The tabularew package*

Diego Saba

2009/06/01

Abstract

This article proposes an extended implementation of the LaTeX tabular environment. It adds a new command to gain access to a quantity, called here the *excess width*, involved in the process of calculating the column widths. This allows to modify the default algorithm, that produces undesirable effects in some circumstances.

Its principal merit is to solve the problem of centering multicolumn headings when they are wider than the text underneath them.

The same extension can easily be added in the future to the similar array environment and to the star forms of both.

This extension evaluates the whole table three times. It consumes more resources than the standard environment and cannot be made 100% compatible, as it conceptually should.

1 Introduction

This package extends the implementation of the tabular environment contained in the array package. More information can be found in [1].

Let us start with an example. Suppose that you have to modify the following table, in order to align the decimal separators.

sez.	σ [MPa]	τ [MPa]	$\beta_{\mathrm{pl},\sigma}$	$\beta_{\mathrm{pl},\tau}$	$S_{\mathrm{pl},\sigma}$	$S_{\mathrm{pl},\tau}$	S_{pl}	$S_{\mathrm{pl,am}}$
1	7,4	2,9	0,83	0,83	37,0	54,7	30,7	1,2
2	59,9	8,9	0,83	0,83	4,61	17,9	4,46	1,2
3	64,0	7,5	0,83	0,83	4,31	21,1	4,22	1,2
4	46,3	4,8	0,83	0,83	5,95	33,5	5,86	1,2
5	48,4	6,8	0,83	0,83	5,70	23,3	5,53	1,2

This can be done using a well known stratagem: each number can be split so that the integer part belongs to a column, and the separator sign and the fractional part belong to the following column. The labels must now span two columns.

The code will change accordingly:

*This file has version number v0.1, last revised 2009/06/01.

```
\begin{tabular}{*{9}{c}}
  sez. &
  $\sigma$ &
  $\tau$ &
  ...
  \hline
  1 & 7,4 & 2,9 & ...
  ...
\end{tabular}
```

⇒

```
\begin{tabular}{c*{8}{r@{}l}}
  sez. &
  \multicolumn{2}{c}{$\sigma$} &
  \multicolumn{2}{c}{$\tau$} &
  ...
  \hline
  1 & 7&,4 & 2&,9 & ...
  ...
\end{tabular}
```

The result is probably unexpected:

sez.	σ [MPa]	τ [MPa]	$\beta_{\mathrm{pl},\sigma}$	$\beta_{\mathrm{pl},\tau}$	$S_{\mathrm{pl},\sigma}$	$S_{\mathrm{pl},\tau}$	S_{pl}	$S_{\mathrm{pl,am}}$
1	7,4	2,9	0,83	0,83	37,0	54,7	30,7	1,2
2	59,9	8,9	0,83	0,83	4,61	17,9	4,46	1,2
3	64,0	7,5	0,83	0,83	4,31	21,1	4,22	1,2
4	46,3	4,8	0,83	0,83	5,95	33,5	5,86	1,2
5	48,4	6,8	0,83	0,83	5,70	23,3	5,53	1,2

Note that the last label is not centered on its data: it is aligned to the left and hangs to the right. The same is true of the third label. Actually, seven of the nine labels are wider than their data, though some of them by very little. When a `\multicolumn` is wider than the columns it spans, it is aligned to the left, disregarding the alignment directive.

The intent was to center all the labels, like this:

sez.	σ [MPa]	τ [MPa]	$\beta_{\mathrm{pl},\sigma}$	$\beta_{\mathrm{pl},\tau}$	$S_{\mathrm{pl},\sigma}$	$S_{\mathrm{pl},\tau}$	S_{pl}	$S_{\mathrm{pl,am}}$
1	7,4	2,9	0,83	0,83	37,0	54,7	30,7	1,2
2	59,9	8,9	0,83	0,83	4,61	17,9	4,46	1,2
3	64,0	7,5	0,83	0,83	4,31	21,1	4,22	1,2
4	46,3	4,8	0,83	0,83	5,95	33,5	5,86	1,2
5	48,4	6,8	0,83	0,83	5,70	23,3	5,53	1,2

To understand the reason of this behaviour, and the difficulty to "correct" it, the algorithm used by LaTeX and by the underlying TeX to compute the column widths must be examined in greater detail.

