The **bm** package[*][†]

David Carlisle with support by Frank Mittelbach

2019/07/24

1   Introduction

This package defines commands to access bold math symbols. The basic command is `\bm` which may be used to make the math expression in its argument be typeset using bold fonts.

The syntax of `\bm` is:

`\bm{`⟨*math expression*⟩`}`

So `$\alpha \not= \bm{\alpha}$` produces $\alpha \neq \boldsymbol{\alpha}$.

`\bm` goes to some trouble to preserve the spacing, so that for instance `\bm<` is a bold $<$ but with the correct `\mathrel` spacing that TeX gives to $<$. The calculations that TeX needs to do for `\bm` can be quite involved and so a definition form is provided.

`\DeclareBoldMathCommand[`⟨*math version*⟩`]{`⟨*cmd*⟩`}{`⟨*math expression*⟩`}`

Defines `\cmd` to be the bold form of the math expression. The ⟨*math version*⟩ defaults to 'bold' (i.e., `\boldmath`).

For relatively simple expressions, the resulting definitions are very efficient, for instance after:

`\DeclareBoldMathCommand\balpha{\alpha}`

`\balpha` is a single 'mathchardef' token producing a bold alpha, and so is just as fast to execute as `\alpha`.

The above command is mainly intended for use in packages. For occasional use in LaTeX documents, and for compatibility with the plain TeX support for the mathtime fonts, a 'user-level' version, `\bmdefine` is provided that is equivalent to:

`\DeclareBoldMathCommand[bold]`.

If there is a 'heavy' math version defined (usually accessed by a user-command `\heavymath`) then a similar command `\hm` is defined which access these 'ultra bold' fonts. Currently this is probably only useful with the 'mathtime plus' font collection. Definitions of commands that use these fonts may be made by specifying the optional argument 'heavy' to `\DeclareBoldMathCommand`. Again an abbreviation,

---

[*]This file has version number v1.2d, last revised 2019/07/24.

[†]Development of this package was commissioned by Y&Y.

`\hmdefine`, is provided, equivalent to:
`\DeclareBoldMathCommand[heavy]`.

The command names (but not the implementation) are taken from Michael Spivak's macros to support the mathtime fonts for plain TeX. In those original macros, the syntax for `\bmdefine` was `\bmdefine\balpha{\bm\alpha}` (with a nested `\bm`). This syntax also works with this package.

2   Font allocation

In order to access bold fonts in the simplest and quickest possible manner, the package normally allocates symbol fonts for bold (and possibly heavy) fonts into the 'normal' math version. By default it allocates at most four fonts for `\bm` and at most three fonts for `\hm`. This means that if the mathtime plus font set is being used, seven additional symbol fonts will be used, in addition to the basic four that LaTeX already declares. The mathtime package also declares an extra symbol font, bringing the total to twelve. The maximum number of symbol *and* math alphabet fonts that can be used in a math version is sixteen. So the above allocation scheme does not leave room for many extra math symbols (such as the AMS symbols) or math alphabets (such as `\mathit`).

Before loading the bm package you may define `\bmmax` and `\hmmax` to be suitable values, for instance you may want to set `\newcommand\hmmax{0}` if you will not be using `\hm` much, but you do have a heavy math version defined.

Even if `\bmmax` is set to zero, `\bm` will still access the correct bold fonts (by accessing the fonts via `\boldmath`) but this method is slower, and does not work with delimiters. Delimiters can only be made bold if the bold font has been allocated.

Conversely if you have a non standard font set that makes available extra math delimiters and accents in bold and medium weights you may want to *increase* `\bmmax` so that fonts are allocated for your font set.

3   Features

In most cases this package should work in a fairly self explanatory way, but there are some things that might not be obvious.

3.1   Interaction with Math Alphabet Commands

As mentioned above, `\bm` goes to some trouble to try to make a command that is just like its argument, but using a bold font. This does not always produce the effect that you might expect.

```
$1 g \bm{g}$
$2 \mathrm{g \bm{g}}$
$3 {g} \bm{{g}}$
$4 \mathrm{{g} \bm{{g}}}$
$5 \mathrm{g} \bm{\mathrm{g}}$
```

produces the following:

1gg 2gg 3gg 4gg 5g**g**

In math mode 'g' is effectively a command that produces the letter 'g' from the 'letters' alphabet, unless a Math Alphabet command is in effect, in which case the 'g' comes from the specified alphabet. `\bm{g}` makes an equivalent command, but which defaults to a bold letter alphabet. So in the first example `\bm{g}` is bold math italic, but in the second example the `\mathrm` applies to both `g` and `\bm{g}` in the same way, and so they are both roman.

`\bm` only inspects the 'top level' definition of a command, for more complicated expressions, and anything inside a `{ }` group, `\bm` forces bold fonts by essentially the same (slow) technique used by the AMS `\boldsymbol` command (but `\bm` still takes more care of the spacing). So the third example produces identical output to the first (but TeX takes more time producing it).

In the fourth example the `\mathrm{\bm{g}}` is essentially equivalent to `\mathrm{\mbox{\boldmath$g$}}`. Currently math alphabet settings are not passed down to 'nested' math lists, and so in this example, the `\mathrm` has no effect, and a bold math italic g is obtained.

Similarly the last example is equivalent to `$\mbox{\boldmath$\mathrm{g}$}}` and so in this case, one obtains a bold roman $\mathbf{g}$.

3.2 Delimiters

TeX can treat character tokens in two[1] ways. If there is a preceding `\left` or `\right` it can treat them as a delimiter, otherwise it can treat them as a standard character. For example `\left<\right>` produces $\langle\rangle$, which is totally different from `<>`, which produces $<>$.

TeX can only do this for character tokens. Commands such as `\langle` do not act in this way. This means that `\bm` has to decide whether to treat a character as a delimiter or not. The rule it uses is, it makes a delimiter command for a character if the previous token in the argument was `\left` or `\right`. So `\left\bm{<}` does not work, but `\bm{\left<}` does.

3.3 Command Arguments

Normally if a command takes arguments the full command, including any arguments, should be included in `\bm`.

