Abstract

This package provides spreadsheet features for \LaTeX table environments.

The main feature allows the user to construct tables in a manner similar to a spreadsheet where cells are used in formulas to generate values in other cells. The package computes the formulas in the correct order and finally displays the table with the numeric results.
## Contents

1 **Introduction** ........................................... 2  
   1.1 Presentation ........................................ 2  
   1.2 Motivation .......................................... 3  

2 **Basic features** ........................................ 3  
   2.1 Absolute references ................................ 3  
   2.2 Relative references ................................ 4  
   2.3 Text cells ........................................... 4  
   2.4 Mixed cells ......................................... 5  
   2.5 Copy a formula ...................................... 5  

3 **Features for formatting the table** ..................... 7  
   3.1 Decimal separator ................................... 7  
   3.2 Number formatting and the \texttt{fp} package ........ 8  
   3.3 End of lines and horizontal rules ................... 9  
   3.4 Hide a row or column ................................ 10  
   3.5 Save the result of a cell ............................ 11  
   3.6 Display the value of a cell .......................... 11  
   3.7 The use of \texttt{\multicolumn} ..................... 12  

4 **Macro-functions** ........................................ 13  
   4.1 Mathematical macro-functions ....................... 13  
      4.1.1 Sum cells ....................................... 13  
      4.1.2 The \texttt{fact} macro .......................... 13  
      4.1.3 The \texttt{sumprod} macro ...................... 14  
      4.1.4 Random numbers ................................ 14  
      4.1.5 GCD and LCM ................................... 15  
      4.1.6 Scientific notation ............................. 15  
      4.1.7 Identity ....................................... 16  
   4.2 Tests ................................................ 16  
   4.3 Macro-functions manipulating dates ................ 17  
      4.3.1 Date to number with \texttt{engshortdatetonum} . . 17  
      4.3.2 From a number to a date ....................... 18  
   4.4 Coordinate macro functions ........................ 19  

5 **Particular care** ........................................ 20  
   5.1 Defining new commands with \texttt{\hline} ............. 20  
   5.2 The use of \texttt{\multicolumn} and \texttt{\Sthidecol} ....... 21  
   5.3 Messages delivered by \texttt{spreadtab} ............... 22  
   5.4 Debug mode ......................................... 23  

6 **Examples** .............................................. 24  
   6.1 Pascal’s triangle again! ................................ 24  
   6.2 The convergence of a series .......................... 24  
   6.3 Convergence to the golden ratio ..................... 25  
   6.4 A billing table ..................................... 26  
   6.5 A magic square ..................................... 26  
   6.6 A pyramid of additions ................................ 27
1 Introduction

1.1 Presentation

This package allows us to construct tables in a manner similar to a spreadsheet. The cells of a table have row and column indices and these can be used in formulas to generate values in other cells. The package is loaded with the classic syntax in the preamble:

```
\usepackage[<options>]{spreadtab}
```

where the options "fp" or "xfp" select the calculation engine. If no option is specified, the engine will be fp. If both options are specified, the active engine at the beginning of the document will be xfp, but in this case, you can change the engine at any time by running \STusefp or \STusexfp. This documentation is compiled with the engine xfp, unless examples where fp engine is selected. As a general rule, the two calculation engines must give identical results, but

- **fp** is a fixed point calculation engine (18 digits before and 18 after) and therefore gives more digits after the decimal point than xfp which performs floating point calculations with 16 significant digits; on the other hand, xfp natively allows tests, manages infinity and NaN, understands implicit multiplication by juxtaposition, etc.
- differences exist between the two engines regarding the functions available both in terms of syntax and their existence. Indeed, the function max(...) only admits two numbers as arguments with fp while xfp accepts as many arguments as you want. Similarly, the exponential is obtained with e^x or pow(x,e) with fp while the syntax with xfp is exp(x). Moreover, the functions recognized by the two engines are not the same, as can be seen from their documentation: ceil for example, only exists for xfp. In general, we can consider xfp to have more features than fp;
- the functions rand and randint are not available with the engine xfp when compiling with an old version of XeLaTeX.

The package requires \LaTeX\, L\ script\,2\,\LaTeX\ format and the fp or xfp packages, which performs arithmetic on cell values. Also, the xstring package is needed.

The package is compatible with all tabular environments, and assumes that ‘&’ is used to delimit columns and ‘\’ to end lines (but see page 3). This compatibility requirement led me to program spreadtab so that it works independently of the table environment. Thus, reading the table, processing and calculating the formulas is done before the environment name 'sees' the body of the table.

Consequently, spreadtab proceeds in 3 main stages before \begin{<table environment>} sees the table:

- first, it reads the body of the table, divides it in lines and cells, and in each cell, seeks a possible formula;
- then, it computes the formulas in the cells, taking care for each to previously calculate all the dependent cells. The calculations are made by the package fp or xfp, provided that the formulas to be evaluated in the cells are compatible with the syntaxes of the macros \FPeval of the package fp or \fpeval of the package xfp;
- finally, it is necessary to rebuild the table, replacing each formula by its numerical calculated value and handing over to the environment name specified by the user.

The syntax in both the following is allowed (and equivalent), where \texttt{<name>} is the name of any table environment available with \LaTeX\ or with a package:

\begin{verbatim}
\begin{spreadtab}{{<name>}{<parameters>}}
table with formulas and numbers
\end{spreadtab}
\end{verbatim}

After the work of spreadtab, we get a display as if we had written:

\begin{verbatim}
\begin{spreadtab}{{<name>}{<parameters>}}
table with formulas and numbers
\end{spreadtab}
\end{verbatim}

\textsuperscript{1}If you want to use the logical ‘AND’ of xfp ‘&&’, it is mandatory to put this operator between brackets in an array so that the tokens && are not understood as column separators. In a cell of a table, we will write for example a1>1 \{&&\} a1<10.
Although having features resembling those of a spreadsheet with \LaTeX is appreciable, the 3 stages described above take time. The spreadtab environment leads to much slower compilation than with a classical table.