The column widths are first calculated without considering the `\multicolumn` cells, unless they span just one column and so are "fake" `\multicolumn`s. Let us call the result of this first computation the *intrinsic width* of a column. Then each `\multicolumn` in turn is taken into consideration. If it is wider than the columns it spans, comprehensive of the space between adjacent columns, the last column is widened to accommodate the `\multicolumn`. The order matters: first the `\multicolumn`s that end at the second column are considered, then those that end at the third column, and so on. Let us call the column widths so obtained

the *extrinsic widths*. The difference between the extrinsic width and the intrinsic width is the *excess width*[1].

The tabular environment relies on the TEX primitive `\halign`, from which it inherits the algorithm. It is explained in a different and more formal way in The TEXbook [2, page 235], but the result is the same.

This algorithm sometimes produces undesirable effects, as the example above shows and Donald Knuth himself points out [2, *ibidem*].

2 Features

The tabularew environment has the same syntax as the tabular environment, and should behave in the same way, except for the new features (and a new bunch of incompatibilities and bugs ...).

`\begin{tabularew}{`⟨*pream*⟩`}` ... `\end{tabularew}`

`\GetExcessWidth` In the body of the environment a new command is available.

`\GetExcessWidth{`⟨*column*⟩`}`
⟨*column*⟩ ⟶ ⟨optional sign⟩ ⟨number⟩

`\ExcessWidth` The ⟨*column*⟩ indication can be absolute or relative to the current position. It is relative if the first character is '`+`' or '`-`', otherwise it is absolute[2]. Its effect is to set the dimension register `\ExcessWidth` to the excess width of ⟨*column*⟩.

`\spew` The command[3]

`\spew{`⟨*factor*⟩`}{`⟨*column*⟩`}`

is just a shorthand for the idiom

`\GetExcessWidth{`⟨*column*⟩`}\hspace{`⟨*factor*⟩`\ExcessWidth}`

Since there is no way to access the widths directly during the evaluation of a table, a trick must be used. The table is evaluated three times. During the first pass every call to `\GetExcessWidth` sets `\ExcessWidth` to a null length and every call to `\multicolumn` generates an empty cell; thus the *intrinsic widths* are collected. The second pass restores the normal behaviour of `\multicolumn`; the *extrinsic widths* are collected and the *excess widths* calculated. In the third and last pass every call to `\ExcessWidth` returns the requested measure.

The column widths can change between the second and third pass, but the total width of the table will not, if `\spew` is used in the following way, that is the way I had in mind when this package was conceived.

When there is a `\multicolumn` cell, the columns crossed by it form a group. It is sometimes useful to access the excess width of the *last* column of the group from

[1] Other names that I have considered are *proper*, *inherent*, *implicit*, *essential*, *innate* for *intrinsic* and *relational*, *explicit*, *accidental*, *acquired* for *extrinsic*.

[2] I cannot see a useful application for the '`-`' form, but it was natural to include it.

[3] The name stands for *add a* SPace *proportional to the* Excess Width, but the effect seems a bit unfortunate.

within the preceding columns *of the same group*. Moreover, the sum of the spaces `\spew`ed in each of the preceding columns should not exceed the excess width of the last column; that is, the sum of all the ⟨*factor*⟩ arguments should not exceed one. In this way the net result is to redistribute the excess width in a flexible way among the group of columns.

`\CurrentColumn` `\NumberOfColumns`

The two counters `\CurrentColumn` and `\NumberOfColumns` need not normally be used explicitly, since `\GetExcessWidth` makes use of them behind the scenes. Nonetheless I decided to expose them too to the user. They should only be read and not assigned to.

