The **tensor**[*] package for LaTeX2e

Philip G. Ratcliffe[†]
Dipartimento di Scienza e Alta Tecnologia
Università degli Studi dell'Insubria—Como

(v2.2, last revision 2023/07/18)

Abstract

This is a complete revision and extension of Mike Piff's original `tensor` package; it defines two commands for typesetting tensors with mixed upper and lower indices in which the correct horizontal spacing must be observed. Various forms of alignment are available and spaces may be replaced by dots or other symbols. Consistent preposing of indices is now made possible while backwards compatibility is maintained. A special-purpose command to typeset nuclides is also defined.

1 Introduction

It is common in both physics and mathematics to use tensors with mixed upper and lower indices in which the relative horizontal positions and spacing are significant, for example

$$\Gamma^{\mu}{}_{\nu\rho}, \quad R^{\mu}{}_{\nu}{}^{\rho}{}_{\sigma} \quad \text{or} \quad \epsilon^{\mu\nu\rho}{}_{\sigma}.$$

The macros defined in this package automatically maintain consistent horizontal positioning. Another common need addressed is the preposing of upper and lower indices, as in

$${}_{\mathrm{H}}\langle q', t'|\mathcal{U}(t,t')|q,t\rangle_{\mathrm{H}} \quad \text{or} \quad {}^{14}_{6}\mathrm{C}.$$

Note the correct spacing of the pre-index H in the above example. It should also be noted that constant vertical positioning is maintained for lone indices; consider the following (examine carefully the last lower index o on the right):

$$\mathrm{M}^{o}_{o}|{}_{o}\mathrm{M} \quad \textit{cf.} \quad \mathrm{M}^{o}_{o}|_{o}\mathrm{M},$$

where the former group was typeset using `\indices`, the latter using '`_`' and '`^`'.

2 Usage

Two robust math-mode commands, `\tensor` and `\indices`, are defined (the first of which remains backwards compatible with Mike Piff's original definition). A new, robust text- and math-mode command, `\nuclide`, is also defined specifically for typesetting nuclides, as in the above example.

---

[*] Based on and extending the original package of the same name by Mike Piff (1996/06/03).

[†] E-mail: philip.ratcliffe@uninsubria.it

2.1 User commands

`\indices` To produce a mathematical expression (typically a tensor) with mixed upper and lower indices, simply enter ⟨*object*⟩`\indices{ ^`⟨*sup*$_1$⟩`_`⟨*sub*$_1$⟩ `^`⟨*sup*$_2$⟩`_`⟨*sub*$_2$⟩ `...}`. Thus, in math mode it is sufficient to type *e.g.*

`M\indices{^a_b^{cd}_e}` to obtain $M^{a}{}_{b}{}^{cd}{}_{e}$.

`\tensor` This variant has been retained in a completely backwards compatible form while also being considerably extended; the syntax for the previous expression is `\tensor{M}{^a_b^{cd}_e}`, for which the resulting output is identical. The extended form of `\tensor` defined here has an optional argument for indices to be placed *before* the tensor, thus:

`\tensor[^a_b^c_d]{M}{^e_f^g_h}` produces ${}^{a}{}_{b}{}^{c}{}_{d}M^{e}{}_{f}{}^{g}{}_{h}$.

A fairly robust (if somewhat crude) attempt is made to ensure the correct spacing and skew of the preposed indices with respect to the tensor object itself.

Note that also `\sb` and `\sp` may be used in place of '`_`' and '`^`' respectively for both the above macros.

`\indices*` `\tensor*` These two macros have starred forms, which collapse the spacing (*i.e.* return to standard form). While `\indices*` is clearly redundant (and is included merely for symmetry), `\tensor*` also *right* justifies the *pre*-index strings, so that *e.g.* nuclides may be typeset as follows (though see below for a purpose-built command):

`\tensor*[^{14}_6]{\mathrm{C}}{}` produces ${}^{14}_{6}\mathrm{C}$.