So `\bm{\overbrace{abc}}` (producing $\overbrace{\boldsymbol{abc}}$) not `\bm{\overbrace}{abc}`. If you do not include all the arguments you will typically get the error message:
`Runaway argument?`
`! Forbidden control sequence found while scanning use of ...`

However commands defined in terms of the TeX accent and radical primitives *may* be used without their arguments. So `\bm{\hat}{a}` produces $\boldsymbol{\hat{}}a$, a bold accent over a non-bold a (compare $\hat{a}$) whereas `\bm{\hat{a}}` makes both the a and the accent bold, $\boldsymbol{\hat{a}}$. Similarly, although the LaTeX command `\sqrt` must be used with its arguments, `\sqrtsign` may be used as in `\bm\sqrtsign{abc}` to produce $\boldsymbol{\sqrt{}}abc$ rather than $\sqrt{abc}$ or $\boldsymbol{\sqrt{abc}}$

If you really need to make a command with arguments use bold fonts without making all of the arguments bold, you can explicitly reset the math version in the argument, eg:

`\sqrt{xyz}`	`\bm{\sqrt{xyz}}`	`\bm{\sqrt{\mbox{\unboldmath$xyz$}}}`
$\sqrt{xyz}$	$\boldsymbol{\sqrt{xyz}}$	$\sqrt{xyz}$

---

[1] Well more than two really.

3.4 Bold fonts

This package interrogates the font allocations of the bold and heavy math versions, to determine which bold fonts are available. This means that it is best to load the package *after* any packages that define new symbol fonts, or (like the `mathtime` package) completely change the symbol font allocations.

If no bold font appears to be available for a particular symbol, `\bm` will use 'poor man's bold' that is, overprinting the same character in slightly offset positions to give an appearance of boldness.

In the standard Computer Modern font set, there is no bold 'large symbols' font. In the 'mathptm' and (standard) mathtime font sets there are no bold math fonts. In the 'mathtime plus' font set there are suitable fonts for bold and heavy math setting, and so `\bm` and `\hm` work well. Similarly in the basic Lucida New Math font set there are no bold math fonts, so `\bm` will use 'poor man's bold. However if the Lucida Expert set is used, Then `\bm` will detect, and use the bold math fonts that are available.

As discussed above, one may set `\bmmax` higher or lower than its default value of four to control the font allocation system. Finer control may be gained by explicitly declaring bold symbol fonts. Suppose you have a symbol font 'xyz' that is available in medium and bold weights, then you would declare this to LaTeX via:
`\DeclareSymbolFont{extras}  {OMS}{xyz}{m}{n}`
`\SetSymbolFont{extras}{bold}{OMS}{xyz}{bx}{n}`
At this point the symbols will be available in the normal math version, and their bold variants in `\boldmath`. If you also declare:
`\DeclareSymbolFont{boldextras}{OMS}{xyz}{bx}{n}`
That is, declare a symbol font whose name is formed by prefixing 'bold' (or 'heavy') to an existing symbol font, then `\bm` (or `\hm`) will use this font directly, rather then accessing the 'extras' symbol font via `\boldmath`.

3.5 Strange failures

In order to get the correct spacing, `\bm` has to 'investigate' the definition of the commands in its argument. It is possible that some strange constructions could 'confuse' this investigation. If this happens then LaTeX will almost certainly stop with a strange error. This should not happen with any of the math symbols defined in the base LaTeX or AMS distributions, or any commands defined in terms of those symbols using normal LaTeX math constructs. However if some command does fail to work inside `\bm` you should always be able to surround it with an extra set of braces `\bm{{\cmd}}` rather than `\bm{\cmd}`. `\bm` will not then attempt to set the correct spacing, so you may need to set it explicitly, for instance, for a relation, `\bm{\mathrel{\cmd}}`.

3.6 AMS package **amsbsy**

The `\bm` command shares some functionality with the `\boldsymbol` command from the AMS LaTeX collection. To aid in moving documents between these two packages, this package defines `\boldsymbol` and `\heavysymbol` as alternative names for `\bm` and `\hm`.

4   Implementation

The commands \bm and \hm work by defining a number of additional symbol fonts corresponding to the standard ones 'operators', 'letters', 'symbols', and 'largesymbols'. The names for these symbols fonts are produced by prefixing the usual name with 'bold' or 'heavy'.

For maximum flexibility we get the font definitions by looking in the corresponding math versions, i.e., into \mv@bold and if defined into \mv@heavy.

> 1 ⟨*package⟩

\bm@table
\bm@boldtable
\bm@heavytable

The table, \bm@table, (which is locally \let to either the bold or heavy version) defines, for each ⟨math group⟩ (⟨fam⟩), the 'offset' to the bold version of the specified symbol font. If there is no bold symbol font defined, the offset will be set to zero if there is a bold font assigned to this slot in the bold math version, or −1 if the font in the bold math version is the same as the one in the normal math version. In this case a 'poor man's bold' system of overprinting is used to achieve boldness where this is possible.

The settings are made at the time this package is read, and so it is best to load this package late, after any font loading packages have been loaded. Symbol fonts loaded after this package will get the offset of zero, so they will still be made bold by \bm as long as an appropriate font is declared for the bold math version.

\bm@boldtable and \bm@heavytable are set up using very similar code, which is temporarily defined to \bm, to save wasting a csname. Similarly \bm@pmb... (which will be defined later) are used as scratch macros.

The general plan. Run through the fonts allocated to the normal math version. Ignore ⟨math alphabet⟩ allocations[2] but for each math symbol font, look in the math version specified by #1 (bold or heavy). If the font there is different, then allocate a new symbol font in the normal math version to access that bold font and place the numerical difference between the allocations of the bold and normal font into the table being built (\bm@boldtable, if #1 is bold). If the symbol allocation is already greater than \bmmax do not allocate a new symbol font, but rather set the offset in the table to zero. \bm will detect this, and use \boldmath on its argument in this case, so the bold font will be accessed but more slowly than using a direct access to a bold font allocated into the normal math version. If the font allocated in the bold math version is the same as the font in the normal math version, set the offset to −1, which is a flag value that causes \bm to use 'poor man's bold' overprinting three copies of the symbol, offset slightly to give an appearance of boldness.

