The lualatex-math package*

Philipp Stephani
p.stephani2@gmail.com

2022/01/01

Contents

1 Introduction

LuaTeX brings major improvements to all areas of TeX typesetting and programming. They are made available through new primitives or the embedded Lua interpreter, and combining them with existing LaTeX 2ε packages is not a task the average LaTeX user should have to care about. Therefore a multitude of LaTeX 2ε packages have been written to bridge the gap between documents and the new features. The lualatex-math package focuses on the additional possibilities for mathematical typesetting. The most eminent of the new features is the ability to use Unicode and OpenType fonts, as provided by Will Robertson's unicode-math package. However, there is a smaller group of changes unrelated to Unicode: these are to be dealt with in this package. While in principle most TeX documents written for traditional engines should work just fine with LuaTeX, there is a small number of breaking changes that require the attention of package authors. The lualatex-math package tries to fix some of the issues encountered while porting traditional macro packages to LuaLaTeX.

The decision to write patches for existing macro packages should not be made lightly: monkey patching done by somebody different from the original package author ties the patching package to the implementation details of the patched functionality and breaks all rules of encapsulation. However, due to the lack of

---

*This document corresponds to lualatex-math v1.12, dated 2022/01/01.

alternatives, it has become an accepted way of providing new functionality in LaTeX. To keep the negative impact as small as possible, the lualatex-math package patches only the LaTeX 2ε kernel and a small number of popular packages. In general, this package should be regarded as a temporary kludge that should be removed once the math-related packages are updated to be usable with LuaTeX. By its very nature, the package is likely to cause problems; in such cases, please refer to the issue tracker[1].

2 Interface

The lualatex-math package can be loaded with `\usepackage` or `\RequirePackage`, as usual. It has no options and no public interface; the patching is always done when the package is loaded and cannot be controlled. As a matter of course, the lualatex-math package needs LuaLaTeX to function; it will produce error messages and refuse to load under other engines and formats. The package depends on the expl3 bundle, the etoolbox package and the filehook package. The lualatex-math package is independent of the unicode-math package; the fixes provided here are valid for both Unicode and legacy math typesetting.

Currently patches for the LaTeX 2ε kernel and the amsmath, mathtools and icomma packages are provided. It is not relevant whether you load these packages before or after lualatex-math. They should work as expected (and ideally you shouldn't notice anything), but if you load other packages that by themselves overwrite commands patched by this package, bad things may happen, as it is usual with LaTeX.

`\mathstyle`

`\frac`, `\binom`, `\genfrac`

One user-visible change is that the new `\mathstyle` primitive should work in all cases after the lualatex-math package has been loaded, provided you use the high-level macros `\frac`, `\binom`, and `\genfrac`. The fraction-like TeX primitives like `\over` or `\atopwithdelims` and the plain TeX leftovers like `\brack` or `\choose` cannot be patched, and you shouldn't use them.

3 Implementation of the LaTeX 2ε package

3.1 Requirements

```
⟨*package⟩
⟨@@=lltxmath⟩
\NeedsTeXFormat{LaTeX2e}[2020/02/02]
\RequirePackage{expl3}[2018/06/18]
\ProvidesExplPackage{lualatex-math}{2022/01/01}{1.12}%
  {Patches for mathematics typesetting with LuaLaTeX}
\RequirePackage { etoolbox } [ 2007/10/08 ]
\cs_if_exist:NF \newluabytecode
  { \RequirePackage { luatexbase } [ 2010/05/27 ] }
\directlua{require("lualatex-math")}
```

`\@@_restore_catcode:N`

Executing the exhaustive expansion of `\@@_restore_catcode:N`⟨*character token*⟩ restores the category code of the ⟨*character token*⟩ to its current value.

```
\cs_new_nopar:Npn \@@_restore_catcode:N #1 {
  \char_set_catcode:nn { \int_eval:n { `#1 } }
    { \char_value_catcode:n { `#1 } }
}
```

---

[1] https://github.com/phst/lualatex-math/issues

We use the macro defined above to restore the category code of the dollar sign. There are packages that make the dollar sign active; hopefully they get loaded after the packages we are trying to patch.

```
\exp_args:Nx \AtEndOfPackage {
  \@@_restore_catcode:N \$
}
\char_set_catcode_math_toggle:N \$
```