Moreover, spreadtab cannot stand in for a spreadsheet program. Indeed, it has very few features, and it does not provide visual assistance. This point may cause difficulty\(^2\) for big or complex tables. The syntax of spreadtab is also another difficulty. However, the advantage of this package is that it makes it possible to write \textit{in the \LaTeX code} tables involving calculation when these tables are usually exported\(^3\) from a spreadsheet program to \LaTeX code. Consequently, it becomes possible to avoid the disadvantages of the exportation programs: fine tuning often necessary to obtain exactly what you want, exported tables containing the values only (formulas are lost when exportation is done), no compatibility with all types of environments, exportation must be started again if a single number or formula is modified in the table.

### 1.2 Motivation

A few months before I started to write this package, Derek O’CONNOR had pointed out that nothing was available in the world of \LaTeX packages to imitate – even a little – the behaviour of spreadsheet programs. I found the challenge interesting and I started writing this package as a good programming exercise.

The road was long before reaching this version and I especially want to thank Christophe CASSEAU for his early interest and for the suggestions he made, and more recently Derek O’CONNOR for his advice and for the constructive discussions we have had. I also thank Andrew PARSLOE for proofreading the english translation of this manual.

### 2 Basic features

A table is a rectangular array of cells which may be viewed as a set of cells arranged in horizontal rows or vertical columns.

By default, spreadtab expects a ‘\’ at the end of lines, which is usual in tables. This end of line marker can be changed via the \texttt{\STeol{<macro>}} command. We can write for example \texttt{\STeol{\tabularnewline}}. It is important to remember that the ends of lines to be inserted in the final table will always be ‘\’ even if the end of line marker that spreadtab sees when it \textit{reads} the table is different.

#### 2.1 Absolute references

A table cell is identified by the pair \texttt{<colref><rowref>}\(^4\), where:

- \texttt{<colref>} is a letter from a to z, and a \textit{is} the first column on the left: it is limited to 26 columns, which should be sufficient for the majority of cases; the letter can be upper or lowercase;
- \texttt{<rowref>} is a positive integer representing row number. The row number 1 is the top row.

Here are examples of absolutes references: b4 or C1 or d13. Locations of cells appear clearly in the spreadsheet-like table below:

\[
\begin{array}{|c|c|c|c|c|}
\hline
& A & B & C & D \\
1 & & & & \\
2 & & & & \\
3 & & & & \\
4 & & & & \\
5 & & & & \\
\hline
\end{array}
\]

---

\(^2\)I certify that, with use, this discomfort tends to disappear (if you do not work with huge tables, of course).

\(^3\)I mention the two main exportation programs: \texttt{calc2latex} for ‘calc’ (Open Office), and \texttt{excel2latex} for ‘excel’ (Microsoft Office).

\(^4\)Note: this is the opposite to the standard matrix convention.
This example calculates the sum of each row and each column and then calculates the grand total:

\begin{spreadtab}{rrr}
22 & 54 & a1+b1 \\
43 & 65 & a2+b2 \\
49 & 37 & a3+b3 \\
\hline
a1+a2+a3 & b1+b2+b3 & a4+b4
\end{spreadtab}

For people familiar with maths, this other example calculates the first lines of Pascal’s triangle:

\begin{spreadtab}{cccccc}
1 & & & & & \\
a1 & a1 & & & & \\
a2 & a2+b2 & b2 & & & \\
a3 & a3+b3 & b3+c3 & c3 & & \\
a2 & a4+b4 & b4+c4 & c4+d4 & d4 & \\
\end{spreadtab}

\begin{spreadtab}{ccc}
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
\end{spreadtab}

### 2.2 Relative references

To refer to a cell, it may be convenient to specify its position relative to where the formula is written. Thus, the relative coordinates of a cell are 2 relative numbers written using this syntax: \([x,y]\) where \(x\) is the horizontal offset from the cell containing the formula and \(y\) is the vertical offset. For example, \([-2,3]\) refers to the cell located 2 columns before (on the left) and 3 rows after (below) the cell where the formula is located.

Here is the same table as above but the references are relatives and the \texttt{matrix} environment of the \texttt{amsmath} package is used:

\begin{spreadtab}{matrix}
[0,-1] & [-1,-1] \\
[0,-1] & [-1,-1]+[0,-1] & [-1,-1] \\
[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1] \\
[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]
\end{spreadtab}

We note that relative references are more appropriate here, since only 2 different references are used: \([0,-1]\) which refers to the cell above and \([-1,-1]\) which refers to the cell located to the NW of the current cell.

Absolute and relative references can be mixed in a formula.

### 2.3 Text cells

If you want to put only text in a cell, you must tell spreadtab that the cell should not be calculated. Simply place somewhere in the cell the character ‘@‘ with its usual catcode 12. The cell will be ignored by spreadtab which will consider it as an inert cell impossible to reference elsewhere in the table.

Example:

\begin{spreadtab}{|r|ccc|}
\hline
\@ values of 5x & -5 & -1 & 4 \\
\@ \text{If}(x)=2x & 2v[0,-1] & 2v[0,-1] & 2v[0,-1] \\
\hline
\end{spreadtab}

\footnote{There is an exception, see page 17.}
The control sequence \texttt{\STtextcell} expands to the character ‘@’. It is possible to redefine it; for example, after \texttt{\renewcommand\STtextcell{\textquoteleft}}, a cell containing the char ‘ will be understood as a text cell.

Moreover, if a cell is empty or filled with spaces, \spreadtab will consider it as a text cell.