For those familiar with the `amsmath` package, this is more-or-less a generalisation of (though *not intended* as a substitute for) the `\sideset` command (which itself is *only valid* for objects defined with `\mathop`). Note that to use `\tensor*` as a substitute for `\sideset`, it is necessary to insert a `\nolimits` command, thus:

`\tensor*[^*_*]{\prod\nolimits}{^*_*}` produces ${}^{*}_{*}\prod\nolimits^{*}_{*}$.

The output appears identical to that of `\sideset{_*^*}{_*^*}{\prod}`.

`*` The `\indices*` and `\tensor*` forms *alone*, allow a `*` to also be placed as the first entry in either index-list argument, causing alignment (*left* justification) of the successive pairs of upper and lower indices. A warning is issued if a `*` appears in an argument string of either *non*-starred commands. Thus,

`\tensor*{M}{*^{i_1}_{m_1}^{i_2}_{m_2}^{i_3}_{m_3}^{i_4}_{m_4}}`

produces $M^{i_1\;i_2\;i_3\;i_4}_{m_1m_2m_3m_4}$ (*cf.* $M^{i_1i_2i_3i_4}_{m_1m_2m_3m_4}$).

Note that *no warning* is issued for improper pairing of successive indices.

`\indexmarker` In analogy with the `tensind` package, the command `\indexmarker` (by default empty) may redefined (using `\renewcommand`) to introduce a visible place marker for the index spaces (though not all `tensind` functionality is reproduced here); a simple possibility is

`\renewcommand\indexmarker{\cdot}`,

after which,

`\tensor{M}{^a_b^c_d}`

produces

$$M^{a\cdot c\cdot}_{\cdot b\cdot d} \quad \text{instead of} \quad M^{a}{}_{b}{}^{c}{}_{d}\,.$$

`\nuclide` This command, available in both math and text modes, is defined with the same purpose and result as the `\isotope` command (from the package of the same name). The syntax is

`\nuclide[`⟨*mass no.*⟩`][`⟨*atomic no.*⟩`]{`⟨*symbol*⟩`}`.

Thus, the earlier example of ${}^{14}_{6}\mathrm{C}$ is obtained with `\nuclide[14][6]{C}` while `\nuclide[4][2]{\alpha}` gives ${}^{4}_{2}\alpha$. As indicated by the square brackets, the ⟨*mass no.*⟩ and ⟨*atomic no.*⟩ arguments are optional. Note that there is a little more space (`1mu`) between the numbers and the chemical symbol than appears in the example constructed manually with `\tensor*`.

All the above-defined commands may be used recursively, *i.e.* a `\tensor` may occur as an index to another `\tensor` and should behave according to the current superscript–subscript level. The user commands are defined here as 'robust'; they may thus appear as so-called moving arguments, *i.e.* to `\caption`, `\section` *etc.*

`\nuclideFont` `\massnumFont` By default, the fonts used in `\nuclide` for the chemical symbol, mass and atomic numbers are `\mathrm`; *i.e.*, `\nuclideFont` (for the chemical symbol) is initially defined as `\mathrm` and `\massnumFont` (for the mass/atomic numbers) as `\nuclideFont` (for backwards compatibility). This then now allows for independent font variation of the chemical symbol and mass/atomic numbers. Both macros may be reset with `\renewcommand` to `\mathsf`, `\mathbf`, `\mathtt` *etc.*, or simply `\relax` (this last for `\nuclideFont` has the chemical symbol font default to `\mathit` for correct spacing, while for `\massnumFont` the mass and atomic numbers revert to standard math font).