3.2 Messages

`luatex-required`

Issued when not running under LuaTeX.

```
\msg_new:nnn { lualatex-math } { luatex-required } {
  The~ lualatex-math~ package~ requires~ LuaTeX. \\
  I~ will~ stop~ loading~ now.
}
```

`macro-expected`

Issued when trying to patch a non-macro. The first argument must be the detokenized macro name.

```
\msg_new:nnn { lualatex-math } { macro-expected } {
  I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
}
```

`wrong-meaning`

Issued when trying to patch a macro with an unexpected meaning. The first argument must be the detokenized macro name; the second argument must be the actual detokenized meaning; and the third argument must be the expected detokenized meaning.

```
\msg_new:nnn { lualatex-math } { wrong-meaning } {
  I've~ expected~ #1~ to~ have~ the~ meaning \\
  #3, \\
  but~ it~ has~ the~ meaning \\
  #2.
}
```

`patch-macro`

Issued when a macro is patched. The first argument must be the detokenized macro name.

```
\msg_new:nnn { lualatex-math } { patch-macro } {
  I'm~ going~ to~ patch~ macro~ #1.
}
```

3.3 Initialization

Unless we are running under LuaTeX, we issue an error and quit immediately.

```
\sys_if_engine_luatex:F {
  \msg_error:nn { lualatex-math } { luatex-required }
  \endinput
}
```

3.4 Patching

`\@@_temp:w`

A scratch macro.

```
\cs_new_eq:NN \@@_temp:w \prg_do_nothing:
```

`\@@_patch:NNnnn`
`\@@_patch:cNnnn`

The auxiliary macro `\@@_patch:NNnnn`⟨*command*⟩⟨*factory command*⟩`{`⟨*parameter text*⟩`}{`⟨*expected replacement text*⟩`}{`⟨*new replacement text*⟩`}` tries to patch ⟨*command*⟩. If ⟨*command*⟩ is undefined, do nothing. Otherwise it must be a macro with the given ⟨*parameter text*⟩ and ⟨*expected replacement text*⟩, created by the given ⟨*factory command*⟩ or equivalent. In this case it will be overwritten using the ⟨*parameter text*⟩ and the ⟨*new replacement text*⟩. Otherwise issue a warning and don't overwrite.

```
\cs_new_protected_nopar:Npn \@@_patch:NNnnn #1 #2 #3 #4 #5 {
  \cs_if_exist:NT #1 {
    \token_if_macro:NTF #1 {
      \group_begin:
      #2 \@@_temp:w #3 { #4 }
      \cs_if_eq:NNTF #1 \@@_temp:w {
        \msg_info:nnx { lualatex-math } { patch-macro }
          { \token_to_str:N #1 }
        \group_end:
        #2 #1 #3 { #5 }
      } {
        \msg_warning:nnxxx { lualatex-math } { wrong-meaning }
          { \token_to_str:N #1 } { \token_to_meaning:N #1 }
          { \token_to_meaning:N \@@_temp:w }
        \group_end:
      }
    } {
      \msg_warning:nnx { lualatex-math } { macro-expected }
        { \token_to_str:N #1 }
    }
  }
}
\cs_generate_variant:Nn \@@_patch:NNnnn { c }
```

`\@@_set_mathchar:NN`

The macro `\@@_set_mathchar:NN`⟨*control sequence*⟩⟨*token*⟩ defines the ⟨*control sequence*⟩ as an extended mathematical character shorthand whose mathematical code is given by the mathematical code of the character `` ` ``⟨*token*⟩. We cannot use the `\Umathcharnumdef` primitive here since we would then rely on the `\Umathcodenum` primitive which is currently broken.[2]

```
\cs_new_protected_nopar:Npn \@@_set_mathchar:NN #1 #2 {
  \Umathchardef #1
  \lua_now:e {
    lualatex.math.print_class_fam_slot( \int_eval:n { `#2 } )
  }
  \scan_stop:
}
```