Fonts containing delimiters and math accents *must* be allocated into the normal math version if they are to be used with \bm. (In these cases \bm will produce the normal weight symbol, rather than using \boldmath or poor man's bold.)

> 2 \def\bm#1#2{%

This code can not work inside a group, as that would affect any symbol font allocations, so instead use some scratch macros to save and restore the definitions of commands we need to change locally.

> 3   \let\bm@pmb\install@mathalphabet
> 4   \let\bm@pmb@\getanddefine@fonts
> 5   \let\bm@pmb@@\or

---

[2]For now?

```
6    \edef\bm@general{\f@encoding/\f@family/\f@series/\f@shape/\f@size}%
```

#2 specifies the maximum number of fonts to allocate (either `\bmmax` or `\hmmax`). First check against `\count18` that there are that many slots left, and if not reduce accordingly. Put the resulting value in `\@tempcnta`.

```
7    \@tempcnta#2%
8    \count@-\count18%
9    \advance\count@-\@tempcnta
10   \advance\count@15\relax
11   \ifnum\count@<\z@
12      \advance\@tempcnta\count@
13   \fi
```

Make `\or` non-expandable, so we can build an `\ifcase` bit-by-bit in a sequence of `\edef`s.

```
14   \let\or\relax
```

Initialise the table (to `\@gobble` to remove the first `\or`).

```
15   \expandafter\let\csname bm@#1table\endcsname\@gobble
```

Helper macro that adds the next entry to the table being built.

```
16   \def\bm@define##1{%
17      \expandafter\xdef\csname bm@#1table\endcsname{%
18         \csname bm@#1table\endcsname\or##1}}%
```

Each symbol font is recorded in the math version list by a sequence such as:
`\getanddefine@fonts \symsymbols \OMS/cmsy/m/n`
Where the first argument is a chardef token carrying the number allocated (to symbols, in this example), and the second argument is a csname whose *name* denotes the font used. So locally redefine `\getanddefine@fonts` to compare #2 with the name in the appropriate slot in the bold math version.

```
19   \def\getanddefine@fonts##1##2{%
20      \def\@tempa{##2}%
21      \def\@tempb####1##1####2####3\@nil{\def\@tempb{####2}}%
22      \expandafter\expandafter\expandafter
23         \@tempb\csname mv@#1\endcsname\@nil
```

Now `\@tempa` and `\@tempb` contain the names of the fonts allocated to this slot in the two math versions.

```
24   \ifx\@tempa\@tempb
```

If they are the same, set this offset to −1, as a flag to use poor man's bold.

```
25      \bm@define\m@ne
26   \else
```

Else make a new name by adjoining #1 to the name of the symbol font eg, `\symboldsymbols` to match `\symsymbols`. If that font has already been allocated, or if `\@tempcnta` is positive so we can allocate a new slot for this font, then the table will be set with the offset between the two fonts. otherwise set the offset to zero (so `\boldmath` will be used to access the font).

```
27      \edef\@tempa{sym#1\expandafter\@gobblefour\string##1}%
28      \ifnum\@tempcnta<%
29         \expandafter\ifx\csname\@tempa\endcsname\relax
30            \@ne
31         \else
```

```
32            \m@ne
33          \fi
34        \bm@define\z@
35      \else
```

If the font is not yet allocated, allocate it now, using an internal hack into
\DeclareMathSymbolFont.

However before allocating it look in the bold math version to see if it is the
same, and if so use that. For example with Mathtime the 'operators' font in the
'heavy' math version is different from that in 'normal', but it is the same as the
font in 'bold' (Times bold). So rather than allocate \symheavyoperators just set
it equal to \symboldoperators.

```
36        \expandafter\ifx\csname\@tempa\endcsname\relax
37          \begingroup
38          \escapechar\m@ne
39          \edef\@tempb{\endgroup
40            \noexpand\split@name
41            \expandafter\string\@tempb}%
42          \@tempb/\@nil
43         \expandafter\ifx
44           \csname symbold\expandafter\@gobblefour\string##1\endcsname
45           \relax
```

If no font has been allocated for \bm yet, then allocate it now.

```
46            \expandafter\new@mathgroup\csname\@tempa\endcsname
47            \expandafter\new@symbolfont\csname\@tempa\endcsname
48                    \f@encoding\f@family\f@series\f@shape
```

Reduce by one the number of fonts we can still allocate.

```
49            \advance\@tempcnta\m@ne
50          \else
```

Else do a similar look into the bold mathgroup. Use \bm@expand as a scratch
macro to save on string space.

```
51          \def\bm@expand####1##1####2####3\@nil{\def\bm@expand{####2}}%
52     \expandafter\expandafter\expandafter
53          \bm@expand\csname mv@bold\endcsname\@nil
54          \ifx\bm@expand\@tempb
```

If the font just found (in heavy) is the same as the font in bold use the slot (in
normal) previously allocated for the bold font. (That clear?)

```
55            \expandafter\let\csname\@tempa\expandafter\endcsname
56              \csname symbold\expandafter
57                    \@gobblefour\string##1\endcsname
58          \else
```

Otherwise allocate a new slot for it.

```
59            \expandafter\new@mathgroup\csname\@tempa\endcsname
60            \expandafter\new@symbolfont\csname\@tempa\endcsname
61                    \f@encoding\f@family\f@series\f@shape
62            \advance\@tempcnta\m@ne
63          \fi
64        \fi
```

```
65          \else
```

If the font has been allocated already, use the existing allocation.

```
66              \PackageInfo{bm}%
67                {Symbol font \@tempa\space already defined.\MessageBreak
68                 Not overwriting it}%
69          \fi
```

Whether the font has just been allocated, or whether it was previously allocated, compute the offset and add it to the table.

```
70          \count@\csname\@tempa\endcsname
71          \advance\count@-##1%
72          \bm@define{\the\count@\relax}%
73        \fi
74     \fi}%
```