3 Example

The "correct" table shown in the introduction was obtained by

```
\begin{tabularew}{c*{8}{>{\spew{.5}{+1}}r@{}l}}
  sez. &
  \multicolumn{2}{c}{$\sigma$} &
  \multicolumn{2}{c}{$\tau$} &
   ...
  \hline
  1 & 7&,4 & 2&,9 & ...
   ...
\end{tabularew}
```

4 The Code

I have succeeded in writing this code, thanks to David Carlisle's tabulary environment and his well-commented code. I have boldly and shamelessly copied it, without even fully understanding it, and modified it to suite my needs.

To simplify things, I have eliminated the `\verb` and colortbl support (at least for the moment).

```
⟨*package⟩
\RequirePackage{array}
\catcode`\Z=14
\DeclareOption{debugshow}{\catcode`\Z=9\relax}
\ProcessOptions
```

`\ExcessWidth` `\CurrentColumn` `\NumberOfColumns`

Allocate the registers for the user interface.

```
\newdimen\ExcessWidth
\newcount\CurrentColumn
\newcount\NumberOfColumns
```

`\tabularew` `\TEW@get@body` `\TEW@find@end` `\endtabularew`

The tabularew environment uses the same mechanism of grabbing its body as tabulary, tabularx, and the AMS alignment environments. The use of `{\ifnum0=`}\fi` to begin a grouping and of `\ifnum0=`{\fi}` to end it is discussed in tabularx. See also The TEXbook [2, page 385]. It is needed to allow the environment inside an alignment.

```
\def\tabularew{%
  \edef\TEW@{\@currenvir}%
  {\ifnum0=`}\fi
  \TEW@setup
  \toks@{}\TEW@get@body}
\long\def\TEW@get@body#1\end
  {\toks@\expandafter{\the\toks@#1}\TEW@find@end}
\def\TEW@find@end#1{%
  \def\@tempa{#1}%
  \ifx\@tempa\TEW@\def\@tempa{\end{#1}}\expandafter\@tempa
  \else\toks@\expandafter
    {\the\toks@\end{#1}}\expandafter\TEW@get@body\fi}
\def\endtabularew{%
Z \message{^^J^^JEW Table - first pass^^J}%
  \TEW@firstpass
Z \message{\@spaces\@spaces\space - second pass^^J}%
  \TEW@secondpass
Z \message{\@spaces\@spaces\space - last pass^^J}%
  \TEW@lastpass
  \TEW@cleanup
  \ifnum0=`{\fi}}
```

`\TEW@setup` Save locally all the things that tabularew will assign to globally. The values will be restored at the end by `\TEW@cleanup`.

```
\def\TEW@setup{%
  \edef\@restorecounters{%
    \global\NumberOfColumns\the\NumberOfColumns
    \global\CurrentColumn\the\NumberOfColumns\relax}%
```

The excess widths are stored as macro definitions because this way is easier, and probably more efficient, than dealing with an "array" of dimension registers.

```
  \count@\z@
  \@tempswatrue
  \@whilesw\if@tempswa\fi{%
    \advance\count@\@ne
    \expandafter\ifx\csname TEW@\the\count@\endcsname\relax
      \@tempswafalse
    \else
      \expandafter\let\csname TEW@S\the\count@
        \expandafter\endcsname\csname TEW@\the\count@\endcsname
    \fi}%
```

These will only change locally.

```
  \let\@arraycr\TEW@arraycr
  \let\multicolumn\TEW@multicolumn
  \ExcessWidth\z@
```

This will only exist locally. But I'm not sure of this choice.

```
  \let\spew\TEW@spew
  }
```

`\TEW@cleanup`

```
\def\TEW@cleanup{%
  \count@\z@
  \@tempswatrue
  \@whilesw\if@tempswa\fi{%
    \advance\count@\@ne
    \expandafter\ifx\csname TEW@S\the\count@\endcsname\relax
      \@tempswafalse
    \else
      \global\expandafter\let\csname TEW@\the\count@
        \expandafter\endcsname\csname TEW@S\the\count@\endcsname
    \fi}%
  \@restorecounters
  }
```