`\@@_before_package:nn`
`\@@_after_package:nn`

The macro `\@@_before_package:nn{`⟨*package*⟩`}{`⟨*code*⟩`}` executes the ⟨*code*⟩ before the ⟨*package*⟩ is loaded. Accordingly, `\@@_after_package:nn{`⟨*package*⟩`}{`⟨*code*⟩`}` executes the ⟨*code*⟩ after the ⟨*package*⟩ is loaded. If the ⟨*package*⟩ is already loaded, nothing happens. We prefer using native LaTeX 2ε hooks if possible.

```
\@ifl@t@r \fmtversion { 2021/11/15 } {
  \cs_new_protected_nopar:Npn \@@_before_package:nn #1 #2 {
    \AddToHook { package/#1/before } { #2 }
  }
  \cs_new_protected_nopar:Npn \@@_after_package:nn #1 #2 {
    \AddToHook { package/#1/after } { #2 }
  }
}{
  \@ifl@t@r \fmtversion { 2020/10/01 } {
    \cs_new_protected_nopar:Npn \@@_before_package:nn #1 #2 {
      \AddToHook { package/before/#1 } { #2 }
    }
```

---

[2] http://tug.org/pipermail/luatex/2012-October/003794.html

```
    \cs_new_protected_nopar:Npn \@@_after_package:nn #1 #2 {
      \AddToHook { package/after/#1 } { #2 }
    }
  } {
    \RequirePackage { filehook } [ 2011/03/09 ]
    \cs_new_protected_nopar:Npn \@@_before_package:nn #1 #2 {
      \AtBeginOfPackageFile { #1 } { #2 }
    }
    \cs_new_protected_nopar:Npn \@@_after_package:nn #1 #2 {
      \AtEndOfPackageFile { #1 } { #2 }
    }
  }
}
```

`\@@_after_package_or_now:nn`

The macro `\@@_after_package_or_now:nn{`⟨*package*⟩`}{`⟨*code*⟩`}` executes the ⟨*code*⟩ after the ⟨*package*⟩ is loaded. If the ⟨*package*⟩ is already loaded, the ⟨*code*⟩ is executed immediately.

```
\cs_new_protected_nopar:Npn \@@_after_package_or_now:nn #1 #2 {
  \@ifpackageloaded { #1 } { #2 } { \@@_after_package:nn { #1 } { #2 } }
}
```

3.5 LaTeX 2ε kernel

LuaTeX enables access to the current mathematical style via the `\mathstyle` primitive. For this to work, fraction-like constructs (e. g., ⟨*numerator*⟩ `\over` ⟨*denominator*⟩) have to be enclosed in a `\Ustack` group. `\frac` can be patched to do this, but the plain TeX remnants `\choose`, `\brack` and `\brace` should be discouraged.

`\frac`

Here we assume that nobody except amsmath redefines `\frac`. This is obviously not the case, but we ignore other packages (e. g., nath) for the moment. We only patch the LaTeX 2ε kernel definition if the amsmath package is not loaded; the corresponding patch for amsmath follows below. Since `\frac` is declared by `\DeclareRobustCommand`, we must patch the macro `\frac␣`.

```
\AtEndPreamble {
  \@ifpackageloaded { amsmath } { } {
    \@@_patch:cNnnn { frac~ } \cs_set:Npn { #1 #2 } {
      {
        \begingroup #1 \endgroup \over #2
      }
    } {
```

To do: do we need the additional set of braces around `\Ustack`?

```
      {
        \Ustack { \group_begin: #1 \group_end: \over #2 }
      }
    }
  }
}
```

3.6 amsmath

The popular amsmath package is subject to three LuaTeX-related problems:

- The `\mathcode` primitive is used several times, which fails for Unicode math characters. `\Umathcode` should be used instead.

- Legacy font dimensions are used for constructing stacks in the `\substack` command and the `subarray` environment. This doesn't work if a Unicode math font is selected.
- The fraction commands `\frac` and `\genfrac` don't use the `\Ustack` primitive.

These problems have been fixed in version 2.17i of amsmath, so we don't attempt to patch it if that version is loaded.

`\c_@@_std_minus_mathcode_int`
`\c_@@_std_equal_mathcode_int`

These constants contain the standard TEX mathematical codes for the minus and the equal signs. We temporarily set the math codes to these constants before loading the amsmath package so that it can request the legacy math code without error.

```
\int_const:Nn \c_@@_std_minus_mathcode_int { "2200 }
\int_const:Nn \c_@@_std_equal_mathcode_int { "303D }
```