The math version list also contains information about math alphabet commands, but we want to ignore those here, so ...

```
75     \let\install@mathalphabet\@gobbletwo
```

Having set up the local definitions, execute the list for the normal math version.

```
76     \mv@normal
```

So now the offsets are all entered into the table, separated by `\or`. Finish off the definition by making this an `\ifcase`. Add a default value of zero, so that any symbol fonts declared later will also work, as long as a bold version is assigned to the bold math version.

```
77     \expandafter\xdef\csname bm@#1table\endcsname{%
78       \noexpand\ifcase\@tempcnta
79         \csname bm@#1table\endcsname
80       \noexpand\else
81         \z@
82       \noexpand\fi}%
```

Put things back as they were.

```
83     \expandafter\split@name\bm@general\@nil
84     \let\install@mathalphabet\bm@pmb
85     \let\getanddefine@fonts\bm@pmb@
86     \let\or\bm@pmb@@}
```

\bmmax   To save declaring too many symbol fonts, do not auto-declare any more than \bmmax bold symbol fonts into the normal math version. Any bold fonts not so allocated will be accessed via \boldmath which is slower and doesn't work for delimiters and accents. It may be set in the preamble with \newcommand but use \chardef here for a slight efficiency gain.

If this is set to a higher value before this package is loaded, keep that value.

```
87 \ifx\bmmax\@undefined
88   \chardef\bmmax=4
89 \fi
```

If there is no bold math version, It is very easy to set up the table, no need to use all the tricky code above. Also, at the end of the package redefine the internal macro that \bm uses to call \boldmath, to use poor man's bold instead.

```
90 \ifx\mv@bold\@undefined
91   \def\bm@boldtable{\m@ne}
```

```
92  \AtEndOfPackage{%
93    \def\bm@gr@up#1#2{%
94      \bm@pmb{#2}}}
95 \else
```

Otherwise use the definition of \bm above to set up \bm@boldtable by comparing the fonts available in the normal and bold math versions.

```
96  \bm{bold}\bmmax
```

\mathbf   As the bold font has been defined as a symbol font, make \mathbf access that rather than have it allocate a new math group for the same font. (Just in case there were no free slots wrap this in an extra test.)

```
97  \@ifundefined{symboldoperators}
98    {}
99    {\DeclareSymbolFontAlphabet\mathbf{boldoperators}}

100 \fi
```

\hmmax   Same for heavy (but default to three this time (enough for mathtime plus, as no heavy operators font).

```
101 \ifx\hmmax\@undefined
102   \chardef\hmmax=3
103 \fi
```

Similarly if there is a heavy math version, set up \bm@heavytable. (If there is no heavy math version, do nothing here, as \hm will be set to \bm later, once that is defined.)

```
104 \ifx\mv@heavy\@undefined
105 \else
106   \bm{heavy}\hmmax
107 \fi
```

\bm@general   \bm is pretty much \bmdefine\bm@command followed by executing \bm@command. It would in principle be possible to execute the emboldened tokens directly, rather than building up a macro first, but (as I learned the hard way) it's difficult to do this in the midst of all these nested \if constructs. First extract the central bit of code for \hm \bm \hmdefine and \bmdefine. Note that in the case of the inline versions they take an argument and brace it, rather than relying on \bm@general to pick up the argument. This makes the code robust with respect to premature expansion.

```
108 \begingroup
109 \catcode`\'=\active
110 \catcode`\_=\active
111 \@firstofone{\endgroup
112 \def\bm@general#1#2#3#4#5{%
113   \begingroup
```

First locally disable \bm and \hm, as they would mess things up terribly, and the original Spivak versions used the syntax \bmdefine\balpha{\bm\alpha}.

```
114   \let\bm\@firstofone
115   \let\hm\@firstofone
```

Now initialise the commands used to save the tokens constructed.

```
116   \global\let\bm@command\@empty
117   \let\@let@token\@empty
```

As we want to expand the macros to look at their definition turn off protection. Otherwise the `\protect` will be carried over and apply to the wrong token, eg `{`.

```
118      \let\protect\@empty
119      \let\@typeset@protect\@empty
```

Set up either bold or heavy

```
120      \def\bm@mathchoice{\bm@m@thchoice#1}%
121      \def\bm@group{\bm@gr@up#1}%
122      \let\bm@table#2%
```

Make sure `\left` and `\right` are really non expandable, and not `\ifx` equal to anything else.

```
123      \let\left\holdinginserts
```

These three save on the number of `\ifx` tests below.

```
124      \let\right\left
125      \let\mskip\mkern
126      \let\hskip\kern
```

Definition of `'` locally modified so as not to use `\futurelet` in the look ahead, but to make the `\prime` available at the top level to be made bold, or heavy or whatever. `'` is locally active for this definition.

```
127      \let\bm@prime\copy
128      \let_\relax
129      \def'{\bm@prime\prime\relax}%
```

For optional argument commands. This expandable version of `\@ifnextchar` is not 100% safe, but works for `\sqrt` unless you put something really strange in the arguments.

```
130      \def\@ifnextchar##1##2##3##4{%
131        \if##1##4%
132          \expandafter\@firstoftwo
133        \else
134          \expandafter\@secondoftwo
135        \fi
136        {##2##4}{##3{##4}}}%
```

For Vladimir Volovich...

```
137      \def\GenericWarning##1##2{%
138        \unvcopy{\GenericWarning{##1}{##2}}}%
139      \def\GenericError##1##2##3##4{%
140        \unvcopy{\GenericError{##1}{##2}{##3}{##4}}}%
```

For AMS definitions.

```
141      \let\DN@\copy
142      \let\FN@\copy
143      \let\next@\copy
144      \global\let\bm@first\@empty
```

For AMS version of `\sqrt`: don't expand just wrap in brace group so that it can be made bold in a safe but slow way. Do the same for internal accent command

