathchoice` we have to temporarily disable the `\@nomath` error. (The error message is intended to keep people from accidentally emboldening a preceding part of a math formula, since only the mathgroups defined at the end of a math formula will determine the fonts used in that formula; but we are going to typeset our bold symbol not in the current formula but in an embedded formula, so that this danger doesn't apply here.)

```
\DeclareRobustCommand{\boldsymbol}[1]{%
```

Start a group to localize the change of `\@nomath`:

```
\begingroup
```

Disable `\@nomath` so that we don't have to leave math mode before executing `\mathversion`:

```
\let\@nomath\@gobble \mathversion{bold}%
```

`\math@atom` is a test macro which looks at its argument and produces a math atom of the proper class.

```
\math@atom{#1}{%
```

Although it is tempting to use `\text` here, to save some main memory, that caused a bug in the past due to some internal interactions with `\mathversion`.

```
\mathchoice%
  {\hbox{$\m@th\displaystyle#1$}}%
  {\hbox{$\m@th\textstyle#1$}}%
  {\hbox{$\m@th\scriptstyle#1$}}%
  {\hbox{$\m@th\scriptscriptstyle#1$}}}%
```

End the group we started earlier.

```
\endgroup}
```

`\math@atom` The macro `\math@atom` looks at its argument and produce a correct math atom, i.e. a primitive like `\mathopen`. Until the day we have a real implementation for

all cases we use the `\binrel@` command from  $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{T}\mathcal{E}\mathcal{X}$  which can distinguish between binary, relation and ord atoms.

```
\def\math@atom#1#2{%
  \binrel@{#1}\binrel@@{#2}}
```

`\pmb` Poor man's bold command, works by typesetting multiple copies of the given argument with small offsets.

```
\DeclareRobustCommand{\pmb}{%
  \ifmmode\else \expandafter\pmb@@\fi\mathpalette\pmb@}
```

`\pmb@@` is called by `\pmb` in the non-math-mode case. Discard the first two arguments which are for the math-mode case.

```
\def\pmb@@#1#2#3{\leavevmode\setboxz@h{#3}%
  \dimen@-\wdz@
  \kern-.5\ex@ \copy\z@
  \kern\dimen@ \kern.25\ex@ \raise.4\ex@ \copy\z@
  \kern\dimen@ \kern.25\ex@ \box\z@
}
```

```
\newdimen\pmbraise@
```

Note: because of the use of `\mathpalette`, if `\pmb@` is applied to a single math italic character (or a single character from some other slanted math font), the italic correction will be added. This will cause subscripts to fall too far away from the character in some cases, e.g.,  $T_1$  or  $\mathcal{T}_1$ .

```
\def\pmb@#1#2{\setbox8\hbox{\math@th#1{#2}}%
  \setboxz@h{\math@th#1\mkern.5mu}\pmbraise@\wdz@
  \binrel@{#2}%
  \dimen@-\wd8 %
  \binrel@@{%
    \mkern-.8mu\copy8 %
    \kern\dimen@\mkern.4mu\raise\pmbraise@\copy8 %
    \kern\dimen@\mkern.4mu\box8 }%
}
```

```
\def\binrel@#1{\begingroup
  \setboxz@h{\thinmuskip0mu
    \medmuskip\m@ne mu\thickmuskip\@ne mu
    \setbox\tw@\hbox{#1\math@th}\kern-\wd\tw@
    #1\math@th}%
}
```

The `\noexpand` here should be unnecessary, but just in case ...

```
\edef\@tempa{\endgroup\let\noexpand\binrel@@
  \ifdim\wdz@<\z@ \mathbin
  \else\ifdim\wdz@>\z@ \mathrel
  \else \relax\fi\fi}%
\@tempa
}
```

For completeness, assign a default value for `\binrel@@`.

```
\let\binrel@@\relax
```

The usual `\endinput` to ensure that random garbage at the end of the file doesn't get copied by `docstrip`.

`\endinput`