### 2.4 Mixed cells

In reality, each cell is composed of two fields. The first is a numeric field containing the formula; the second is a text field, ignored by the calc engine and not taken into account for calculations:

- if nothing is specified in a cell, the entire cell is the number field, and the text field is empty (this was the case for all table cells of Pascal’s triangle seen above);
- if the cell contains the ‘@’ character, then the entire cell is the text field. The numeric field is empty and inaccessible;
- if the cell contains the marker ‘:=’, then the following argument between braces is the numeric field, and everything else is the text field. The cell has this structure:

\[
<\text{text field}>:=<\text{numeric field}> <\text{end of text field}>
\]

The marker ‘:=’ is the expansion of the control sequence \texttt{\STnumericfieldmarker}. It is possible to redefine it, for example:

\texttt{\renewcommand\STnumericfieldmarker{\=}}

In this case, the expansion of the marker ‘\=’ would have no importance and would not be involved in the process. For \spreadtab, it is only a token showing where the formula begins. This token is sought and recognized but is never expanded.

Once the numeric field is computed, ‘:=\{numeric field\}’ is replaced by the numeric value.

Note that ‘:=\{numeric field\}’ may be inside brackets, whatever be the level of nesting. For example, if a cell contains \texttt{\textbf{:=\{(a1+1)\}}} and if the numeric value of the cell a1 is 5, then finally, the cell will contain \texttt{\textbf{6}}

Here is a simple example:

\begin{spreadtab}{{tabular}{|c|c||c|}}
\hline
value 1 : :=\{50\} & value 2 : :=\{(29)\} & average : \textbf{:=\{\((a1+b1)/2\)\}} \\
\hline
\end{spreadtab}

\begin{tabular}{|c|c||c|}
\hline
value 1 : 50 & value 2 : 29 & average : \textbf{39.5} \\
\hline
\end{tabular}

If ‘:=’ is written in a cell with an empty argument like this ‘:=\{}’, then the cell is understood as a text cell. In fact, ‘\{\}’ behaves like ‘@’, but they are not equivalent: ‘:=\{}’ allows the cell to receive a formula from another with \texttt{\STcopy} (see next section) while it is impossible with ‘@’.

### 2.5 Copy a formula

To avoid having to copy formulas into adjacent cells, the \spreadtab package provides the \texttt{\STcopy} command. This command must be written in a cell with this syntax:

\texttt{\STcopy{\(x,\{y\)}\{formula\}}
where \( x \) and \( y \) are positive numbers that represent horizontal and vertical offsets relative to the cell where the command is. With the cell containing the command (the source cell), these offsets define a range of cells which will receive the \(<\text{formula}>^6\). The command \STcopy must not be in a cell where there is a numeric field marker ‘:=’.

Here is how the copy is made: it starts from the cell where the command \STcopy is. For the other cells, all the coordinates in the formula are modified taking into account the offsets from the source cell. For example, if the source cell contains the formula \( a1+b2+c3 \), and the target cell is located 2 columns rightwards and 5 rows below then, this formula becomes: \( c6+d7+e8 \). The formula can also contain relative references but, since they are relative, they are not modified.

Preceded by ‘!’, a coordinate in a formula is not modified when the formula is copied. For example, if the source cell contains \( a!1+b!2+c!3 \) and the target cell is located 2 columns rightwards and 5 rows below then, this formula becomes: \( c1+b7+c3 \). The feature is compatible with relative coordinates. Let’s suppose a cell contains this formula: \((-1,-1)+[(-1,1)+[1,2]]\). As usual, let’s say that this formula is copied to the cell located 2 columns rightwards and 5 rows below: this formula becomes: \((-1,-6)+([-3,1]+[-1,-3])\).

The ‘!’ character is the expansion of the control sequence \STtransposecar. It may be changed to any other with \renewcommand\STtransposecar{<char>}. The ‘!’ character, used by default, keeps its 13 catcode and remains active if the babel package is loaded with the french option.

In \STcopy{[x,y]}{formula}, if \( x \) is omitted, the copy is made to the cells rightwards, up to the right edge of the table. With \( y \), it is the same: if this number is omitted, the copy is done to the cells below until the bottom of the table is reached. If \( x \) or \( y \) is equal to 0, the copy is limited to the column or row of the source cell. Instead of writing \( v0 \) or \( >0 \), it is possible to write \( v \) or \( > \).

Here are some examples:

\begin{verbatim}
{>3,v1}  copy to 3 columns rightwards and 1 row below  
{>3}    copy to 3 cells on the right  
{v1}    copy to the cell below  
{>}    copy rightwards up to the right edge  
{v}    copy below  
{v,>}  copy to the right and below until the end of the table
\end{verbatim}

It is easy to generate the multiplication table from 1 to 10:

\begin{spreadtab}
\begin{tabular}{|c|*{10}{c}|}
\hline
\times & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
2 & 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 18 & 20 \\
3 & 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 & 27 & 30 \\
4 & 4 & 8 & 12 & 16 & 20 & 24 & 28 & 32 & 36 & 40 \\
5 & 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 & 50 \\
6 & 6 & 12 & 18 & 24 & 30 & 36 & 42 & 48 & 54 & 60 \\
7 & 7 & 14 & 21 & 28 & 35 & 42 & 49 & 56 & 63 & 70 \\
8 & 8 & 16 & 24 & 32 & 40 & 48 & 56 & 64 & 72 & 80 \\
9 & 9 & 18 & 27 & 36 & 45 & 54 & 63 & 72 & 81 & 90 \\
10 & 10 & 20 & 30 & 40 & 50 & 60 & 70 & 80 & 90 & 100 \\
\hline
\end{tabular}
\end{spreadtab}

\footnote{The copy can only be done to cells located on the right and below the cell containing the macro.}
3 FEATURES FOR FORMATTING THE TABLE

If the numeric field of a target cell is not empty, it is not replaced and the copy is not done for this cell.