2.2 Package options

As of v2.2, the package includes four options relating to the vertical alignment of indices. LaTeX's behaviour in this regard is not always optimal or what the user may desire. Consider the following output (constructed using '`_`' and '`^`'.):

$$\epsilon^{\mu}{}_{\rho}{}^{\nu}{}_{\tilde\lambda}\, g^{\mu\nu}\,.$$

While the indices within each single mathematical object are mutually vertically aligned correctly, between separate objects they may not be. This is because LaTeX sets the baseline according to the height and depth of the given indices on a per-object basis. To obviate such behaviour, this package takes the simplest route of using `\smash` to hide the height and depth of each superscript and subscript string so that they are always set with the same baselines. This naturally leads to a somewhat cramped form (superscripts are set a little too low and subscripts high) and so a specially defined `\strut` is included, which slightly raises superscripts and lowers subscripts; by default, this is only implemented in displayed math, as the impact on inline text may be too disruptive.

`align` `text` `nosmash` `nostrut` The options thus introduced are `align`, `text`, `nosmash` and `nostrut`. The first implements both `\smash` and a `\strut` as outlined above, with `text` extending the implementation of the `\strut` to inline text, while `nosmash` and `nostrut` cancel the single effects (using both entirely negates `align`). Option ordering is irrelevant and the last three are inoperative without the first.

`\tensorSmash` `\tensorStrut` The desired effects are implemented via two internal commands, which may also be redefined by the user. The first, `\tensorSmash`, is set equal to `\smash`, which then takes each index string as an argument. The second, `\tensorStrut`, is set equal to the height of '`l`' and depth of '`j`' in the relevant font, (by default though only inside displayed math environments) and is appended to each `\smash`'ed index string.

2.3 Caveats

Grouping of multi-token indices should be performed as normal (*i.e.* via enclosure within a brace pair `{}`). Moreover, owing to the method by which index strings are parsed, any index constructs such as `\mathrm{H}` must also be entirely enclosed in braces, thus: `\indices{_{\mathrm{H}}^x}`.

Spacing is not guaranteed to always appear optimal, especially when between *pre*-pended indices and the tensor object itself. Recall too that screen viewing often distorts small spaces owing to resolution effects.

2.4 External package requirements

No external packages are required or called.

2.5 Package conflicts

There are few conflicts with standard LaTeX2e packages; a problem with the `color` package in the first version has now been corrected, as too a recently flagged problem with the `underscore` package.

However, the macros defined here fail as arguments of `\bm` from the `bm` package (due to parsing conflicts) or, consequently, of macros defined by the `\maybebm` package. A work around for, say, a chapter or section header is

`{\let\nuclideFont\maybebm \nuclide[4][2]{\textup{He}}}`,

which should render $\boldsymbol{^{4}_{2}\mathbf{He}}$ in the header, but $^{4}_{2}\mathrm{He}$ in the contents listing.

3 Implementation

3.1 User options

First, the package options with their related `\if`... conditionals are defined and processed.

```
\newif\iftnsr@Aln
\DeclareOption{align}{\tnsr@Alntrue}
\newif\iftnsr@Txt
\DeclareOption{text}{\tnsr@Txttrue}
\newif\iftnsr@Sma \tnsr@Smatrue
\DeclareOption{nosmash}{\tnsr@Smafalse}
\newif\iftnsr@Str \tnsr@Strtrue
\DeclareOption{nostrut}{\tnsr@Strfalse}
\ProcessOptions
```

3.2 User commands

The `tensor` package defines three basic user commands:

`\tensor` The first takes three possible arguments (an optional index string to be *preposed*, the tensor object, the index string) and also has a starred form, which suppresses spacing (it is backwards compatible with Mike Piff's original version).

```
\DeclareRobustCommand\tensor{%
  \tnsr@Prp
  \@ifstar{\tnsr@Spcfalse\tnsr@Aux}{\tnsr@Spctrue\tnsr@Aux}%
}
```