`\l_@@_minus_mathchar`
`\l_@@_equal_mathchar`

These mathematical characters are saved before amsmath is loaded so that we can temporarily assign the TEX values to the mathematical codes of the minus and equals signs. The amsmath package queries these codes, and if they represent Unicode characters, the package loading will fail. If amsmath has already been loaded, there is nothing we can do, therefore we use the non-starred version of `\AtBeginOfPackageFile`.

```
\tl_new:N \l_@@_minus_mathchar
\tl_new:N \l_@@_equal_mathchar
\@@_before_package:nn { amsmath } {
  \@ifpackagelater { amsmath } { 2020/08/24 } { } {
    \@@_set_mathchar:NN \l_@@_minus_mathchar \-
    \@@_set_mathchar:NN \l_@@_equal_mathchar \=
```

Now we temporarily reset the mathematical codes.

```
    \char_set_mathcode:nn { `\- } { \c_@@_std_minus_mathcode_int }
    \char_set_mathcode:nn { `\= } { \c_@@_std_equal_mathcode_int }
    \@@_after_package:nn { amsmath } {
```

`\std@minus`
`\std@equals`

The amsmath package defines the control sequences `\std@minus` and `\std@equal` as mathematical character shorthands while loading, but uses our restored mathematical codes, which must be fixed.

```
      \cs_set_eq:NN \std@minus \l_@@_minus_mathchar
      \cs_set_eq:NN \std@equal \l_@@_equal_mathchar
```

Finally, we restore the original mathematical codes of the two signs.

```
      \Umathcodenum `\- \l_@@_minus_mathchar
      \Umathcodenum `\= \l_@@_equal_mathchar
    }
  }
}
```

All of the following fixes work even if amsmath is already loaded.

`\@begindocumenthook`

amsmath repeats the definiton of `\std@minus` and `\std@equal` at the beginning of the document, so we also have to patch the internal kernel macro `\@begindocumenthook` which contains the hook code.

```
\@@_after_package_or_now:nn { amsmath } {
  \@ifpackagelater { amsmath } { 2020/08/24 } { } {
    \tl_replace_once:Nnn \@begindocumenthook {
      \mathchardef \std@minus \mathcode `\- \relax
      \mathchardef \std@equal \mathcode `\= \relax
    } {
```

```
    \@@_set_mathchar:NN \std@minus \-
    \@@_set_mathchar:NN \std@equal \=
  }
}
```

subarray

The `subarray` environment uses legacy font dimensions. We simply patch it to use LuaTeX font parameters (and LaTeX3 expressions instead of TeX arithmetic). Since subscript arrays are conceptually vertical stacks, we use the sum of top and bottom shift for the default vertical baseline distance (`\baselineskip`) and the minimum vertical gap for stack for the minimum baseline distance (`\lineskip`).

```
\@ifpackagelater { amsmath } { 2020/09/23 } { } {
  \@@_patch:NNnnn \subarray \cs_set:Npn { #1 } {
    \vcenter
    \bgroup
    \Let@
    \restore@math@cr
    \default@tag
    \baselineskip \fontdimen 10~ \scriptfont \tw@
    \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
⟨@@=⟩
    \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
⟨@@=ltxmath⟩
    \lineskiplimit \lineskip
    \ialign
    \bgroup
    \ifx c #1 \hfil \fi
    $ \m@th \scriptstyle ## $
    \hfil
    \crcr
  } {
    \vcenter
    \c_group_begin_token
    \Let@
    \restore@math@cr
    \default@tag
    \skip_set:Nn \baselineskip {
      \Umathstacknumup \scriptstyle
      + \Umathstackdenomdown \scriptstyle
    }
    \lineskip \Umathstackvgap \scriptstyle
    \lineskiplimit \lineskip
    \ialign
    \c_group_begin_token
    \token_if_eq_meaning:NNT c #1 { \hfil }
    \Ustartmath
    \m@th
    \scriptstyle
    \alignmark \alignmark
    \Ustopmath
    \hfil
    \crcr
  }
```

\frac

Since `\frac` is declared by `\DeclareRobustCommand`, we must patch the macro `\frac␣`.

```
  \@@_patch:cNnnn { frac~ } \cs_set:Npn { #1 #2 } {
    {
```

```
⟨@@=⟩
        \begingroup #1 \endgroup \@@over #2
      }
    } {
      {
        \Ustack { \group_begin: #1 \group_end: \@@over #2 }
⟨@@=lltxmath⟩
      }
    }
```