If 2 or more \STcopy commands in several source cells have the same target cell, then, the formula this latter receives is the one contained in the last \STcopy command when the table is read from top left to bottom right.

In the spreadsheet-like example below, the \STcopy in the pink cell B1 has its target range partially covered by the one of the green cell C3.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>\STcopy(v,&gt;{a1+1})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td>\STcopy(&gt;{2,v1}&gt;{a3+10})</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Here is this example, treated by spreadtab below. In this table, the cell b5 (numeric field alone) and the cell c5 (text field + numeric field) stay unchanged since their numeric fields are not empty:

```
\begin{spreadtab}{{tabular}{|*6{c|}}}
\hline
1 & \STcopy(v,>{a1+1}) & & & & \\
2 & & & & & \\
3 & \STcopy(>{2,v1}>{a3+10}) & & & & \\
4 & & & & & \\
5 & \textbf{-1} & \textbf{a0b} & & & \\
\hline
\end{spreadtab}
```

As mentioned in the last chapter, you can also copy a formula to a text cell containing an empty numeric field (that is to say a cell containing `:=('{}). In this case, the formula is copied inside the brackets. On the other hand, a cell containing text `@` cannot receive a formula when copying and the cell remains purely textual.

Example :

```
\begin{spreadtab}{{tabular}{|*6{c|}}}
\hline
1 & 2 & 3 & 4 & 5 & 6 \\
2 & 2 & 3 & 4 & 5 & 6 \\
3 & \textbf{X} & \textbf{Y} & \textbf{X} & \textbf{Y} & \textbf{X} \\
4 & \textbf{X} & \textbf{Y} & \textbf{X} & \textbf{Y} & \textbf{X} \\
5 & \textbf{X} & \textbf{Y} & \textbf{X} & \textbf{Y} & \textbf{X} \\
\hline
\end{spreadtab}
```

3 Features for formatting the table

3.1 Decimal separator

The \texttt{fp} and \texttt{xfp} packages return results with the decimal point separator. After the job of calculation, spreadtab can change this decimal separator: everything happens as if the results returned by \texttt{fp} or \texttt{xfp} were taking this into account. The command \texttt{\STsetdecimalsep} takes a mandatory argument which is the character used as decimal separator:

\texttt{\STsetdecimalsep{<char>}}
For example, French users should write this in the preamble of the document:

```latex
\STsetdecimalsep{,}
```

For numeric fields located in math mode, the comma is considered as a math punctuation, which explains why it is followed by a space. To prevent this behaviour, it can be written inside brackets:

```latex
3,14 is not displayed like $3,14$.\par
3,14 is displayed like $3\{,\}14$
```

When cells are in math mode, you can use this feature and ask spreadtab to replace the decimal point by a comma inside braces with the command `\STsetdecimalsep{,}`. In these tables where each cell is in math mode, the space after the commas are neutralized in the second table:

```latex
\begin{spreadtab}{{tabular}{|*3{>{$}r<{$}}|}}
\hline
@x & @y & @\text{Average} \\
\hline
5 & -4 & (a2+b2)/2 \\
-6.1 & -8 & (a3+b3)/2 \\
9.85 & 3.7 & (a4+b4)/2 \\
\hline
\end{spreadtab}
```

3.2 Number formatting and the `fp` package

With `fp` option, all calculations are made by the `\FPeval` macro of the `fp` package. This package provides all necessary arithmetical functions along with various scientific and trigonometric functions. Calculations are made with 18 decimal digits of precision, and `fp` displays all the decimals! Without taking precautions, you can end up with a lot of numbers in the decimal parts of some results. This section is compiled with `\STusefp`.

The number of digits displayed can be controlled in various ways:

- the `numprint` package can be used in order to properly display numbers;
- `fp` can round or truncate numbers with `round(number,integer)` or `trunc(number,integer)` but the syntax makes this tedious to write if this is needed for many cells;
- spreadtab can round all the numbers in the table with the macro `\STautoround` whose argument is number of digits in the decimal part. If the argument is empty, no rounding is done. If the starred macro `\STautoround*` is used, the decimal part is filled with 0 as necessary (only with `fp`).

In this example, floating point numbers are rounded to 6 digits:

```latex
\STautoround{6}
```

It is preferable to use the `numprint` package to format the results. You can also change the math code of the comma: `\mathcode',="013B`\relax. This trick puts the comma in the class 0 of the ordinary signs while its natural class is 6 (punctuation signs).

This macro accepts infix or postfix notation. Consequently, both can be used to write formulas in a cell. For example the infix formula `\texttt{a1+b1}` is equivalent to the postfix ones `\texttt{a1 b1 add}` or `\texttt{a1 b1 +}`.
3.3 End of lines and horizontal rules

spreadtab recognizes the usual line breaks and horizontal rules `\` and `\hline`. It is also possible to specify the optional argument in line break: `\\<dimension>`.

For horizontal rules, it is possible to use:

- `\hline`;
- `\cline{x-y}` where `x` and `y` define the start and the end of the rule;

All the numbers contained in the final table, either typed as is or coming from a calculation are processed by the macro `\STprintnum`. By default, this macro has no effect on its argument and is defined this way:

```
\newcommand{\STprintnum}{#1}
```

It is possible to round all the numbers via the `\numprint` command from the `numprint` package. To achieve this, the `\STprintnum` command must be redefined:

```
\renewcommand{\STprintnum}[1]{\numprint{#1}}
\nprounddigits{6}
\begin{spreadtab}{{tabular}{cccccccc}}
\hline
x & 1 & 2 & 3 & 4 & 5 & 6 &7 \\
\hline
1/x &1/b1&1/c1&1/d1&1/e1&1/f1&1/g1&1/h1
\end{spreadtab}
```