Code for AMS accent allows bm to be used (just) with accent but stops the nested accents stacking correctly, this can be corrected by using an extra brace group as usual. `\bm{{\hat{\hat{F}}}}`

```
145      \ifx\uproot@\undefined\else
146        \def\root##1\of##2{{\root##1\of{##2}}}%
```

```
147        \fi
148        \def\mathaccentV##1{\mathaccent"\accentclass@}%
```

For breqn definitions.

```
149        \let\@ifnext\@ifnextchar
150        \let\measure@lhs\copy
151        \let \rel@break\copy
152        \let \bin@break\copy
153        \let \after@open\copy
154        \let \after@close\copy
```

Make sure things like `\pounds` take the 'math branch' even in `\bmdefine` (which is not executed in math mode).

```
155        \let\ifmmode\iftrue
```

We have to ensure that the math alphabets have definitions that correspond the "bold" math version we are going to switch to. As these definitions are globally assigned when a math version is changed it is likely that right now we have those of the normal math version active. Argument `#3` holds either `\mv@bold` or `\mv@heavy` and we execute that after redefining `\install@mathalphabet` and `\getanddefine@fonts` suitably. The definitions are reverted back to their original the moment the scanning is done

```
156          \let\install@mathalphabet\def
157          \let\getanddefine@fonts\@gobbletwo
158          #3%
```

The last redefinition just makes `\mathit` type commands re-insert themselves (more or less) as if they are allowed to expand they die horribly if the expansions are put into `\mathchoice` and so executed more than once.

```
159        \def\select@group##1##2##3##4{{%
160          \protect##1{##4}}}%
161        \def\use@mathgroup##1##2##3{{%
162          \protect\use@mathgroup##1{##2}{##3}}}%
```

So now start looking at the argument.

```
163        \bm@expand#5\bm@end
164      \endgroup
```

Finally outside the group either execute `\bm@command` (for `\bm`) or save its definition (for `\bmdefine`).

```
165      #4}
```

End of the `\@firstofone` above, and the scope of the active '.

```
166  }
```

`\bm` Set up the bold (rather than heavy) version, and run `\bm@command` right at the end, to execute the emboldened argument. The argument is grabbed by the top level function, and explicitly braced, so that `\bm` works even if the braces are omitted round its argument in a 'moving argument'.

```
167  \DeclareRobustCommand\bm{%
168    \bm@general\boldmath\bm@boldtable\mv@bold\bm@command}
169  \protected@edef\bm#1{\bm{#1}}
```

`\DeclareBoldMathCommand` `DeclareBoldMathCommand[⟨mathversion⟩]{⟨command⟩}{⟨math expression⟩}`
`\bm@declare` looks like `\bm` except at the end the specified command is globally defined to be `\bm@command`. The ⟨mathversion⟩ defaults to 'bold'.

```
170 \def\DeclareBoldMathCommand{\@testopt\bm@declare{bold}}
```

```
171 \def\bm@declare[#1]#2{%
172   \expandafter\bm@general
173       \csname #1math\expandafter\endcsname
174       \csname bm@#1table\expandafter\endcsname
175       \csname mv@#1\endcsname
176       {\bm@define#2}}
```

\bmdefine   \bmdefine Shorthand for \DeclareBoldMathCommand[bold].

     \bm is empty within the definition, so that either
\bmdefine\balpha{\bm\alpha} or \bmdefine\balpha{\alpha}
may be used. (The former just for compatibility with the original version for plain
TEX).

```
177 \def\bmdefine{\DeclareBoldMathCommand[bold]}
```

\hm   Same again for \hm.

\hmdefine
```
178 \ifx\mv@heavy\@undefined
```

     If there is no heavy math version defined, let \hm be defined as \bm. Currently
there is no warning given, perhaps there should be, or even an error?

```
179   \let\hm\bm
180   \let\heavymath\boldmath
181   \let\bm@heavytable\bm@boldtable
```

```
182 \else
```

     Otherwise define \hm and \hmdefine in direct analogy with the above.

```
183   \DeclareRobustCommand\hm{%
184     \bm@general\heavymath\bm@heavytable\mv@heavy\bm@command}
```

```
185   \protected@edef\hm#1{\hm{#1}}
```

```
186   \def\hmdefine{\DeclareBoldMathCommand[heavy]}
```

```
187 \fi
```

\bm@end   Normally speaking \outer declarations should be avoided at all costs. (LATEX
redefines all of plain TEX's allocation macros to be non-outer.) However this is
one place where it seems like a good idea. If a command taking an argument is put
in \bm without its argument, then the \@@end terminating token would be taken
as the argument, and so the rest of the paragraph would be gobbled up and the
LATEX would die horribly. So make the internal terminating token \outer. (The
actual test for termination is made against \@@end not \bm@end as this macro will
be expanded by the look-ahead system.)

```
188 \outer\def\bm@end{\@@end}
```

\bm@expand   \afterassignment trick to fully expand the following tokens until the first non-
\bm@exp@nd   expandable token is revealed. This may discard a space token (which is what TEX
is looking for) but that doesn't matter in math mode. The expansion lookahead
is done twice in case any stray space tokens have crept in.[3]

```
189 \def\bm@expand{\afterassignment\bm@exp@nd\count@`\a}
```

---

[3]The need for this was noticed while testing \sqrt. The definition of \root inherited from
plain TEX has an anomalous space token, that is normally harmless (just wastes memory), but
which killed earlier versions of this package.

190 `\def\bm@exp@nd{\afterassignment\bm@test\count@`\a}`

\bm@test   Normally we will grab the non-expandable token as a macro argument but better
check it is not `{` first. Save the previous token so we can check later if it was `\left`,
in which case use the delcode rather than the mathcode if the current token is a
character.

191 `\def\bm@test{%`
192 `  \let\bm@previous\@let@token`
193 `  \futurelet\@let@token\bm@test@}`

\bm@test@   If looking at a single token, switch to `\bm@test@token`, else if looking at a `{ }`
group, grab the whole group with `\bm@group`. A `\bgroup` token will take the
wrong branch here (currently not trapped).