`\indices` The second is a '*lightweight*' form, which is placed immediately *following* the tensor object, takes just one argument (the index string) and also has a starred form (this form was *not* however present in the original package).

```
\DeclareRobustCommand\indices{%
  \tnsr@Prp
  \@ifstar{\tnsr@Spcfalse\ndcs@Aux}{\tnsr@Spctrue\ndcs@Aux}%
}
```

`\nuclide` This additional new command takes one direct argument (an optional mass number) and two indirect arguments (an optional atomic number, the chemical symbol—these last two are handled by an auxiliary macro). Since usage is common in text, math mode is ensured.

```
\DeclareRobustCommand\nuclide[1][]{%
  \ncld@Mno{#1}%
  \ncld@Aux
}
```

`\nuclideFont` `\massnumFont` These set the fonts for `\nuclide`; the defaults are `\mathrm` for both `\nuclideFont` and `\massnumFont`. They may be redefined as *e.g.* `\mathsf`, `\mathbf`, `\mathtt`, `\mathit` *etc.*, or even simply `\relax` or `\renewcommand\nuclideFont{}`.

```
\newcommand\nuclideFont{\mathrm}
\newcommand\massnumFont{\nuclideFont}
```

3.3 Internal token registers

`\tnsr@Sps` `\tnsr@Sbs` `\tnsr@Spe` `\tnsr@Sbe` The token registers that hold the upper and lower index strings, and the most recent upper and lower index elements respectively:

```
\newtoks\tnsr@Sps
\newtoks\tnsr@Sbs
\newtoks\tnsr@Spe
\newtoks\tnsr@Sbe
```

`\ncld@Mno` This token register temporarily holds the mass number for `\nuclide`.

```
\newtoks\ncld@Mno
```

3.4 Internal switches

`\iftnsr@Spc` The switch to select or suppress index element spacing.

```
\newif\iftnsr@Spc
```

3.5 Internal macros

`\tnsr@Prp` `\tnsr@Wrn` Here we simply reset token registers and the warning macro before commencing.

```
\newcommand\tnsr@Wrn{}
\newcommand\tnsr@Prp{%
  \tnsr@Sps{}%
  \tnsr@Sbs{}%
  \def\tnsr@Wrn{}
}
```

`\ndcs@Aux` This (lightweight) auxiliary macro for `\indices` takes one argument (an index string); it calls `\tnsr@Set`, prints the indices and then issues any warnings.

```
\newcommand\ndcs@Aux[1]{%
  \tnsr@Erx
  \def\tnsr@Obj{}%
  \tnsr@Set{#1}%
  \tnsr@Fin
  \tnsr@Wrn
}
```

`\tnsr@Aux` This auxiliary macro for `\tensor` takes three possible arguments (an optional pre-index string, the tensor object, the post-index string) and passes everything via `\mathpalette` to `\tnsr@Plt`.

```
\newcommand\tnsr@Aux[3][]{%
  \tnsr@Erx
  \mathpalette{\tnsr@Plt{#1}{#3}}{#2}%
  \tnsr@Wrn
}
```

`\tnsr@Plt` This takes four arguments (the pre-index string—may be empty, the post-index, the current math style, the tensor object) and calls `\tnsr@Set` separately for both pre- and post-index strings.

```
\newcommand\tnsr@Plt[4]{%
  \def\tnsr@Obj{#3#4}%
  \def\tnsr@Tmp{#1}%
  \ifx\tnsr@Tmp\@empty\else
    \tnsr@Set{#1}%
    \hphantom{{}\tnsr@Fin}%
    \tnsr@Sps\expandafter{%
      \expandafter\tnsr@Krn\expandafter{\the\tnsr@Sps}%
    }%
    \tnsr@Sbs\expandafter{%
      \expandafter\tnsr@Krn\expandafter{\the\tnsr@Sbs}%
    }%
  \fi
  \tnsr@Set{#2}%
  #4\tnsr@Fin
}
```