Here is another similar example where we test if the number to display is negative with the command `\FPifneg` from the `fp` package. If this happens, the number is displayed in red. The command `\STautoround` was preferred to `\nprounddigits` from the `numprint` package because the latter adds unnecessary 0’s.

```
\STautoround{6}
\begin{spreadtab}{{tabular}{cccccccc}}
\hline
x & -1 & 2 & -3 & 4 & -5 & 6 & -7 \\
\hline
1/x &-1 &0,5 &-0,333 333 &0,25 &-0,2 &0,166 667 &-0,142 857
\end{spreadtab}
```

The numbers typed as is or coming from a calculation are processed by the macro `\STprintnum`. By default, this macro has no effect on its argument and is defined this way:

```
\newcommand{\STprintnum}{#1}
```

It is possible to round all the numbers via the `\numprint` command from the `numprint` package. To achieve this, the `\STprintnum` command must be redefined:

```
\renewcommand{\STprintnum}[1]{\numprint{#1}}
\nprounddigits{6}
\begin{spreadtab}{{tabular}{cccccccc}}
\hline
x & 1 & 2 & 3 & 4 & 5 & 6 &7 \\
\hline
1/x &1/b1&1/c1&1/d1&1/e1&1/f1&1/g1&1/h1
\end{spreadtab}
```

The numbers typed as is or coming from a calculation are processed by the macro `\STprintnum`. By default, this macro has no effect on its argument and is defined this way:

```
\newcommand{\STprintnum}{#1}
```

It is possible to round all the numbers via the `\numprint` command from the `numprint` package. To achieve this, the `\STprintnum` command must be redefined:

```
\renewcommand{\STprintnum}[1]{\numprint{#1}}
\nprounddigits{6}
\begin{spreadtab}{{tabular}{cccccccc}}
\hline
x & 1 & 2 & 3 & 4 & 5 & 6 &7 \\
\hline
1/x &1/b1&1/c1&1/d1&1/e1&1/f1&1/g1&1/h1
\end{spreadtab}
```

The numbers typed as is or coming from a calculation are processed by the macro `\STprintnum`. By default, this macro has no effect on its argument and is defined this way:

```
\newcommand{\STprintnum}{#1}
```

It is possible to round all the numbers via the `\numprint` command from the `numprint` package. To achieve this, the `\STprintnum` command must be redefined:

```
\renewcommand{\STprintnum}[1]{\numprint{#1}}
\nprounddigits{6}
\begin{spreadtab}{{tabular}{cccccccc}}
\hline
x & 1 & 2 & 3 & 4 & 5 & 6 &7 \\
\hline
1/x &1/b1&1/c1&1/d1&1/e1&1/f1&1/g1&1/h1
\end{spreadtab}
```
3 FEATURES FOR FORMATTING THE TABLE

- \hhline{<type>} where \texttt{<type>} is the type of rule (read the manual of the \texttt{hhline} package).
- any command of the \texttt{booktabs} package: \texttt{\toprule, \midrule, \bottomrule, \cmidrule, \addlinespace, \morecmidrule and \specialrule}. All the arguments of these macros, optional or mandatory are taken into account;
  - \texttt{\noalign} and its mandatory argument can be written after \texttt{\}. Example:

\begin{spreadtab}{{tabular}{*5c}}
\hline
[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]\end{spreadtab}

3.4 Hide a row or column

Sometimes, a column or a row is intended for intermediate calculations and does not need to be displayed in the final table. For this, \texttt{spreadtab} provides two control sequences \texttt{\SThiderow} and \texttt{\SThidecol} which, when placed in a cell, hide the row or column that contains the cell.

An example:

\begin{spreadtab}{{tabular}{|r|ccc|}}
\hline
@ values of $x$ & -1 & 0 & \texttt{\SThidecol} & 2 & 3 \\
\hline
f(x)=2x-1 & 2*[0,-1]-1 & 2*[0,-1]-1 & 2*[0,-1]-1 & 2*[0,-1]-1 \\
\hline
g(x)=x-10 & 0 & 0 & 0 & 0 \\
\hline
h(x)=1-x & 1 & 1 & 1 & 1 \\
\hline
\end{spreadtab}

The row containing $g(x)$ and column corresponding to the value 0 are hidden.

Remember that the hidden rows and columns are invisible to the tabular environment chosen by the user. Thus, only 4 columns have been defined (\texttt{|r|ccc|}) and not 5 as seen by \texttt{spreadtab}.

Just to see the difference, here is the table obtained when setting 5 columns in the preamble and not hiding any row or column:

\begin{spreadtab}{{tabular}{|r|ccc|}}
\hline
@ values of $x$ & -1 & 0 & 2 & 3 \\
\hline
f(x)=2x-1 & -3 & 3 & 5 \\
h(x)=1-x & 2 & -1 & -2 \\
\hline
\end{spreadtab}
3.5 Save the result of a cell

It may be necessary to save the numerical value of a cell to display it outside a formula or even outside the table. Here is how to do it:

\texttt{\textbackslash STsavecell\{control sequence\}\{absolute reference\}}

With a \texttt{\textbackslash global\textbackslash def}\(^9\), this command globally saves in \texttt{control sequence} the result of the formula contained in the cell \texttt{absolute reference}.

Only absolute references can be used since this command must be placed in the optional argument of the \texttt{spreadtab} environment.

Example:

\begin{spreadtab}
\begin{tabular}{|c|c|c|c|c|}
\hline
10 & a1+10 & b1+10 & a1+b1+c1 & \textbackslash cell c1 : \textbackslash result \\
\hline
\end{tabular}
\end{spreadtab}

Here is the cell c1 : 30

In order to save several cells, the command \texttt{\textbackslash STsavecell} can be put several times in the optional argument.

