The `supertabular` environment*

Johannes Braams and Theo Jurriens

2025/06/26

1 Introduction

The package supertabular offers a new environment, the supertabular environment. As the name indicates it is an extension of the normal tabular environment.

With the original tabular environment a tabular must always fit on *one* page. If the tabular becomes too large the text overwrites the page's bottom margin and you get an `Overfull vbox` message.

The supertabular environment uses the tabular environment internally, but it evaluates the used space every time it gets a `\\` command. If the tabular reaches the textheight, it automatically inserts an optional tabletail, an `\end{tabular}` command, starts a new page, a new tabular environment and inserts the optional tablehead on the new page continuing the tabular.

2 User interface

The package supertabular has three options, they control the amount of information that is written to the `.log` file.

1. The option errorshow (the default) doens't write any extra information.
2. The option pageshow writes information about when and why supertabular decides to break the tabular environment in order to produce a new page.
3. The option debugshow also adds information about each row that is added to the tabular.
4. The option estimate (the default) has the package use the old estimation-based algortihm to establish the height of individual rows.
5. The option calculate has the package use the new calculation-based algortihm to establish the height of individual rows.

Below is a description of the new commands and environments that this package provides.

`\tablefirsthead` The command `\tablefirsthead` takes one argument, it defines the contents of the first occurence of the tabular head.
The use of this command is optional. Don't forget to close the head by a `\\`.

`\tablehead` The command `\tablehead` takes one argument, it defines the contents of all

*This file has version number v4.2d, last revised 2025/06/26.

subsequent ocurrences of the tabular head.
Don't forget to close the head by a \\

`\tabletail` The command `\tabletail` takes one argument, it defines something which should be inserted before each `\end{tabular}`, except the last.

`\tablelasttail` The command `\tablelasttail` takes one argument, it defines something which should be inserted before the last `\end{tabular}`.
The use of this command is optional.

`\topcaption` `\bottomcaption` `\tablecaption` These commands all take the same arguments as LaTeX's standard `\caption` command. They provide a caption for the super-table, either at the top or at the bottom of the table. When `\tablecaption` is used the caption will be placed at the default location, which is at the top.

`supertabular` (*env.*) `supertabular*` (*env.*) The environments supertabular and supertabular* can be used much like the standard LaTeX environments tabular and tabular*.

`mpsupertabular` (*env.*) `mpsupertabular*` (*env.*) The environments mpsupertabular and mpsupertabular* work like the supertabular and supertabular* environments but put each page into a minipage first. Thus it is possible to have footnotes inside a mpsupertabular. The footnotetext is printed at the end of each page.

`\shrinkheight` The allowed maximimum height of a part of the supertabular on a page can be adjusted using the command `\shrinkheight`. It takes one argument, the length with which to shrink (positive value) or grow (negative value) the allowed height.

3 Weak points

- When the material of a normal entry (not a p-arg) becomes larger than the estimated `\ST@rowht`, overfull `\vbox`es will be produced.
- When the last p-arg on a page gets more than 4 lines (probably even more than 3 lines) it will result in an overfull `\vbox`. Also some combinations of `\baselinestretch` `\arraystretch` and a large font may lead to one row too much.
- if accidentally the last row of the tabular produces a newpage, on the next page the tabletail will be written immediately after the tablehead. Depending on the contents, this may result in an error message regarding misplaced `\noalign`.

  A quick but not very elegant solution: shrink the allowed height of the table with the command `\shrinkheight{...pt}` after the first `\\` of the supertabular.
- The mpsupertabular environment sometimes has problems with pagebreaks when footnotes appear in the lower part of the tabular.

4 Examples

Here is an example of a supertabular. First, here is (part of) the user input for the table below:

```
\begin{center}
\tablefirsthead{%
  \hline
```

```
  \multicolumn{1}{|c}{\tbsp Number} &
  \multicolumn{1}{c}{Number$^2$} &
  Number$^4$ &
  \multicolumn{1}{c|}{Number!} \\
  \hline}
\tablehead{%
  \hline
  \multicolumn{4}{|l|}{\small\sl continued from previous page}\\
  \hline
  \multicolumn{1}{|c}{\tbsp Number} &
  \multicolumn{1}{c}{Number$^2$} &
  Number$^4$ &
  \multicolumn{1}{c|}{Number!} \\
  \hline}
\tabletail{%
  \hline
  \multicolumn{4}{|r|}{\small\sl continued on next page}\\
  \hline}
\tablelasttail{\hline}
\bottomcaption{This table is split across pages}

\begin{supertabular}{|r@{\hspace{6.5mm}}|r@{\hspace{5.5mm}}|r|r|}
1   &     1  &        1  &           1    \\
2   &     4  &       16  &           2    \\
3   &     9  &       81  &           6    \\
4   &    16  &      256  &          24    \\[5mm]
...
19  &   361  &   130321  &  1.21645100E+17\\
20  &   400  &   160000  &  2.43290200E+18\\
\end{supertabular}
\end{center}
```

Then the table should be split across the page boundary:

Number	Number^2	Number^4	Number!
1	1	1	1
2	4	16	2
3	9	81	6
4	16	256	24
5	25	625	120
6	36	1296	720
7	49	2401	5040
8	64	4096	40320
9	81	6561	362880
10	100	10000	3628800
11	121	14641	39916800
12	144	20736	479001600

continued on next page

continued from previous page

Number	Number^2	Number^4	Number!
13	169	28561	6.22702080E+9
14	196	38416	8.71782912E+10
15	225	50625	1.30767437E+12
16	256	65536	2.09227899E+13
17	289	83521	3.55687428E+14
18	324	104976	6.40237370E+15
19	361	130321	1.21645100E+17
20	400	160000	2.43290200E+18

Table 1: This table is split across pages

Here is another example whith a `p` column-definition. The tablehead is the same as above. The tabletail is a double `\hline`; `\arraystretch` is set to `1.5` and the font size is `\small`.