`\tnsr@Set` This takes one argument (a pre- or post-index string) and starts processing.

```
\newcommand\tnsr@Set[1]{%
  \let\tnsr@Swx\relax
  \tnsr@Pro#1\tnsr@Err
}
```

`\tnsr@Krn` This has one argument (a processed index string) and inserts the necessary offsets.

```
\newcommand\tnsr@Krn[1]{%
  \settowidth\@tempdima{$\m@th\tnsr@Obj^{#1}\mkern-1mu$}%
  \kern-\@tempdima
  #1
  \settowidth\@tempdima{$\m@th\tnsr@Obj$}%
  \kern\@tempdima
}
```

`\tnsr@Pro` This is the index-string processing macro, it takes one argument (an index string):

```
\newcommand\tnsr@Pro[1]{%
  \ifx#1\tnsr@Err
    \let\tnsr@Nxt\relax
  \else
    \if#1*
      \iftnsr@Spc
        \gdef\tnsr@Wrn{%
          \PackageWarning{tensor}{%
            '*' not allowed in argument here; I shall ignore it.%
            \MessageBreak Either remove it or use '\string\tensor*'%
          }%
        }%
      \else
        \let\tnsr@Swx\tnsr@Swa
      \fi
      \let\tnsr@Nxt\tnsr@Pro
    \else
      \if#1^
        \def\tnsr@Nxt{\tnsr@Add{\tnsr@Sps}{\tnsr@Sbs}{\tnsr@Spe}}%
      \else
        \if#1_
          \def\tnsr@Nxt{\tnsr@Add{\tnsr@Sbs}{\tnsr@Sps}{\tnsr@Sbe}}%
        \else
          \tnsr@Err
          \let\tnsr@Nxt\tnsr@Pro
        \fi
      \fi
    \fi
  \fi
  \tnsr@Nxt
}
```

`\tnsr@Swa` Here we flip the state of `\tnsr@Swx` to `\tnsr@Swb`.

```
\newcommand\tnsr@Swa{\let\tnsr@Swx\tnsr@Swb}
```

`\tnsr@Swb` Here we flip the state of `\tnsr@Swx` to `\tnsr@Swa` then calculate and insert the necessary padding for horizontal index alignment.

```
\newcommand\tnsr@Swb{%
  \let\tnsr@Swx\tnsr@Swa
  \settowidth\@tempdima{$\m@th\tnsr@Obj{}^{\the\tnsr@Spe}$}%
  \settowidth\@tempdimb{$\m@th\tnsr@Obj{}_{\the\tnsr@Sbe}$}%
  \addtolength\@tempdima{-\@tempdimb}%
  \ifdim\@tempdima=\z@\else
    \ifdim\@tempdima>\z@
      \tnsr@Sbs\expandafter\expandafter\expandafter{%
        \expandafter\the\expandafter\tnsr@Sbs
        \expandafter\kern\the\@tempdima
      }%
    \else
      \@tempdima=-\@tempdima
      \tnsr@Sps\expandafter\expandafter\expandafter{%
        \expandafter\the\expandafter\tnsr@Sps
        \expandafter\kern\the\@tempdima
      }%
    \fi
  \fi
}
```

`\tnsr@Add` This macro takes four arguments (the token-register target for the next index token, the token-register target for the phantom element, the token-register target for the most-recent element, the next index token). It adds the next index token to the upper or lower string and (if spacing is *on*) a place-holder (`\tnsr@Hph`) of the same size to the lower or upper string, respectively. It also calls `\tnsr@Swx` to flip state (if activated). The use of `\leavevmode` avoids conflict with the `color` package.

```
\newcommand\tnsr@Add[4]{%
  #1\expandafter{\the#1\leavevmode{#4}}%
  \iftnsr@Spc
    #2\expandafter{\the#2\tnsr@Hph{#4}}%
  \fi
  #3{\leavevmode{#4}}%
  \tnsr@Swx
  \tnsr@Pro
}
```