Example:

\begin{spreadtab}
\begin{tabular}{|rc|rc|rc|}
\hline
@Speed (km/h) & \textbackslash SThidecol & \textbackslash SThidecol & \textbackslash SThidecol & 35 \\
\hline
@Time (h min s) & \textbackslash SThidecol & \textbackslash SThidecol & \textbackslash SThidecol & \textbackslash SThidecol & \textbackslash SThidecol & 5 h 8 min 34.2 s \\
\hline
\end{tabular}
\end{spreadtab}

It lasts more than 5 hours.

3.6 Display the value of a cell

In order to display the numeric field of a cell in a textual field, we have seen that we could save this value in a control sequence and use this control sequence anywhere in the table. The process is somewhat tedious… Indeed, \texttt{\textbackslash STsavecell} is not intended to be used in such a way. Its aim is to save the value of a numeric field for further use outside the table.

There is a simpler way to display the numeric field of a cell in a textual field using the syntax \texttt{\textless reference\textgreater} which is replaced by the numeric field of the cell reference, where the reference can be absolute or relative. If the text between \texttt{\textless and \textgreater} is not a reference, then \texttt{\textless text\textgreater} is left as is. The reference must not contain any space; for example, if you write \texttt{\textless a1\textgreater}, spreadtab does not understand it as a reference because of the space before ‘a1’.

Example in a textual cell a3:

\(^9\)The \texttt{\textbackslash def} command does not check if the macro it defines already exists.
\begin{spreadtab}{{tabular}{|c|c||c|}}
\hline
23 & 32 & Average $= \frac{a1+b1}{2} = \frac{23+32}{2} = 27.5$
\hline
\end{spreadtab}

The marker ‘:=’ is necessary in every cell where the command \texttt{\multicolumn} is written. Without it, \texttt{spreadtab} would consider that the whole cell \texttt{(i.e. \multicolumn{2}{|c|}{<formula>})} as the formula, which is impossible to calculate.
4 Macro-functions

The fp package provides a limited set of operations and functions. If these are not sufficient then spreadtab allows the advanced programmer to write macros using the operations and functions of fp. This section presents the macro-functions currently available. There will be more details on how to program macro functions in the next version of this manual.

4.1 Mathematical macro-functions

4.1.1 Sum cells

The macro-function sum sums one or several ranges of cells.

It should be used like this: sum(<range 1>;<range 2>;...;<range n>), where a range of cells is:

- either a single cell like a1 or [2,1];
- or a rectangular area bounded by the upper-left cell and lower-right with this syntax:
  <cell 1>:<cell 2>

Here are some examples of such areas: a2:d5, [-1,-1]:[2,3], b4:[5,1].

In cell ranges, if a cell does not have a numeric field (empty cell or text cell or merged cell with \multicolumn), it is seen as 0 by sum.

In the following table, the sum of the binomial coefficients of Pascal’s triangle is calculated:

\begin{spreadtab}{{tabular}{*5c}}
\\multicolumn{5}{c}{sum: :={sum(a2:e6)}}\\
0,1 & -[1,1]+[0,1] & -[1,1]+[0,1] & -[1,1]+[0,1] & -[1,1] & \\
[0,1] & -[1,1]+[0,1] & -[1,1]+[0,1] & -[1,1] & \\
[0,1] & -[1,1]+[0,1] & -[1,1] & \\
[0,1] & -[1,1] & \\
1 & & & &
\end{spreadtab}

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>24</td>
<td>120</td>
<td>720</td>
<td>5040</td>
<td>40320</td>
</tr>
</tbody>
</table>

4.1.2 The fact macro

The macro-function fact(<number>) computes the factorial of its argument. With the engine fp, the argument must be less than 19 to avoid overflows. The <number> can also be a reference to a cell whose numeric field contains an integer.

Here are the factorials from 0 to 8:

\begin{spreadtab}{{tabular}{*9c}}
\multicolumn{9}{c}{sum: 31}\\
\begin{tabular}{ccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
fact(a1)&fact(b1)&fact(c1)&fact(d1)&fact(e1)&fact(f1)&fact(g1)&fact(h1)&fact(i1)
\end{tabular}
\end{spreadtab}

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>24</td>
<td>120</td>
<td>720</td>
<td>5040</td>
<td>40320</td>
</tr>
</tbody>
</table>

\footnote{Many others should be written soon and be available in future versions of the package.}
4.1.3 The sumprod macro

The function `sumprod` multiplies the corresponding elements of 2 or more rectangular ranges and then adds these products.

It should be used like this: `sumprod(<range 1>; <range 2>; ...; <range n>).` All the ranges must have the same dimensions.

In this simple example, the average age of a group of children aged from 10 to 15 years old is calculated:

\[
\begin{spreadtab}{{tabular}{r*6c}}
\hline
\text{Ages} & 10 & 11 & 12 & 13 & 14 & 15 \\
\text{Number} & 5 & 8 & 20 & 55 & 9 & 3 \\
\hline
\text{Average} & \frac{\text{sumprod}(b1:g1; b2:g2)}{\text{sum}(b2:g2)} \\
\end{spreadtab}
\]

If any cell in the ranges is empty, pure text or merged with `\multicolumn`, its numeric field is replaced by 0.

4.1.4 Random numbers

The macro-functions `randint` and `rand` each returns a random number depending on its argument.

**Random numbers with fp calc engine**

With this engine, `randint` and `rand` are macro functions of `spreadtab` that rely on the macro `\FPrandom` of `fp`.