Table 2: This table should also be split accross pages.

Number	Number^2	Number^4	Number!
1	1	1	here is a relative short entry
2	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
3	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
4	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
5	1	1	here is a relative short entry
6	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
7	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur

continued on next page

continued from previous page			
Number	Number^2	Number^4	Number!
8	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
9	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
10	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
11	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
12	1	1	here is a relative short entry
13	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
14	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
15	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
16	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
17	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
18	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur

Here is the same table again, but this time using the supertabular* environment and stretching the table to the full width of the text.

Table 3: This table should also be split accross pages.

Number	Number^2	Number^4	Number!
1	1	1	here is a relative short entry
			continued on next page

continued from previous page

Number	Number^2	Number^4	Number!
2	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
3	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
4	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
5	1	1	here is a relative short entry
6	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
7	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
8	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
9	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
10	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
11	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
12	1	1	here is a relative short entry
13	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
14	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
15	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
16	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur

continued on next page

continued from previous page			
Number	Number^2	Number^4	Number!
17	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur
18	1	1	and here is a long entry, where line breaks and line breaks and line breaks have to occur

5 Known problems

- When a float occurs on the same page as the start of a supertabular you can expect unexpected results.

  When the float was defined on the same page you might end up with the first part of the supertabular on a page by its own.

- You should not use the supertabular *inside* a floating-environment such as `table` as this will result in TEX trying to put the whole supertabular on *one* page.

- In some instances you might still end up with overfull `\vbox` messages.

- Sometimes the last page of the supertabular contains just an empty head an tail.

6 The Implementation

First we define a few options that control the level of tracing output this package delivers. the option errorshow is the default situation.

```
1 ⟨*package⟩
2 \newcount\c@tracingst
3 \DeclareOption{errorshow}{\c@tracingst\z@}
4 \DeclareOption{pageshow}{\c@tracingst\thr@@}
5 \DeclareOption{debugshow}{\c@tracingst5\relax}
```

In version 4.1g a new way of determining the height of the average row was introduced. Instead of using an estimation, based on the value of `\baselineskip` a computation was introduced, based on the size of the `\strutbox`. The effect of this new method was that the parial tabulars fill the page better, so less `underfull vbox` message would appear. Unfortunately this had a negative effect on existing documents (that don;t come out like they used to) and especially when a very small font (`\tiny` or `\scriptsize` was chosen for the tabular and the cells contain subscripts or superscripts. It turned out that the heigth and depth of a formula like 5^5_5 exceeds the size of `\strutbox`. The result is that the rows have more height (and/or depth) thatn what the algorithm computed. Hence, when more rows are added to the partial tabular than fit on the page. Especially on the first part thisis a probem as the partial tabular becomes too high and doesn't fit in the avialbale space. TEX then decides to move it to the next page, resulting in two consecutive pages that have a lot of white space on them.

```
\DeclareOption{calculate}{\def\ST@calculate@rowht{\ST@compute@rowht}}
\DeclareOption{estimate}{\def\ST@calculate@rowht{\ST@estimate@rowht}}
%\def\ST@calculate@rowht{\ST@estimate@rowht}
```

The default for the options is to only show errors and use the old estmation algorithm (so as not to break old documents).

```
\ExecuteOptions{errorshow,estimate}
\ProcessOptions
```

`\topcaption` `\bottomcaption` The user-commands `\topcaption` and `\bottomcaption` set the flag `@topcaption` to determine where to put the tablecaption. The default is to put the caption on the top of the table

```
\newif\if@topcaption \@topcaptiontrue
\def\topcaption{\@topcaptiontrue\tablecaption}
\def\bottomcaption{\@topcaptionfalse\tablecaption}
```

`\tablecaption` This command has to function exactly like `\caption` does, except it has to store its argument (and the optional argument) for later processing *within* the supertabular environment.

```
\long\def\tablecaption{%
  \refstepcounter{table}\@dblarg{\@xtablecaption}}
\long\def\@xtablecaption[#1]#2{%
  \long\gdef\@process@tablecaption{\ST@caption{table}[#1]{#2}}}
\global\let\@process@tablecaption\relax
```

`\ifST@star` This switch is used in the internal macros to remember which kind of environment was started.

```
\newif\ifST@star
```

`\ifST@mp` This switch is used in the internal macros to remember if the tabular should be put into a minipage.

```
\newif\ifST@mp
```

`\ST@wd` For the supertabular* environment it is necessary to store the intended width of the tabular.

```
\newdimen\ST@wd
```

`\ST@rightskip` `\ST@leftskip` `\ST@parfillskip` For the mpsupertabular environments we need special versions of `\leftskip`, `\rightskip` and `\parfillskip`.

```
\newskip\ST@rightskip
\newskip\ST@leftskip
\newskip\ST@parfillskip
```

`\ST@captionroom` When a supertabular is preceded by a caption that fact might not yet be visible in the amount of space occupoed on the page sofar. Therefore we include the possibility to reduce the height of the first part of the supertabular. In order to this we need a macro that indicates a caption has been put in front of the table. We do this to reduce the risk that the first part of the table is too high after all and is pushed onto the next page due to an overfull `\vbox` condition.

```
\def\ST@captionroom{\z@}
```