194 `\def\bm@test@{%`
195 `  \ifx\@let@token\bgroup`
196 `    \expandafter\bm@group`
197 `  \else`
198 `    \expandafter\bm@test@token`
199 `  \fi}`

\bm@gr@up   If faced with a group, If we are in math mode, stick it in a `\boldsymbol` like con-
struct and then recurse on `\bm@expand`. Otherwise just use `\bfseries\boldmath`.
The actual test is deferred till 'run time'. Here and elsewhere could deal with the
inner list with an inner call to `\bm`, but that doesn't seem to gain very much, and
complicates the code quite a bit.

   `#1` is either `\boldmath` or `\heavymath`. Need to add an extra set of explicit
braces around `#2` as otherwise the math style commands applied in `\mathchoice`
might only apply to the first half of an `\over` construction.

200 `\def\bm@gr@up#1#2{%`
201 `  \bm@add{{\bm@gr@@p#1{{#2}}}}}`

\bm@gr@@p   `#1` is either `\boldmath` or `\heavymath`.

202 `\def\bm@gr@@p#1#2{%`
203 `  \ifmmode`
204 `    \bm@mchoice#1{#2}{#2}{#2}{#2}%`
205 `  \else`
206 `    \bfseries#1#2%`
207 `  \fi}`

\bm@test@token   If not facing a `{ }` group then test to see what we have. Basic idea: Trap
`\mathchardef` tokens, character tokens, and calls to `\mathchar`, `\mathaccent`,
etc, and change the *math-group* (fam) to point at the bold version. Other things
just copy straight over to the command being built. (Anything inside a `\mathop`
or similar will end up being made bold as the `\mathop` will be copied over, but its
argument will be made bold by the group code above.

208 `\def\bm@test@token#1{%`
209 `  \let\bm@next\@empty`

   Stop here. Note that it is vital that the terminating token is non-expandable
and defined, rather than the usual LaTeX terminators `\@nil` or `\@@`. (Worse still
would be a 'quark' like `docstrip`'s `\qStop`.)

210 `  \ifx#1\@@end`

13

\bm@mathchoice uses macro arguments, so need to make the tail recursion explicit here. All the other cases recurse by way of \afterassignment which means all the trailing \fi are eaten while making the assignment.

```
211    \else\ifx#1\mathchoice
212      \let\bm@next\bm@mathchoice
```

The main point: Find these expressions, and change the mathgroup.

```
213    \else\ifx#1\mathchar
214      \afterassignment\bm@mathchar\count@
215    \else\ifx#1\mathaccent
216      \afterassignment\bm@mathaccent\count@
217    \else\ifx#1\delimiter
218      \afterassignment\bm@delimiter\count@
219    \else\ifx#1\radical
220      \afterassignment\bm@radical\count@
```

Need to trap spaces otherwise digits will get turned to bold mathchars.

```
221    \else\ifx#1\mkern
222      \bm@register#1{\muskip\z@}%
223    \else\ifx#1\kern
224      \bm@register#1\skip@
225    \else\ifx#1\penalty
226      \bm@register#1\count@
```

\vcopy is a flag to copy the next group unchanged to the result command.

```
227    \else\ifx#1\unvcopy
228      \let\bm@next\bm@add
229    \else\ifcat\noexpand#1\relax
```

Other command, look if it's a mathchardef token (otherwise just add it).

```
230      \xdef\meaning@{\meaning#1}%
231      \expandafter\bm@mchar@test\meaning@""""\@nil#1%
```

Character token. If it is of catcode 11 or 12, get its mathcode. If that is "8000 replace the token by its active version, and then let bm expansion look again at the character. Being really active this time, it will expand away (probably).

If the previous token was \left or \right, get the delcode instead of the mathcode.

```
232    \else\ifcat.\ifcat a#1.\else#1\fi
233      \count@\mathcode`#1\relax
234      \ifnum\count@=\mathcode`\'%
235        \begingroup\uccode`\~`#1\uppercase{\endgroup
236          \def\bm@next{\bm@expand~}}%
237      \else
238        \ifx\bm@previous\left
239          \count@\delcode`#1\relax
240          \bm@delimiter
241        \else
```

Here we need to check for LuaTeX merging mathchar values with Umathchar.

```
242          \ifnum\count@>"8000
243            \Umathcharnumdef\@tempa\count@
244            \xdef\meaning@{\meaning\@tempa}%
245            \expandafter\bm@mchar@test\meaning@""""\@nil\@tempa
246          \else
```

14

```
247            \bm@mathchar
248          \fi
249        \fi
250      \fi
251    \else
```

And final possibility: a character token of catcode other than 11 or 12.

```
252      \bm@add{#1}%
253    \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
254    \bm@next}
```

\bm@define   End code for `\bmdefine`. Define the given command name to the robust form of the accumulated code.

     If `\bm@command` is equal to `\@gtempa` then it is a macro whose expansion is a single call to `\mathchar`, so that can be optimised with a `\mathchardef`.

```
255 \def\bm@define#1{%
256    \begingroup
257      \ifx\bm@command\@gtempa
258        \def\mathchar{\global\mathchardef#1}%
259        \bm@command
260      \else
```

Rather than simply `\let#1\bm@command`, make the defined command robust. `\bm@first` is normally empty, but might be something like `\DOTSI` which needs to be lifted to the top level, in front of any `\protect` because of the lookahead mechanism used in the amsmath package.

```
261        \toks@\expandafter{\bm@command}%
262        \xdef#1{\bm@first\noexpand\bm@protect\noexpand#1{\the\toks@}}%
263      \fi
264    \endgroup}
```

\bm@protect   Commands defined by `\bmdefine` re-insert themselves if protection is enabled.

```
265 \def\bm@protect#1{%
266    \ifx\protect\@typeset@protect
267      \expandafter\@firstofone
268    \else
269      \protect#1\expandafter\@gobble
270    \fi}
```

\bm@mchoice   `\boldsymbol`, more or less. #1 is either `\boldmath` or `\heavymath`.

```
271 \def\bm@mchoice#1#2#3#4#5{%
272    \mathchoice{\hbox{#1$\displaystyle\m@th#2$}}%
273             {\hbox{#1$\textstyle\m@th#3$}}%
274             {\hbox{#1$\scriptstyle\m@th#4$}}%
275             {\hbox{#1$\scriptscriptstyle\m@th#5$}}}
```

\bm@m@thchoice   Action if you find a `\mathchoice`. Add the bold version to `\bm@command` then recurse.