It should be noted that the seed initializing the random sequence depends on the date and the minute at which the compilation is done. Thus, the random sequence of numbers given by this function will change between two compilations made at different times. If random numbers need to be repeatable, the private macro `\ST@seed` must be cancelled and a seed should be chosen for `fp`. Here is the code to do this:

\[
\begin{spreadtab}{{tabular}{|l|cccc|}}
\hline
\text{numbers in } [0;1] & \text{rand()} & \text{rand()} & \text{rand()} & \text{rand()} \\
\hline
\text{numbers in } [-5;5] & \text{randint}(-5,5) & \text{randint}(-5,5) & \text{randint}(-5,5) & \text{randint}(-5,5) \\
\text{numbers in } [0;20] & \text{randint}(20) & \text{randint}(20) & \text{randint}(20) & \text{randint}(20) \\
\hline
\end{spreadtab}
\]

The macro-function `randint([<number1>, <number2>])` returns a random integer depending on its arguments: `<number1>` is an optional integer with default value 0. The random integer returned is in the interval `[<number1>; <number2>]`.

The macro-function `rand()` returns a random number between 0 and 1.
Random numbers with \texttt{xfp} engine. With this engine, \texttt{randind} and \texttt{rand} are functions recognized natively by \texttt{xfp}. A warning will be issued if Xe\LaTeX{} is used in an older version that does not implement pseudo-random number generation primitives; in this case, using the functions \texttt{randind} and \texttt{rand} will not be enabled and will throw compilation errors.

As specified in the documentation of \texttt{interface3}, the function \texttt{randint(<a>,<b>)} produces a random integer between the two arguments, but if \texttt{<a>} is omitted, \texttt{randint(<b>)} produces an integer between 1 and \texttt{<b>} which is different from the \texttt{randint} macro function of \texttt{spreadtab} used with \texttt{fp} which produces an integer between 0 and \texttt{<b>}.

\begin{spreadtab}{|l|cccc|}
\hline
\text{Numbers in [0;1]} & \texttt{rand()} & \texttt{rand()} & \texttt{rand()} & \texttt{rand()} \\
\hline
\text{Numbers in [-5;5]} & \texttt{randint(-5,5)} & \texttt{randint(-5,5)} & \texttt{randint(-5,5)} & \texttt{randint(-5,5)} \\
\hline
\text{Numbers in [0;20]} & \texttt{randint(20)} & \texttt{randint(20)} & \texttt{randint(20)} & \texttt{randint(20)} \\
\hline
\end{spreadtab}

\begin{align*}
nombres dans [0;1] & \begin{array}{c}0.111028 \ 0.43194 \ 0.730495 \ 0.818504\end{array} \\
nombres dans [-5;5] & \begin{array}{c}-4 \ -4 \ -3 \ -3\end{array} \\
nombres dans [1;20] & \begin{array}{c}9 \ 13 \ 5 \ 19\end{array}
\end{align*}

4.1.5 \hspace{1em} GCD and LCM

The macro functions \texttt{gcd} and \texttt{lcm} compute the Greatest Common Divisor and the Least Common Multiple of the list of numbers in their argument:

\begin{align*}
gcd(number1,number2,...,numberN) \\
lcm(number1,number2,...,numberN)
\end{align*}

Example:

\begin{spreadtab}{|r|r|r||c|c|}
\hline
\text{Numbers} & \text{GCD} & \text{LCM} \\
\hline
24 & 18 & 12 & 6 & 72 \\
15 & 10 & 25 & 5 & 150 \\
16 & 12 & 15 & 1 & 240 \\
\hline
\end{spreadtab}

4.1.6 \hspace{1em} Scientific notation

The macro function \texttt{scitodec} converts a number written in scientific notation into a decimal number, understandable by \texttt{fp} to achieve its calculations. The syntax is \texttt{scitodec(<text>),} where \texttt{<text>} is:

\begin{itemize}
\item a sequence of characters with the syntax \texttt{<mantissa>EE<exponent>} where \texttt{<mantissa>} is a decimal number and the \texttt{<exponent>} is an integer. The \texttt{EE} may be written uppercase or lowercase.

\texttt{<mantissa>EE<exponent>} means the number \texttt{<mantissa>} $\times 10^{\texttt{<exponent>}}$

\item a reference to the \texttt{textual} field of a cell containing \texttt{<mantissa>EE<exponent>}
\end{itemize}
Example:

\begin{spreadtab}{{tabular}{|r|r|}}
\hline
Scientific notations & Decimal notations \\
\hline
4EE2 & \STcopy{v}{scitodec([-1,0])} \\
-3.1EE-3 & \\\n15ee5 & \\\n-0.025ee7 & \\\n92.125EE0 & \\\n83.1575EE-4 & \\\n\hline
\end{spreadtab}

The engine xfp natively understands numbers written in scientific notation in the form \(<a>e<b>\) but this syntax is impossible to use with spreadtab because the number \(4e3\), which is 4000, would be understood by spreadtab as 4 followed by the content of the cell e3.

4.1.7 Identity

The simplest macro function is \(\text{id(<number>)}\). It returns the number in its argument. Mathematically, it is not very useful, but with spreadtab, it makes possible the writing of mathematical expressions in arguments of macro functions where they are not otherwise allowed – in the argument of \(\text{sum}\) for example.

In the code below, the \text{id} macro function is used to compute the range of cells to add with \(\text{sum}\). In this example, the numeric field of the cell ‘a2’ contains 8. Therefore, \(\text{sum}([0,-1]:[\text{id}(a2-1),-1])/a2\) is equivalent to \(\text{sum}([0,-1]:[7,-1])/8\):

\begin{spreadtab}{{tabular}{r*{10}c}}
\hline
Integers & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
Mean of the :={8} first integers & \text{sum}([0,-1]:[\text{id}(a2-1),-1])/a2 \\
\hline
\end{spreadtab}

Mean of the 8 first integers 4.5

4.2 Tests

Since fp and its macro \FPeval do not accept test in its argument, three macro-functions provide tests (they are not interesting with xfp and its terenary operator ?:, see below):

\begin{verbatim}
ifeq(number1,number2,number3,number4)
ifgt(number1,number2,number3,number4)
iflt(number1,number2,number3,number4)
\end{verbatim}

number1 and number2 are compared:

- for ifeq, is number1 = number2?
- for ifgt, is number1 > number2?
• for \texttt{iflt}, is \texttt{number1} < \texttt{number2}?