\ST@caption This is a redefinition of LaTeX's \@caption, \@makecaption is called within a group so as not to return to \normalsize globally. Also a fix is made for the 'feature' of the \@makecaption of the document class article and friends that a caption **always** gets a \vskip 10pt at the top and **none** at the bottom. If a user wants to precede his table with a caption this results in a collision.

```
\long\def\ST@caption#1[#2]#3{\par%
  \addcontentsline{\csname ext@#1\endcsname}{#1}%
                  {\protect\numberline{%
                      \csname the#1\endcsname}{\ignorespaces #2}}
  \begingroup
    \@parboxrestore
    \normalsize
    \if@topcaption \vskip -10\p@ \fi
    \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
    \if@topcaption \vskip 10\p@ \gdef\ST@captionroom{20\p@}\fi
  \endgroup}
```

\tablehead \tablehead activates the new tabular \cr commands.

\tablefirsthead

```
\newcommand\tablehead[1]{%
  \def\@ST@arg{#1}%
  \ifx\@ST@arg\@empty\gdef\@tablehead{}\else
    \gdef\@tablehead{%
    \noalign{%
        \global\let\@savcr=\\
        \global\let\\=\org@tabularcr}%
      #1%
      \noalign{\global\let\\=\@savcr}}%
  \fi}
\tablehead{}
```

It's possible to specify a different tablehead for the first 'part' of the table. That only needs to be used once so it 'undefines' itself at the end. That way we make sure that it doesn't accidentally get used for a second supertabular in the document.

```
\newcommand\tablefirsthead[1]{%
  \def\@ST@arg{#1}%
  \ifx\@ST@arg\@empty\gdef\@table@first@head{}\else
    \gdef\@table@first@head{%
      \noalign{%
        \global\let\@savcr=\\
        \global\let\\=\org@tabularcr}%
      #1%
      \noalign{%
        \global\let\\=\@savcr
        \global\let\@table@first@head\undefined
      }}%
  \fi}
```

\tabletail \tablelasttail If the user uses an extra amount of tabular-data (like \multicolumn) in \tabletail TEX starts looping because of the definition of \ST@cr. So make \\ act just like a \@tabularcr inside this tail to prevent the loop. Save and restore the value of \\.

```
\newcommand\tabletail[1]{%
  \def\@ST@arg{#1}%
```

```
\ifx\@ST@arg\@empty\gdef\@tabletail{}\else
  \gdef\@tabletail{%
    \noalign{%
      \global\let\@savcr=\\
      \global\let\\=\org@tabularcr}%
    #1%
    \noalign{\global\let\\=\@savcr}}%
\fi}
\tabletail{}
```

It's possible to specify a different tabletail for the last 'part' of the table. That only needs to be used once so it 'undefines' itself at the end. That way we make sure that it doesn't accidentally get used for a second supertabular in the document.

```
\newcommand\tablelasttail[1]{%
  \def\@ST@arg{#1}%
  \ifx\@ST@arg\@empty\gdef\@table@last@tail{}\else
    \gdef\@table@last@tail{%
      \noalign{%
        \global\let\@savcr=\\
        \global\let\\=\org@tabularcr}%
      #1%
      \noalign{%
        \global\let\\=\@savcr
        \global\let\@table@last@tail\undefined
      }}%
  \fi}
```

`\sttraceon` `\sttraceoff` There now is a possiblity to follow the decisions supertabular makes about breaking the tabular. This has to be enabled when converting this file with `docstrip` to a `.sty` file.

```
\newcommand\sttraceon{\c@tracingst5\relax}
\newcommand\sttraceoff{\c@tracingst\z@}
```

`\ST@trace` A macro that gets the trace message as its argument

```
\newcommand\ST@trace[2]{%
  \ifnum\c@tracingst>#1\relax
    \GenericWarning
      {(supertabular)\@spaces\@spaces}
      {Package supertabular: #2}%
  \fi
  }
```

`\ST@trace@cr` A variant of `\ST@trace` that can be called from within `\\` as that command is looking for an optional argument and will end up scanning the next line.

`\ST@save@lineno` But because this variant is called from within `\\` we need to save the current input linenumber before TEX starts scanning for the optional argument. If we don't, the reported linenumber depends on whether or not the optional argument is present...

```
\newcommand\ST@save@lineno{%
  \expandafter\gdef\expandafter\ST@LineNo\expandafter{%
    \the\inputlineno}}
```

Within `\ST@trace@cr` we can than locally modify `\on@line` to use this saved line number.

```
\newcommand\ST@trace@cr[2]{%
  \ifnum\c@tracingst>#1\relax
  \begingroup
  \edef\on@line{ on input line \ST@LineNo}%
    \GenericWarning
      {(supertabular)\@spaces\@spaces}
      {Package supertabular: #2}%
  \endgroup
  \fi
  }
```

`\ST@pageleft` This register holds the estimate of the amount of space left over on the current page. This is used in the decision when to start a new page.

```
\newdimen\ST@pageleft
```

`\shrinkheight` A command to diminish the value of `\ST@pageleft` if necessary.

```
\newcommand*\shrinkheight[1]{%
  \noalign{\global\advance\ST@pageleft-#1\relax}}
```

`\setSTheight` A command to set the value of `\ST@pageleft` if necessary.

```
\newcommand*\setSTheight[1]{%
  \noalign{\global\ST@pageleft=#1\relax}}
```