\genfrac Generalized fractions are typeset by the `\genfrac` command. Since `\genfrac` is declared by `\DeclareRobustCommand`, we have to patch the macro `\genfrac␣`.

```
    \@@_patch:cNnnn { genfrac~ } \cs_set:Npn {
      #1 #2 #3 #4 #5 #6
    } {
      {
        \@mathstyle { #4 }
        \genfrac@choice o { #1 }
        {
          \begingroup #5 \endgroup
⟨@@=⟩
          \ifx @ #3 @ \@@over \else \@@above \fi #3 \relax
          #6
        }
        \genfrac@choice c { #2 }
      }
    } {
      {
        \@mathstyle { #4 }
        \genfrac@choice o { #1 }
        {
          \Ustack {
            \group_begin: #5 \group_end:
            \tl_if_empty:nTF { #3 } {
              \@@over
            } {
              \@@above #3 \scan_stop:
            }
⟨@@=lltxmath⟩
            #6
          }
        }
        \genfrac@choice c { #2 }
      }
    }
  }
}
```

3.7 mathtools

mathtools' `\cramped` command and others that make use of its internal version use a hack involving a null radical. LuaTeX has primitives for setting material in cramped mode, so we make use of them.

In newer versions of mathtools, this issue is fixed, in which case we skip the patch.

\MT_cramped_internal:Nn The macro `\MT_cramped_internal:Nn`⟨*style*⟩`{`⟨*expression*⟩`}` typesets the ⟨*expression*⟩ in the cramped style corresponding to the given ⟨*style*⟩ (`\displaystyle` etc.);

all we have to do in LuaTeX is to select the correct primitive. Rewriting the user-level `\cramped` command and employing `\mathstyle` would be possible as well, but we avoid this way since we want to patch only a single command.

```
\@@_after_package_or_now:nn { mathtools } {
  \@ifpackagelater { mathtools } { 2021/03/28 } { } {
    \@@_patch:NNnnn \MT_cramped_internal:Nn
      \cs_set_nopar:Npn { #1 #2 } {
      \setbox \z@ \hbox {
        $
        \m@th
        #1
        \kern -\nulldelimiterspace
        \radical \z@ { #2 }
        $
      }
      \ifx #1 \displaystyle
        \dimen@ = \fontdimen 8 \textfont 3
        \advance \dimen@ .25 \fontdimen 5 \textfont 2
      \else
        \dimen@ = 1.25 \fontdimen 8
        \ifx #1 \textstyle
          \textfont
        \else
          \ifx #1 \scriptstyle
            \scriptfont
          \else
            \scriptscriptfont
          \fi
        \fi
        3
      \fi
      \advance \dimen@ -\ht\z@
      \ht\z@ = -\dimen@
      \ifvmode \leavevmode \fi
      { }
      \box\z@
    } {
```

Here the additional set of braces is absolutely necessary, otherwise the changed mathematical style would be applied to the material after the `\mathchoice` construct. As the original command works in both text and math mode, we use `\ensuremath` here.

```
      {
        \ensuremath {
          \use:c { cramped \cs_to_str:N #1 } #2
        }
      }
    }
  }
}
```

3.8 icomma

The icomma package uses `\mathchardef` to save the mathematical code of the comma character. This breaks for Unicode fonts. The incompatibility was noticed by Peter Breitfeld.[3]

---

[3] https://groups.google.com/forum/#!topic/de.comp.text.tex/Cputk-AJS5I/discussion

`\mathcomma` icomma defines the mathemathical character shorthand `\icomma` at the beginning of the document, therefore we again patch `\@begindocumenthook`.

```
\@@_after_package_or_now:nn { icomma } {
  \@ifl@t@r \fmtversion { 2020/10/01 } {
    \hook_gput_code:nnn { begindocument } { lualatex-math } {
      \@@_set_mathchar:NN \mathcomma \,
      \mathcode `\, = "8000 ~
    }
    \hook_gset_rule:nnnn
      { begindocument } { lualatex-math } { voids } { icomma }
  } {
    \tl_replace_once:Nnn \@begindocumenthook {
      \mathchardef \mathcomma \mathcode `\,
    } {
      \@@_set_mathchar:NN \mathcomma \,
    }
  }
}
⟨/package⟩
```

4 Implementation of the LuaLaTeX module

For the Lua module, we use the standard luatexbase-modutils template.

```
⟨*lua⟩
lualatex = lualatex or {}
lualatex.math = lualatex.math or {}
luatexbase.provides_module({
  name = "lualatex-math",
  date = "2013/08/03",
  version = 1.3,
  description = "Patches for mathematics typesetting with LuaLaTeX",
  author = "Philipp Stephani",
  licence = "LPPL v1.3+"
})
```