     #1 is either `\boldmath` or `\heavymath`.

```
276 \def\bm@m@thchoice#1#2#3#4#5{%
277    \bm@add{\bm@mchoice#1{#2}{#3}{#4}{#5}}}
```

\bm@register   Combined code for setting up `\bm@r@gister` with the correct register type.

```
278 \def\bm@register#1#2{%
279   \def\@tempa{#1\the#2}%
280   \afterassignment\bm@r@gister#2}
```

\bm@r@gister   `\mkern` itself would transfer to `\bm@command` without any special test, but any explicit dimension following would be converted to `\mathchar`. So trap this and grab the muskip as a muskip. This is used in `\iiint`. `\penalty` was needed for the AMS version of `\colon`, and so do most of the others as well.

```
281 \def\bm@r@gister{%
282   \bm@xadd{\@tempa\space}}
```

\bm@mathchar   Change the family (math group) of a mathcode and then use the modified code with `\mathchar`. If there is no suitable bold font in the current math version, use the original unmodified mathcode, but switch to `\boldmath` (if there is a bold font there) or use 'poor man's bold'. Note that these other possibilities are only possible here, not for the otherwise similar code for `\delimiter` or `\mathaccent`, as those commands must work with fonts from the same math version.

     Finally recurse down the list.

```
283 \def\bm@mathchar{%
```

```
284   \@tempcntb\count@
285   \let\@tempa\bm@group
```

     `\bm@changefam` will isolate the math group from the mathcode and look up the offset in the current table.

```
286   \bm@changefam{}%
```

     If the mathcode has changed, then just add the new `\mathchar` (saving `\@gtempa` allows `\bmdefine` to optimise this to a mathchardef if it turns out to be the only symbol in the argument).

```
287   \ifnum\count@>\@tempcntb
```

```
288     \ifx\bm@command\@empty
289       \xdef\@gtempa{\mathchar\the\count@\space}%
290     \fi
291     \bm@xadd{\mathchar\the\count@\space}%
292   \else
```

     Otherwise grab the math class from the math code and add that (locally zapping `\bm@expand` as we don't want to recurse at this point).

```
293     \begingroup
294       \divide\count@"1000
295       \let\bm@expand\relax
296       \bm@xadd\bm@class
297     \endgroup
```

`\@tempa` will be `\bm@group` (which applies `\boldmath` and `\mathchoice`) unless it was changed by `\bm@changefam` to `\bm@pmb` (which applies a 'poor man's bold' construction in a `\mathchoice`).

```
298     \edef\@tempb{%
299       \noexpand\@tempa{\mathchar\the\count@\space}}%
300     \@tempb
301   \fi}
```

\bm@umathchar Version of `\bm@mathchar` for `\Umathchar`, this is easier as no need to take apart the number, the match class and fam are provided as distinct arguments.

```
302 \def\bm@umathchar#1#2#3{%
303 \@tempcnta#2\relax
304 \count@\bm@table
305 \ifnum\count@=\z@
306   \bm@gr@up\boldmath{\Umathchar#1 #2 #3 }%
307 \else
308   \ifnum\count@=\m@ne
309   \else
310     \advance\@tempcnta\count@
311   \fi
312 \bm@xadd{\Umathchar#1\space
313                     \the\@tempcnta\space\space
314                     #3\space}%
315 \fi}
```

\bm@pmb Add a poor man's bold construction to the list being built.

```
316 \def\bm@pmb#1{%
317   \bm@add{\bm@pmb@{#1}}}
```

\bm@pmb@ `\pmb` variant. (See TeXBook, or AMS amsbsy package). This one takes a bit more care to use smaller offsets in subscripts.

```
318 \def\bm@pmb@#1{{%
319   \setbox\tw@\hbox{$\m@th\mkern.4mu$}%
320   \mathchoice
321     \bm@pmb@@\displaystyle\@empty{#1}%
322     \bm@pmb@@\textstyle\@empty{#1}%
323     \bm@pmb@@\scriptstyle\defaultscriptratio{#1}%
324     \bm@pmb@@\scriptscriptstyle\defaultscriptscriptratio{#1}}}
```

\bm@pmb@@ Helper macro. Box #3 and set it three times in the style #1, offset by an amount reduced by the ratio specified in #2.

```
325 \def\bm@pmb@@#1#2#3{{%
326   \setbox\z@\hbox{$\m@th#1#3$}%
327   \dimen@#2\wd\tw@
328   \rlap{\copy\z@}%
329   \kern\dimen@
330   \raise1.5\dimen@\rlap{\copy\z@}%
331   \kern\dimen@
332   \box\z@}}%
```

\bm@class Convert a numeric math class back to a math class command. `\mathord` is omitted in class 0 and 7 to save space and so things work out right in constructions such as `x^a` where `x^\mathord{a}` would not work.

```
333 \def\bm@class{%
334   \ifcase\count@
335     \or
336     \mathop\or
337     \mathbin\or
338     \mathrel\or
339     \mathopen\or
340     \mathclose\or
```

17

```
341     \mathpunct\or
342  \fi}
```

\bm@add   A version of `\g@addto@macro` that internally uses a `\begingroup` rather than a brace group[4], to save creating a mathord.