`\ST@headht` `\ST@tailht` The register `ST@headht` will hold the height of the first head of a supertabular environment; the register `\ST@tailht` will hold the height of table tail (if any)

```
\newdimen\ST@headht
\newdimen\ST@tailht
```

`\ST@pagesofar` The register `\ST@pagesofar` is used to store the estimate of the amount of page already filled up.

```
\newdimen\ST@pagesofar
```

`\ST@pboxht` The measured (total) height of a parbox-argument

```
\newdimen\ST@pboxht
```

`\ST@rowht` `\ST@stretchht` `\ST@prevht` The estimated height of a normal row is stored in `\ST@rowht`. The dimension register `\ST@stretchht` was used to store the difference between the 'normal' row height and the row height when `\arraystretch` has a non-standard value. This was used in the case where p-box entries are added to the tabular. The dimension register `\ST@prevht` is used to store the height of the previous row, to use it as an estimate for the height of the next row. This is needed for a better estimate of when to break the tabular.

```
\newdimen\ST@rowht
\newdimen\ST@prevht
```

`\ST@toadd` When a tabular row is ended with `\\[...]` we need to temporarily store the optional argument in `\ST@toadd`.

```
\newdimen\ST@toadd
```

`\ST@dimen` A private scratch dimension register.

```
\newdimen\ST@dimen
```

\ST@pbox A box register to temporarily store the contents of a parbox.

```
\newbox\ST@pbox
```

\ST@tabularcr These are redefinitions of \@tabularcr and \@xtabularcr. This is needed to
\ST@xtabularcr include \ST@cr in the definition of \@xtabularcr.
\ST@argtabularcr All redefined macros have names that are similar to the original names, except with a leading 'ST'.

```
% \changes{v4.1f}{2019/01/18}{Save the input linenumber before \TeX\
%    scans for an optional argument}
\def\ST@tabularcr{%
  {\ifnum0=`}\fi
  \ST@save@lineno
  \@ifstar{\ST@xtabularcr}{\ST@xtabularcr}}
\def\ST@xtabularcr{%
  \@ifnextchar[%]
    {\ST@argtabularcr}%
    {\ifnum0=`{\fi}\cr\ST@cr}}
\def\ST@argtabularcr[#1]{%
  \ifnum0=`{\fi}%
  \ifdim #1>\z@
    \unskip\ST@xargarraycr{#1}
  \else
    \ST@yargarraycr{#1}%
  \fi}
```

\ST@xargarraycr In this case we need to copy the value of the optional argument of \\ in our private
\ST@yargarraycr register \ST@toadd.

```
\def\ST@xargarraycr#1{%
  \@tempdima #1\advance\@tempdima \dp \@arstrutbox
  \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr
  \noalign{\global\ST@toadd=#1}\ST@cr}
```

Here we need to insert \ST@cr

```
\def\ST@yargarraycr#1{%
  \cr\noalign{\vskip #1\global\ST@toadd=#1}\ST@cr}
```

\ST@startpbox The macros that deal with parbox columns need to be redefined, because we need to know the size of the parbox.

```
\def\ST@startpbox#1{%
```

To achieve our goal we need to save the text in box.

```
  \setbox\ST@pbox\vtop\bgroup\hsize#1\@arrayparboxrestore}
```

\ST@astartpbox Our version of \@astartpbox for array.

```
\def\ST@astartpbox#1{%
  \bgroup\hsize#1%
  \setbox\ST@pbox\vtop\bgroup\hsize#1\@arrayparboxrestore}
```

\ST@endpbox Our version of \@endpbox and \@aendpbox.
\ST@aendpbox

```
\def\ST@endpbox{%
  \@finalstrut\@arstrutbox\par\egroup
  \ST@dimen=\ht\ST@pbox
  \advance\ST@dimen by \dp\ST@pbox
```

```
  \ifnum\ST@pboxht<\ST@dimen
    \global\ST@pboxht=\ST@dimen
  \fi
  \ST@dimen=\z@
  \box\ST@pbox\hfil}
```

The version for array

```
\def\ST@aendpbox{%
  \@finalstrut\@arstrutbox\par\egroup
  \ST@dimen=\ht\ST@pbox
  \advance\ST@dimen by \dp\ST@pbox
  \ifnum\ST@pboxht<\ST@dimen
    \global\ST@pboxht=\ST@dimen
  \fi
  \ST@dimen=\z@
  \unvbox\ST@pbox\egroup\hfil}
```

`\ST@compute@rowht` The height of a row in an array environemnt can be computed as:

- the height of the strutbox `\ht\strutbox` (plus `\extrarowheight` when the array package is loaded),
- multiplied by arraystretch,
- plus the depth of the strutbox (`\dp\strutbox`) mulitplied by `\arraystretch`.

```
\def\ST@compute@rowht{%
```

Temporarily store a formula with superscript and subscript in a box in order to be able to measure its height and depth.

```
  \setbox\@tempboxa=\vbox{\@arrayparboxrestore $5^5_5$}
```

Use the largest height of either `\@tempboxa` or `\strutbox`.

```
  \ifnum\ht\@tempboxa>\ht\strutbox
    \ST@rowht=\ht\@tempboxa
  \else
    \ST@rowht=\ht\strutbox
  \fi
```

If the array pacakge is used we need to add the value of `\extrarowheight`.

```
  \ifx\extrarowheight\undefined\else
    \advance \ST@rowht by \extrarowheight
  \fi
```