`\TEW@firstpass` Build a table that will never show up and is built in a special way with the purpose of taking measures. `\ExcessWidth` is null and `\multicolumn`s are empty. The *intrinsic widths* are collected.

```
\def\TEW@firstpass{%
  \let\multicolumn\TEW@multicolumnempty
  \TEW@tabsample
```

```
  \let\multicolumn\TEW@multicolumn
  \let\@computation\TEW@firstcomp
  \TEW@measure
  }
```

`\TEW@secondpass` Now `\multicolumn`s are honoured and the *extrinsic widths* are collected.

```
\def\TEW@secondpass{%
  \TEW@tabsample
  \let\@computation\TEW@secondcomp
  \TEW@measure
  }
```

`\TEW@lastpass` Just build the real table.

```
\def\TEW@lastpass{%
  \TEW@tabfinal
  }
```

`\TEW@tabsample` Add a row at the end of the table that won't affect the column widths. This row will later be analyzed to collect the widths. The last row provided by the user can't serve this purpose because it could contain multicolumns or be hidden by an hline.

```
\def\TEW@tabsample{%
  \let\GetExcessWidth\@GetExcessWidthz@
  \let\@mkpream\TEW@mkpream
  \setbox\z@\hbox{%
  \gdef\@halignto{}%
  \col@sep\tabcolsep
  \let\d@llarbegin\begingroup\let\d@llarend\endgroup
    \expandafter\TEW@tabarray\the\toks@
      \crcr\omit
      {\count@\NumberOfColumns
        \xdef\TEW@save@row{}%
        \loop
          \advance\count@\m@ne
        \ifnum\count@>\z@
          \xdef\TEW@save@row{\TEW@save@row&\omit}%
        \repeat}%
      \TEW@save@row
      \endarray
    \global\setbox\@ne\lastbox
    }%
  }
```

`\TEW@tabfinal` This mimics a regular tabular, the only difference being that `\CurrentColumn` is updated before evaluating the content of each cell and the command `\GetExcessWidth` is made available.

```
\def\TEW@tabfinal{%
  \leavevmode
  \let\GetExcessWidth\@GetExcessWidth
```

```
  \let\@mkpream\TEW@mkpream
  \gdef\@halignto{}%
  \col@sep\tabcolsep
  \let\d@llarbegin\begingroup\let\d@llarend\endgroup
  \expandafter\TEW@tabarray\the\toks@\endarray}
```

`\TEW@tabarray` `\TEW@array` Handle the optional position argument.

```
\def\TEW@tabarray{\@ifnextchar[{\TEW@array}{\@array[t]}}
\def\TEW@array[#1]{\@array[t]}
```

`\TEW@@mkpream` `\TEW@@arraycr` `\TEW@@multicolumn` Saved versions.

```
\let\TEW@@mkpream\@mkpream
\let\TEW@@arraycr\@arraycr
\let\TEW@@multicolumn\multicolumn
```

`\TEW@mkpream` This is a one-shot customized version, that redifines itself to the regular version. It's not clear to me why this is needed. Maybe because the regular version is used to process the multicolumns' preambles?

```
\def\TEW@mkpream{%
  \global\NumberOfColumns\@ne
  \global\CurrentColumn\@ne
  \let\@addamp\TEW@addamp
  \global\let\@mkpream\TEW@@mkpream % needed!
  \TEW@@mkpream}
```

`\TEW@arraycr`

```
\def\TEW@arraycr{%
  \global\CurrentColumn\@ne
  \TEW@@arraycr
  }
```

`\TEW@multicolumn` For the multicolumn mechanism to work, the first token of the expansion must be `\omit`.

```
\long\def\TEW@multicolumn#1#2#3{%
  % Can't place anything before \omit
  \TEW@@multicolumn{#1}{#2}{\global\advance\CurrentColumn\@ne#3}%
  \global\advance\CurrentColumn#1%
  \global\advance\CurrentColumn\m@ne
  \ignorespaces}
```