`\tnsr@Hph` The place-holder macro, uses `\mathpalette` to call the contents `\tnsr@Mph`:

```
\newcommand\tnsr@Hph{\expandafter\mathpalette\expandafter\tnsr@Mph}
```

`\tnsr@Mph` The place-holder macro contents:

```
\newcommand\tnsr@Mph[2]{%
  \settowidth\@tempdima{$\m@th#1{#2}$}%
  \makebox[\@tempdima][c]{$\m@th#1\indexmarker$}%
}
```

`\indexmarker` The default (blank) placeholder for index spacing:

```
\newcommand\indexmarker{}
```

`\tnsr@Fin` Finally, we put the index strings into place:

```
\newcommand\tnsr@Fin{%
  ^{\tensorSmash{\the\tnsr@Sps}\tnsr@Str}%
  _{\tensorSmash{\the\tnsr@Sbs}\tnsr@Str}%
}
```

`\tensorSmash` Initialise `\tensorSmash` as `\relax` and then conditionally set it equal to `\smash` (it is user redefinable).

```
\let\tensorSmash\relax
\iftnsr@Aln
  \iftnsr@Sma
    \let\tensorSmash\smash
  \fi
\fi
```

`\tensorStrut` `\tnsr@Str` Initialise `\tensorStrut` as `\relax` and then conditionally set it to the height and depth of ‘`jl`’. By default, it is only applied to displayed math environments (passed on via `\tnsr@Str`, which is `\def`’ed as `\tensorStrut` to be user redefinable), but always (*i.e.* extended to inline text) if the package option `text` is present.

```
\newcommand\tensorStrut{}
\let\tnsr@Str\relax
\iftnsr@Aln
  \iftnsr@Str
    \renewcommand\tensorStrut{\vphantom{jl}}
    \iftnsr@Txt
      \def\tnsr@Str{\tensorStrut}
    \else
      \everydisplay\expandafter{\the\everydisplay\let\tnsr@Str\tensorStrut}
    \fi
  \fi
\fi
```

`\ncld@Aux` This auxiliary macro takes two arguments (an optional atomic number and the chemical symbol). The mass number is passed on via `\ncld@Mno`. Math mode is ensured since usage is common in text. The spacing is increased by `1mu` for better appearance.

```
\newcommand\ncld@Aux[2][]{%
  \ensuremath{%
    \tensor*[^{\massnumFont{\the\ncld@Mno}}_{\massnumFont{#1}}]%
      {\mkern1mu{\mathit{\nuclideFont{#2}}{}}}{}%
  }%
}
```

`\tnsr@Err` This is invoked in the only error situations considered.

```
\newcommand\tnsr@Err{}
\newcommand\tnsr@Erx{%
  \def\tnsr@Err{%
    \global\let\tnsr@Err\relax
    \PackageError{tensor}{%
      Misordered sub/superscript items\on@line;
      \MessageBreak index tokens may have been lost.
      \MessageBreak Press <return> and I shall try to continue%
    }{Index string probably has extra/missing ’^’ or ’_’.}%
  }%
}
```

Change History

v1.0
- General: original version 1

v2.0
- General: extended `\tensor`, added `\indices` and `\nuclide`, substituted `\newcommand` with `\DeclareRobustCommand` in user commands, documented and packaged 1

v2.1
- `\indexmarker`: added capability to insert place holders 8
- `\indices`: added starred form, for symmetry with `\tensor*` 5
- `\tnsr@Add`: added `\leavevmode`, to avoid color package conflict . . . 8
- `\tnsr@Krn`: slightly altered spacing 7
- `\tnsr@Mph`: substituted `\hbox` with `\makebox` 8

v2.2
- `\massnumFont`: added independent mass/atomic-no. font control . . 5
- `\tnsr@Fin`: added vertical alignment capability 8
- `\tnsr@Pro`: substituted `\ifx` with `\if` to avoid underscore package conflict 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.