Also use the largest depth of either `\@tempboxa` or `\strutbox`.

```
  \ifnum\dp\@tempboxa>\dp\strutbox
    \advance\ST@rowht \dp\@tempboxa
  \else
    \advance\ST@rowht \dp\strutbox
  \fi
```

And finally multiply by `\arraystretch`.

```
  \ST@rowht = \arraystretch\ST@rowht
  \ST@trace\tw@{Normal Row height: \the\ST@rowht}%
  }
```

`\ST@estimate@rowht` Estimates the height of normal line taking `\arraystretch` into account. Also computes the difference between a normal line and a 'stretched' one.

```
\def\ST@estimate@rowht{%
  \ST@rowht=\arraystretch \baslineskp
  \global\advance\ST@rowht by 1\p@
  \ST@trace\tw@{Average Row height: \the\ST@rowht}%
}
```

`\@calfirstpageht` Estimates the space left on the current page and decides whether the tabular can be started on this page or on a new page.

```
\def\@calfirstpageht{%
  \ST@trace\tw@{Calculating height of tabular on first page}%
```

The TEX register `\pagetotal` contains the height of the page sofar, the LATEX register `\@colroom` contains the height of the column.

```
  \global\ST@pagesofar\pagetotal
  \global\ST@pageleft\@colroom
  \ST@trace\tw@{Height of text = \the\pagetotal; \MessageBreak
                Height of page = \the\ST@pageleft}%
```

When we are in twocolumn mode TEX may still be collecting material for the first column although there seems to be no space left. In this case we have to check against two times `\ST@pageleft`.

```
  \if@twocolumn
    \ST@trace\tw@{two column mode}%
    \if@firstcolumn
     \ST@trace\tw@{First column}%
      \ifnum\ST@pagesofar > \ST@pageleft
        \global\ST@pageleft=2\ST@pageleft
        \ifnum\ST@pagesofar > \ST@pageleft
          \newpage\@calnextpageht
          \ST@trace\tw@{starting new page}%
        \else
```

In this case we're in the second column, so we have to compensate for the material in the first column.

```
          \ST@trace\tw@{Second column}%
          \global\advance\ST@pageleft -\ST@pagesofar
          \global\advance\ST@pageleft -\@colroom
        \fi
```

When `\ST@pagesofar` is smaller than `\ST@pageleft` TEX is still collecting material for the first column, so we can start a new tabular environment like we do on a single column page.

```
      \else
        \global\advance\ST@pageleft by -\ST@pagesofar
        \global\ST@pagesofar\z@
      \fi
    \else
```

When we end up here, TEX has already decided it had enough material for the first column and is building the second column.

```
      \ST@trace\tw@{Second column}
      \ifnum\ST@pagesofar > \ST@pageleft
        \ST@trace\tw@{starting new page}%
```

```
        \newpage\@calnextpageht
      \else
        \global\advance\ST@pageleft by -\ST@pagesofar
        \global\ST@pagesofar\z@
      \fi
    \fi
  \else
```

In one column mode there is a simple decision.

```
    \ST@trace\tw@{one column mode}%
    \ifnum\ST@pagesofar > \ST@pageleft
      \ST@trace\tw@{starting new page}%
      \newpage\@calnextpageht
```

When we are not starting a new page subtract the size of the material already on it from the available space.

```
    \else
      \global\advance\ST@pageleft by -\ST@pagesofar
      \global\ST@pagesofar\z@
    \fi
  \fi
```

When a caption preceeds the first part of the tabular we need to reduce the available height on the page by `\ST@captionroom`.

```
  \if@topcaption\advance\ST@pageleft-\ST@captionroom\fi
  \ST@trace\tw@{Available height: \the\ST@pageleft}%
```

Now we need to know the height of the head of the table. In order to measure this we typeset it in a normal tabular environment.

```
  \ifx\@@tablehead\@empty
    \ST@headht=\z@
  \else
    \setbox\@tempboxa=\vbox{\@arrayparboxrestore
      \ST@restore
      \expandafter\tabular\expandafter{\ST@tableformat}%
      \@@tablehead\endtabular}%
    \ST@headht=\ht\@tempboxa\advance\ST@headht\dp\@tempboxa
  \fi
  \ST@trace\tw@{Height of head: \the\ST@headht}%
```

To decide when to start a new page, we need to know the vertical size of the tail of the table.

```
  \ifx\@tabletail\@empty
    \ST@tailht=\z@
  \else
    \setbox\@tempboxa=\vbox{\@arrayparboxrestore
      \ST@restore
      \expandafter\tabular\expandafter{\ST@tableformat}
        \@tabletail\endtabular}
    \ST@tailht=\ht\@tempboxa\advance\ST@tailht\dp\@tempboxa
  \fi
```

We add the average height of a row to this because when we decide to continue the tabular we need to have enough space left for one row *and* the tail.

```
  \advance\ST@tailht by \ST@rowht
  \ST@trace\tw@{Height of tail: \the\ST@tailht}%
```

```
  \ST@trace\tw@{Maximum height of tabular: \the\ST@pageleft}%
  \@tempdima\ST@headht
```

Now we decide whether we can continue on the current page or whether we need to start on a new page. We assume that the minimum height of a tabular is the height of the head, the tail and one row of data. If that doesn't fit a new page is started.

```
  \advance\@tempdima\ST@rowht
  \advance\@tempdima\ST@tailht
  \ST@trace\tw@{Minimum height of tabular: \the\@tempdima}%
  \ifnum\@tempdima>\ST@pageleft
    \ST@trace\tw@{starting new page}%
    \newpage\@calnextpageht
  \fi
```