`unpack` The function `unpack` needs to be treated specially as it got moved around in Lua 5.2.

```
local unpack = unpack or table.unpack
```

```
local cctb = luatexbase.catcodetables or
  {string = luatexbase.registernumber("catcodetable@string")}
```

`print_class_fam_slot` The function `print_class_fam_slot` takes one argument which must be a number. It interprets the argument as a Unicode code point whose mathematical code is printed in the form ⟨*class*⟩␣⟨*family*⟩␣⟨*slot*⟩, suitable for the right-hand side of `\Umathchardef`.

```
function lualatex.math.print_class_fam_slot(char)
  local code = tex.getmathcode(char)
  local class, family, slot = unpack(code)
  local result = string.format("%i %i %i ", class, family, slot)
  tex.sprint(cctb.string, result)
end
```

```
return lualatex.math
⟨/lua⟩
```

Change History

- v0.1
  - General: Initial version 1
- v0.2
  - General: Added patch for the icomma package 9
- v0.3
  - General: Patched math group allocation to gain access to all families 5
- v0.3a
  - General: Updated for changes in l3kernel 1
- v0.3b
  - `\@begindocumenthook`: Another update for a change in l3kernel 6
- v0.3c
  - `\@@_set_mathchar:NN`: l3kernel renamed `\lua_now:x` to `\lua_now_x:n` 4
- v1.0
  - General: Switched to l3docstrip 1
- v1.1
  - `\@@_set_mathchar:NN`: Update reasoning why `\Umathcharnumdef` is not used here 4
  - General: Add fix and unit test for amsopn 8
- v1.10
  - General: Skip patch if mathtools is recent enough 8
    - Use new LaTeX 2ε hook management if available 9
- v1.11
  - General: Adapt to March 2021 changes to mathtools 8
- v1.12
  - `\@@_after_package:nn`: Use the new generic hook names if available 4
- v1.2
  - `\l_@@_equal_mathchar`: Replace removed macro `\chk_if_free_cs:N` 6
- v1.3
  - General: Stop using the deprecated `module` function 10
- v1.3a
  - `\@@_set_mathchar:NN`: l3kernel has (currently) dropped `\lua_now_x:n` 4
- v1.4
  - `\MT_cramped_internal:Nn`: Added `\ensuremath` to work around issue 11 9
  - General: Removed patch for math group allocation; the kernel itself now supports all available math families 5
- v1.4a
  - `\@@_set_mathchar:NN`: `\lua_now_x:n` is back 4
  - General: Avoid `\RequireLuaModule` 2
    - Load luatexbase only if required 2
    - Load all of luatexbase 10
    - Pick up new name for string catcode table where available 10
    - Use expl3 versions of LuaTeX math primitives 2
- v1.5
  - General: Removed patch for `\Mathstrutbox@`; amsmath now has a definition usable in LuaLaTeX 7
    - Removed unused helper macro `\@@_char_dim:NN` 6
    - Removed unused Lua function `print_fam_slot` 10
- v1.6
  - General: Removed patch for `\newmcodes@`; amsmath now has a definition usable in LuaLaTeX 8
- v1.7
  - `\genfrac`: Adapt patch to changes in amsmath 8
- v1.8
  - `\@@_set_mathchar:NN`: `\lua_now_x:n` is now called `\lua_now:e` 4
    - Stop using `\…:D` control sequences 4
  - `\frac`: Stop using `\…:D` control sequences 5, 7

- **`\genfrac`**: Stop using `\…:D` control sequences . . . 8
- General: Stop using `\…:D` control sequences . . . 6
- **`subarray`**: Stop using `\…:D` control sequences . . . 7

v1.9
- **`\@begindocumenthook`**: Don’t patch newer versions of amsmath . . . 6
- **`\MT_cramped_internal:Nn`**: Stop using `\…:D` control sequences . . . 9
- **`\frac`**: Adapt to changes in LaTeX 2ε kernel . . . 5
- **`\l_@@_equal_mathchar`**: Don’t patch newer versions of amsmath . . . 6
- General: Require 2020 version of LaTeX 2ε . . . 2
  - Use builtin LaTeX 2ε hooks if available . . . 2
- **`subarray`**: Don’t patch newer versions of amsmath . . . 7
  - Stop using `\…:D` control sequences . . . 7

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.