`\TEW@multicolumnempty` To behave exactly as explained in the introduction, the special case of a "fake" multicolumn should be dealt with. I don't think this would give any practical advantage, though.

```
\long\def\TEW@multicolumnempty#1#2#3{\multispan#1\relax}
```

`\TEW@addamp` During the evaluation of `\@mkpream`, while processing the pramble, each '`&`' increases `\NumberOfColumns`. After that, it will keep it's value. During the evaluation of `\@preamble`, while building a row, each '`&`' increases `\CurrentColumn`. The latter is reset at the end of each row when `\@arraycr` is evaluated.

```
\def\TEW@addamp{%
  \if@firstamp\@firstampfalse
  \else
    \global\advance\NumberOfColumns\@ne
    \expandafter\def\expandafter\@preamble\expandafter{\@preamble
      &\global\advance\CurrentColumn\@ne}%
  \fi
  }%
```

`\TEW@measure` Take the last row apart, unbox it, take each cell in turn and note its width.

```
\def\TEW@measure{%
  \setbox\z@\vbox{\unvbox\@ne\unskip\global\setbox\@ne\lastbox}%
  \setbox\tw@\hbox{%
    \count@\NumberOfColumns
    \unhbox\@ne
    \loop
      \unskip
      \setbox\tw@\lastbox
    \ifhbox\tw@
      \@computation{\wd\tw@}%
      \advance\count@\m@ne
    \repeat
    }%
  }
```

`\TEW@firstcomp` The argument is stored in the database.

```
\def\TEW@firstcomp#1{%
Z \message{Col \the\count@: Intrinsic Width=\the#1^^J}%
  \TEW@width\xdef{\the#1}}
```

`\TEW@secondcomp` The argument is used to compute the excess width which is then stored in the database.

```
\def\TEW@secondcomp#1{%
  \TEW@width\dimen@
  \advance\dimen@-#1%
  \multiply\dimen@\m@ne
Z \message{Col \the\count@: Excess width=\the\dimen@^^J}%
  \TEW@width\xdef{\the\dimen@}}
```

`\TEW@width` A shorthand.

```
\def\TEW@width#1{%
  \expandafter#1\csname TEW@\the\count@\endcsname}
```

`\@GetExcessWidth` Look up the excess widths database.

```
\def\@GetExcessWidth#1{%
  \count@\CurrentColumn
```

Process the optional sign. Need to `\relax` at the end of the assignment, to prevent the following token from being expanded too early.

```
  \@ifnextchar+{\advance\count@}{%
    \@ifnextchar-{\advance\count@}{\count@}}#1\relax
```

Check that the column exists.

```
  \@tempswafalse
  \ifnum\count@>\z@
    \ifnum\count@>\NumberOfColumns
    \else\@tempswatrue
    \fi\fi
```

Retrieve the datum.

```
  \if@tempswa\TEW@width\ExcessWidth
  \else\ExcessWidth\z@
    \TEW@warn{nonexistent column \the\count@, assuming EW=0pt}%
  \fi
Z \message{EW in col. \the\CurrentColumn\space references
Z   col. \the\count@: \the\ExcessWidth^^J}%
  }
```

`\@GetExcessWidthz@` To be used in the first two passes, when the excess widths are still unknown.

```
\def\@GetExcessWidthz@#1{\ExcessWidth\z@}
```

`\TEW@spew` A useful shorthand.

```
\def\TEW@spew#1#2{\GetExcessWidth{#2}\hspace{#1\ExcessWidth}}
```

`\TEW@warn` Warning messages.

```
\def\TEW@warn{%
  \PackageWarning{tabularew}}

\catcode`\Z=11
```

⟨/package⟩

References

[1] M. Goossens, F. Mittelbach and A. Samarin. The LaTeX Companion. Addison-Wesley, Reading, Massachusetts, 1994.

[2] D. E. Knuth. The TeXbook (Computers & Typesetting Volume A). Addison-Wesley, Reading, Massachusetts, 1986.