Take the height of the table into account,sosubstract it from the available height. We need to do it like this because the `\\` inside the definition of `\@@tablehead` have their normal definition.

```
  \advance\ST@pageleft-\ST@headht
}
```

`\@calnextpageht` This calculates the maximum height of the tabular on all subsequent pages of the supertabular environment.

```
\def\@calnextpageht{%
  \ST@trace\tw@{Calculating height of tabular on next page}%
  \global\ST@pageleft\@colroom
  \global\ST@pagesofar=\z@
  \ST@trace\tw@{Maximum height of tabular: \the\ST@pageleft}%
```

Take the height of the head into account by subtracting it from the avalable space.

```
  \advance\ST@pageleft-\ST@headht
  }
```

`\x@supertabular` The body of the beginning of both environments is stored in a single macro as the code is shared.

```
\def\x@supertabular{%
```

First save the original definition of `\tabular` and then make it point to `\inner@tabular`. This is done to enable supertabular cells to contain a tabular environment without getting unexpected results when the supertabular would be split accross this inner tabular environment.

```
  \let\org@tabular\tabular
  \let\tabular\inner@tabular
```

The same needs to be done for the tabular* environment. The coding is slightly more verbose.

```
  \expandafter\let
    \csname org@tabular*\expandafter\endcsname
    \csname tabular*\endcsname
  \expandafter\let\csname tabular*\expandafter\endcsname
    \csname inner@tabular*\endcsname
```

If the caption should come at the top we insert it here.

```
  \if@topcaption \@process@tablecaption \fi
```

Save the original definition of \\.

```
\global\let\@oldcr=\\
```

Save the current value of \baselineskip, as we need it in the calculation of the average height of a row.

```
\def\baslineskp{\baselineskip}%
```

We have to check whether array.sty was loaded, because some of the internal macros have different names.

```
\ifx\undefined\@classix
```

Save old \@tabularcr and insert the definition of \ST@tabularcr.

```
  \let\org@tabularcr\@tabularcr
  \let\@tabularcr\ST@tabularcr
```

Activate the new parbox algorithm.

```
  \let\org@startpbox=\@startpbox
  \let\org@endpbox=\@endpbox
  \let\@@startpbox=\ST@startpbox
  \let\@@endpbox=\ST@endpbox
\else
```

When array.sty was loaded things are a bit different.

```
  \let\org@tabularcr\@arraycr
  \let\@arraycr\ST@tabularcr
```

The macro \@startpbox needs to be treated carefully for array as it has been renamed to \@startpbox@action in version 2.6i.

```
  \ifx\undefined\@startpbox@action
    \let\org@startpbox=\@startpbox
    \let\@startpbox=\ST@astartpbox
  \else
    \let\org@startpbox@action=\@startpbox@action
    \let\@startpbox@action=\ST@astartpbox
  \fi
  \let\org@endpbox=\@endpbox
  \let\@endpbox=\ST@aendpbox
\fi
```

Check if the head of the table should be different for the first and subsequent pages.

```
\ifx\@table@first@head\undefined
  \let\@@tablehead=\@tablehead
\else
  \let\@@tablehead=\@table@first@head
\fi
```

The first part of a supertabular may be moved on to the next page if it doesn't fit on the current page after all. Subsequent parts can not be moved; therefor we will have to switch the definition of \ST@skippage around.

```
\let\ST@skippage\ST@skipfirstpart
```

Now we can estimate the average row height and the height of the first page of the supertabular.

```
\ST@calculate@rowht
\@calfirstpageht
\noindent
}
```

\supertabular We start by looking for an optional argument, which will be duly ignored as it seems to make no sense to try to align a multipage table in the middle...

```
\def\supertabular{%
  \@ifnextchar[{\@supertabular}%]
               {\@supertabular[]}}
```

We can now save the preamble of the tabular in a macro.

```
\def\@supertabular[#1]#2{%
  \def\ST@tableformat{#2}%
  \ST@trace\tw@{Starting a new supertabular}%
```

Then remember that this is not a supertabular* environment.

```
  \global\ST@starfalse
```

Don't use minipages.

```
  \global\ST@mpfalse
```

Most of the following code is shared between the supertabular and supertabular* environments. So to avoid duplication it is stored in a macro.

```
  \x@supertabular
```

Finally start a normal tabular environment.

```
  \expandafter\org@tabular\expandafter{\ST@tableformat}%
  \@@tablehead}
```

\supertabular* We start by looking for the optional argument of the tabular environment.

```
\@namedef{supertabular*}#1{%
  \@ifnextchar[{\@nameuse{@supertabular*}{#1}}%
               {\@nameuse{@supertabular*}{#1}[]}%]
  }
```

We start by saving the intended width and the preamble of the tabular*.

```
\@namedef{@supertabular*}#1[#2]#3{%
  \ST@trace\tw@{Starting a new supertabular*}%
  \def\ST@tableformat{#3}%
  \ST@wd=#1\relax
  \global\ST@startrue
  \global\ST@mpfalse
```

Now we can call the common code for both environments.

```
  \x@supertabular
```

And we can start a normal tabular* environment.

```
  \expandafter\csname org@tabular*\expandafter\endcsname
  \expandafter{\expandafter\ST@wd\expandafter}%
  \expandafter{\ST@tableformat}%
  \@@tablehead}%
```