As need to redefine it anyway, save some tokens by making it specific to `\bm@command`, and to execute `\bm@expand` to continue the loop.

```
343 \def\bm@add#1{%
344   \begingroup
345     \toks@\expandafter{\bm@command#1}%
346     \xdef\bm@command{\the\toks@}%
347   \endgroup
348   \bm@expand}
```

\bm@xadd   An `\xdef` version of `\bm@add`.

```
349 \def\bm@xadd#1{%
350   \begingroup
351     \toks@\expandafter{\bm@command}%
352     \xdef\bm@command{\the\toks@#1}%
353   \endgroup
354   \bm@expand}
```

\bm@mathaccent   `\mathaccent` version of `\bm@mathchar`.

```
355 \def\bm@mathaccent{%
356 \bm@changefam{}%
```

The next four lines were added a v1.0e. Without them `\bm{\hat{A}}` makes the accent bold using `\bm` but the group `{A}` is made bold via a `\mathchoice` construction as for any other group, as `\bm` does not attempt to parse inside brace groups. While that produces something acceptable for lower case letters, it produces $\hat{A}$ which is not too good. The braces may simply be omitted: `\bm{\hat A}` would work, producing $\hat{A}$, however I did not want to document such a restriction, so now modify bm so that such brace groups are handled gracefully.

It would be possible to locally make mathaccents take an argument during the bm look-ahead, so the brace groups would then vanish during expansion, however I would then need to explicitly skip past ⟨*filler*⟩ and also make sure that the end of parse token was not gobbled in marginal cases like `$\bm\hat$`.

So instead do the following which gets rid of ⟨*filler*⟩ with a redefinition of `\relax`, and just locally changes `\bm@group` so that instead of doing a `\mathchoice` it simply adds `\bgroup` and `\egroup` around the tokens, and lets bm modify the tokens of the 'argument'. This means that `\bm{\hat{A}}` now produces

```
\mathaccent 29790 \bgroup \mathchar 30017 \egroup
```

The inner math list is a single mathchar, and so TeX will not box it, and the math accent will correctly position, taking into account the skewchar information.

As the normal bm lookahead is used, it is automatic that the parse will end without trying to go past `\bm@end`.

One disadvantage is that the group will mean that `\bm@previous` will not be correctly updated. However that is only used for delimiter checking, so can not matter here.

---

[4]This bug is fixed in the LaTeX kernel of 1996/12/01

```
357  \begingroup
358  \def\bm@group##1{\endgroup\bm@xadd{\bgroup}##1\egroup}%
359  \def\bm@test@token{\endgroup\bm@test@token}%
360  \let\relax\@empty

361  \bm@xadd{\mathaccent\the\count@\space}}
```

\bm@delimiter Change both families (math groups) of a delcode and then use the modified code
with \delimiter. Don't change code '0' as that denotes a null delimiter.

```
362  \def\bm@delimiter{%
363    \ifnum\count@>\z@
364      \bm@changefam{}%
365      \bm@changefam{000}%
366    \fi
367    \bm@xadd{\delimiter\the\count@\space}}%
```

\bm@radical Same for \radical.

```
368  \def\bm@radical{%
369    \bm@changefam{}%
370    \bm@changefam{000}%
371    \bm@xadd{\radical\the\count@\space}}%
```

\bm@mchar@ Catcode 12 \mathchar, for \ifx tests.

```
372  \edef\bm@mchar@{\meaning\mathchar}
```

\bm@umchar@ Catcode 12 \Umathchar, for \ifx tests.

```
373  \edef\bm@umchar@{\string\U\expandafter\@gobble\meaning\mathchar}
```

\bm@mchar@test Test if the \meaning starts with \mathchar. If it does, grab the value into \count@
and call \bm@mathchar, else just copy the command into the accumulated tokens.
#1, #2, #3 are all \meaning produced tokens, or 'dummy tokens' added at the time
this is called. #4 is the original token, in case decide not to use the \meaning.

```
374  \def\bm@mchar@test#1"#2"#3"#4"#5\@nil#6{%
375    \xdef\meaning@{#1}%
376    \ifx\meaning@\bm@mchar@
377      \count@"#2\relax
378      \bm@mathchar
379    \else
```

Test for \Umathchar.

```
380      \ifx\meaning@\bm@umchar@
381        \bm@umathchar{"#2}{"#3}{"#4}%
382      \else
```

Some other command: copy it straight over. If it is the first thing added, and it
is a \relax token, save it in \bm@first for use in \bm@define.

```
383        \ifx\bm@previous\@empty
384          \ifx\relax#6%
385            \gdef\bm@first{#6}%
386          \fi
387        \fi
388        \bm@add{#6}%
389      \fi
390    \fi}
```

\bm@changefam Pull out one specified hex digit and passes it to \bm@modify to change. argument is empty normally but 000 to access the second math group in a delimiter code.

```
391 \def\bm@changefam#1{%
392   \@tempcnta\count@
393   \divide\@tempcnta"1000#1 %
394   \multiply\@tempcnta"1000#1 %
395   \advance\@tempcnta-\count@
396   \divide\@tempcnta-"100#1 %
```

Having isolated the required math group (fam), look up the offset in the current table.

```
397   \@tempcnta\bm@table
```

If the offset is −1, keep \count@ unchanged, but set \@tempa to use poor man's bold. Otherwise increment \count@ to change the math group specified.

```
398   \ifnum\@tempcnta=\m@ne
399     \let\@tempa\bm@pmb
400   \else
401     \multiply\@tempcnta"100#1 %
402     \advance\count@\@tempcnta
403   \fi}
```

\bm@prime Support '. Earlier versions did not make the prime bold in a'.
        \bm{a''} will now produce (with the normal encodings)

```
    \mathchar 30049
    \bm@prime \mathchar 1584 \relax
    \bm@prime \mathchar 1584 \relax
```

So \bm@prime does essentially the same as the active definition of ', which is to start a superscript group then keep adding \prime for each ' (or \bm@prime) following. Here modified to grab a \relax delimited argument and use that instead of \prime. \bm@prime is locally \let to ' so the \ifx tests in \pr@m@s don't need changing.

```
404 \def\bm@prime{^\bgroup
405   \let\bm@prime'%
406   \def\prim@s##1\relax{##1\futurelet\@let@token\pr@m@s}%
407   \prim@s}
```

\boldsymbol Finally, to ease conversion of documents between this package and the amsbsy
\heavysymbol package:

```
408 \let\boldsymbol\bm
```

```
409 \let\heavysymbol\hm
```

```
410 ⟨/package⟩
```