\mpsupertabular This version of the supertabular environment puts each tabular into a minipage, thus making footnotes possible. We start by looking for an optional argument, which will be duly ignored as it seems to make no sense to try to align a multipage table in the middle...

```
\def\mpsupertabular{%
  \@ifnextchar[{\@mpsupertabular}%]
               {\@mpsupertabular[]}}
```

We can now save the preamble of the tabular in a macro.

```
\def\@mpsupertabular[#1]#2{%
  \def\ST@tableformat{#2}%
  \ST@trace\tw@{Starting a new mpsupertabular}%
```

Then remember that this is not a mpsupertabular* environment.

```
  \global\ST@starfalse
```

And remember to close the minipage later.

```
  \global\ST@mptrue
```

Since we are about to start a minipage of `\columnwidth` the horizontal alignment will no longer work. We have to remember the values and restore them inside the minipage.

```
  \ST@rightskip \rightskip
  \ST@leftskip \leftskip
  \ST@parfillskip \parfillskip
```

Most of the following code is shared between the mpsupertabular and mpsupertabular* environments. So to avoid duplication it is stored in a macro.

```
  \x@supertabular
```

Finally start a normal tabular environment.

```
  \minipage{\columnwidth}%
  \parfillskip\ST@parfillskip
  \rightskip \ST@rightskip
  \leftskip \ST@leftskip
  \noindent\expandafter\org@tabular\expandafter{\ST@tableformat}%
  \@@tablehead}
```

`\mpsupertabular*` We start by looking for the optional argument of the tabular environment.

```
\@namedef{mpsupertabular*}#1{%
  \@ifnextchar[{\@nameuse{@mpsupertabular*}{#1}}%
               {\@nameuse{@mpsupertabular*}{#1}[]}%]
  }
```

Now we can save the intended width and the preamble of the tabular*.

```
\@namedef{@mpsupertabular*}#1[#2]#3{%
  \ST@trace\tw@{Starting a new mpsupertabular*}%
  \def\ST@tableformat{#3}%
  \ST@wd=#1\relax
  \global\ST@startrue
  \global\ST@mptrue
  \ST@rightskip \rightskip
  \ST@leftskip \leftskip
  \ST@parfillskip \parfillskip
```

Then we can call the common code for both environments.

```
  \x@supertabular
%    And we can start a normal \textsf{tabular*} environment.
%    \begin{macrocode}
  \minipage{\columnwidth}%
  \parfillskip\ST@parfillskip
  \rightskip \ST@rightskip
  \leftskip \ST@leftskip
  \noindent\expandafter\csname org@tabular*\expandafter\endcsname
```

```
  \expandafter{\expandafter\ST@wd\expandafter}%
  \expandafter{\ST@tableformat}%
  \@@tablehead}%
```

`\endsupertabular` `\endsupertabular*` This closes the environments supertabular and supertabular*.

```
\def\endsupertabular{%
  \ifx\@table@last@tail\undefined
    \@tabletail
  \else
    \@table@last@tail
  \fi
  \csname endtabular\ifST@star*\fi\endcsname
```

Restore the original definition of `\@tabularcr`

```
  \ST@restore
```

Check if we have to insert a caption and restore to default behaviour of putting captions at the top.

```
  \if@topcaption
  \else
    \@process@tablecaption
    \@topcaptiontrue
  \fi
```

Restore the meaning of `\\` to the one it had before the start of this environment. Also re-initialize some control-sequences

```
  \global\let\\\@oldcr
  \global\let\@process@tablecaption\relax
  \ST@trace\tw@{Ended a supertabular\ifST@star*\fi}%
  }
```

The definition of the ending of the supertabular* environment is simple:

```
\expandafter\let\csname endsupertabular*\endcsname\endsupertabular
```

`\endmpsupertabular` `\endmpsupertabular*` This closes the environments mpsupertabular and mpsupertabular*.

```
\def\endmpsupertabular{%
  \ifx\@table@last@tail\undefined
    \@tabletail
  \else
    \@table@last@tail
  \fi
  \csname endtabular\ifST@star*\fi\endcsname
  \endminipage
```

Restore the original definition of `\@tabularcr`

```
  \ST@restore
```

Check if we have to insert a caption and restore to default behaviour of putting captions at the top.

```
  \if@topcaption
  \else
    \@process@tablecaption
    \@topcaptiontrue
  \fi
```

Restore the meaning of \\ to the one it had before the start of this environment. Also re-initialize some control-sequences

```
\global\let\\\@oldcr
\global\let\@process@tablecaption\relax
\ST@trace\tw@{Ended a mpsupertabular\ifST@star*\fi}%
}
```

The definition of the ending of the supertabular* environment is simple:

```
\expandafter\let\csname endmpsupertabular*\endcsname\endmpsupertabular
```

`\ST@restore` This macro restores the original definitions of the macros that handle parbox entries and the macros that handle the end of the row.

```
\def\ST@restore{%
  \ifx\undefined\@classix
    \let\@tabularcr\org@tabularcr
    \let\@startpbox\org@startpbox
  \else
    \let\@arraycr\org@tabularcr
    \ifx\undefined\@startpbox@action
      \let\@startpbox\org@startpbox
    \else
      \let\@startpbox@action\org@startpbox@action
    \fi
  \fi
  \let\@endpbox\org@endpbox
  }
```

`\inner@tabular` `\inner@tabular*` In order to facilitate complete tabular environments to be in a cell of a supertabular environment we need to adapt the definition of the orginal environments somewhat. For the inner tabular a number of definitions need to be restored.

```
\def\inner@tabular{%
  \ST@restore
  \let\\\@oldcr
  \noindent
  \org@tabular}
\@namedef{inner@tabular*}{%
  \ST@restore
  \let\\\@oldcr
  \noindent
  \csname org@tabular*\endcsname}
```

`\ST@cr` This macro is called by each \\ inside the tabular environment. It updates the estimate of the amount of space left on the current page and starts a new page if necessary.

```
\def\ST@cr{%
  \noalign{%
    \ifnum\ST@pboxht<\ST@rowht
```

If there is a non-empty row, but an empty parbox, then `\ST@pboxht` might be non-zero, but too small, thereby breaking the algorithm. Therefor we estimate the height of the row to be `\ST@rowht` in this case.

```
      \global\advance\ST@pageleft -\ST@rowht
```

And we store that fact in \ST@prevht.

```
      \global\ST@prevht\ST@rowht
    \else
```

When the parbox was not empty we take into account its height (plus a bit extra).

```
     \ST@trace@cr\thr@@{Added par box with height \the\ST@pboxht}%
      \global\advance\ST@pageleft -\ST@pboxht
      \global\advance\ST@pageleft -0.1\ST@pboxht
      \global\ST@prevht\ST@pboxht
      \global\ST@pboxht\z@
    \fi
```

\ST@toadd is the value of the optional argument of \\.

```
    \global\advance\ST@pageleft -\ST@toadd
    \global\ST@toadd=\z@
    \ST@trace@cr\thr@@{Space left for tabular: \the\ST@pageleft}%
  }
```

This line is necessary because the tablehead has to be inserted *after* the following \if\else\fi-clause. For this purpose \ST@next is used by \ST@newpage. But we need to make sure that \ST@next is not undefined when \ST@newpage is *not* called. In the middle of tableprocessing it shoud be an *empty* macro (*not* \relax). (15.2.91)

```
  \noalign{\global\let\ST@next\@empty}%
```

When the \ST@pageleft has become negative, the last row was so high that the supertabular doesn't fit on the current page after all. In this case we will skip the current page and start at the top of the next one; otherwise TEX will move this part of the table to a new page anyway, probably with a message about an overfull \vbox.

```
  \ifnum\ST@pageleft<\z@
    \ST@skippage
  \else
```

When there is not enough space left on the current page, we start a new page. To compute the amount of space needed we use the height of the previous row (\ST@prevht) as an estimation of the height of the next row. If we are processing a mpsupertabular we need to take the height of the footnotes into account as well.

```
    \noalign{\global\@tempdima\ST@tailht
      \global\advance\@tempdima\ST@prevht
    \ifST@mp
      \ifvoid\@mpfootins\else
        \global\advance\@tempdima\ht\@mpfootins
        \global\advance\@tempdima 3pt
      \fi
    \fi}
    \ifnum\ST@pageleft<\@tempdima
      \ST@newpage
    \fi
  \fi
  \ST@next}
```

\ST@skipfirstpart This macro skips the current page and moves the entire supertabular that has been built up sofar to the next page.

```
\def\ST@skipfirstpart{%
```

```
\noalign{%
  \ST@trace\tw@{Tabular too high, moving to next page}%
```

In order for this to work properly we need to adapt the value of `\ST@pageleft`. When this macro is called it has a negative value. We should add the height of the next page to that (`\@colroom`). From the result the 'normal' height of the supertabular should be substracted (`\@colroom - \pagetotal`). This could be coded as follows:

```
\ST@dimen\@colroom
\advance\ST@dimen-\pagetotal
\global\advance\ST@pageleft\@colroom
\global\advance\ST@pageleft-\ST@dimen
```

When you examine the code you will note that `\@colroom` is added *and* subtracted. Therefor the code above can be simplified to:

```
  \global\advance\ST@pageleft\pagetotal
```

Then we can set `\ST@pagesofar` to 0...

```
  \global\ST@pagesofar\z@
```

... and start the new page, but in this special case we need to trigger the output routine directly by issuing a `\penalty` rather than calling `\newpage` as that macro effectively issues a `\vskip-\maxdepth` which makes the first two rows of the tabular overlap.

```
  \penalty -\@M
```

Finally we make sure that this macro can only be executed once for each supertabular by changing the definition of `\ST@skippage`.

```
  \global\let\ST@skippage\ST@newpage
  }}
```

`\ST@newpage` This macro performs the actions necessary to start a new page.

```
\def\ST@newpage{%
  \noalign{\ST@trace\tw@{Starting new page, writing tail}}%
```

Output `\tabletail`, close the tabular environment, close a mnipage if necessary, output all material and start a fresh new page.

```
  \@tabletail
  \ifST@star
    \csname endtabular*\endcsname
  \else
    \endtabular
  \fi
  \ifST@mp
    \endminipage
  \fi
```

Then we make sure that the macro `\ST@skippage` can no longer be executed for this supertabular by changing the definition of it.

```
  \global\let\ST@skippage\ST@newpage
  \newpage\@calnextpageht
  \let\ST@next\@tablehead
  \ST@trace\tw@{writing head}%
  \ifST@mp
    \noindent\minipage{\columnwidth}%
```

```
    \parfillskip\ST@parfillskip
    \rightskip \ST@rightskip
    \leftskip \ST@leftskip
  \fi
  \noindent
  \ifST@star
    \expandafter\csname org@tabular*\expandafter\endcsname
    \expandafter{\expandafter\ST@wd\expandafter}%
    \expandafter{\ST@tableformat}%
  \else
    \expandafter\org@tabular\expandafter{\ST@tableformat}%
  \fi}
```

⟨/package⟩