If the test is positive, \texttt{number3} is returned, otherwise it is \texttt{number4}.

Here are some values of the function \( f(x) = \begin{cases} 10 & \text{if } x < 1 \\ 0 & \text{if } x = 1 \\ -10 & \text{if } x > 1 \end{cases} \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.5</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>0.5</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.5</td>
<td>-10</td>
</tr>
<tr>
<td>2</td>
<td>-10</td>
</tr>
<tr>
<td>2.5</td>
<td>-10</td>
</tr>
</tbody>
</table>

The engine \texttt{xfp} and its ternary operator \texttt{<a>?<b>:<c>} makes tests easier in the evaluated expressions: if the test \texttt{<a>} is true, \texttt{<b>} is retained otherwise, it is \texttt{<c>}. Thus, the nested tests above become:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.5</td>
<td>([-1,0]&lt;1 ? 10 : [-1,0]=1 ? 0 : -10)</td>
</tr>
<tr>
<td>0</td>
<td>([-1,0]&lt;1 ? 10 : [-1,0]=1 ? 0 : -10)</td>
</tr>
<tr>
<td>0.5</td>
<td>([-1,0]&lt;1 ? 10 : [-1,0]=1 ? 0 : -10)</td>
</tr>
<tr>
<td>1</td>
<td>([-1,0]&lt;1 ? 10 : [-1,0]=1 ? 0 : -10)</td>
</tr>
<tr>
<td>1.5</td>
<td>([-1,0]&lt;1 ? 10 : [-1,0]=1 ? 0 : -10)</td>
</tr>
<tr>
<td>2</td>
<td>([-1,0]&lt;1 ? 10 : [-1,0]=1 ? 0 : -10)</td>
</tr>
<tr>
<td>2.5</td>
<td>([-1,0]&lt;1 ? 10 : [-1,0]=1 ? 0 : -10)</td>
</tr>
</tbody>
</table>

### 4.3 Macro-functions manipulating dates

#### 4.3.1 Date to number with \texttt{engshortdatetonum}

The macro \texttt{engshortdatetonum} converts a short date like 1789/7/14 to an integer which is the number of days passed since the 1st March of the year 0\textsuperscript{11}. It is important to note that this macro-function requires a \textit{textual}

\textsuperscript{11}This year 0 does not exist but this should not be a problem with recent dates.
argument and not a number or the result of a mathematical calculation. Therefore, if the argument of this macro-function refers to a cell, that cell must be a text cell, i.e. a cell containing ‘@’ or ‘:=().

In the example below, the first two lines show how to refer to a text cell. The third line displays the date 0 on the left, and more interesting on the right, it shows how to calculate the number corresponding to the current date with the use of the \TeX counters \texttt{\textbackslash year}, \texttt{\textbackslash month} and \texttt{\textbackslash day} which contain the numbers of the current year, month and day.

\begin{spreadtab}{\{tabular\}{cc}}
\hline
@1789/7/14 & engshortdatetonum(a1) \\
2001/1/1 :=() & engshortdatetonum(a2) \\
\hline
engshortdatetonum(0/3/1) & engshortdatetonum(\texttt{\number\year/\number\month/\number\day})
\end{spreadtab}

Another macro-function provides the same feature but with a long date like ‘December 25, 1789’ or the string contained in \texttt{\today}:

\begin{spreadtab}{\{tabular\}{cc}}
\begin{tabular}{cc}
732295 & 737422 \\
\end{tabular}
\begin{tabular}{cc}
July 1, 1970 & 719649 \\
\end{tabular}
\end{spreadtab}

4.3.2 From a number to a date

Several macro-functions translate a number into a date. All these macro-functions have in common that their result is \textit{text}. Therefore, the cells containing such results \textit{become cells containing text} and if the cell is composed of two fields, the numeric field becomes empty and ‘:=({<formula>})’ is replaced by its result in the text field.

These macro-functions are:

- \texttt{numtoengshortdate} translate a number into a short date like ‘1789/7/14’;
- \texttt{numtoenglongdate} translate a number into a long date like ‘July 14, 1789’;
- \texttt{numtoengmonth} given a number representing a date, it finds the name of the month;
- \texttt{numtoengday} same as above but it finds the name of the day.

Here is an example in which we consider 1000 days before and 1000 days after 2009/6/1. For each of these 2 dates, we calculate the short date, long date, month and day of the week.

\begin{spreadtab}{\{tabular\}{cc}}
\hline
\begin{tabular}{|c|c|}
1000 & numtoengshortdate(engshortdatetonum(a1)+[-1,0]) \\
1000 & numtoenglongdate(engshortdatetonum(a1)+[-1,0]) \\
1000 & numtoengmonth(engshortdatetonum(a1)+[-1,0]) \\
1000 & numtoengday(engshortdatetonum(a1)+[-1,0]) \\
\hline
-1000 & numtoengshortdate(engshortdatetonum(a1)+[-1,0]) \\
-1000 & numtoenglongdate(engshortdatetonum(a1)+[-1,0]) \\
-1000 & numtoengmonth(engshortdatetonum(a1)+[-1,0]) \\
-1000 & numtoengday(engshortdatetonum(a1)+[-1,0]) \\
\end{tabular}
\hline
\end{spreadtab}
4.4 Coordinate macro functions

Rather than referring to a cell by its coordinates which are difficult to remember and change if you insert a row or column, it is sometimes more convenient to give a name to a cell and refer to it later by name.

The macro function ‘\tag{<name>}’ gives a name to the cell in which it is located. This is not really a macro function like the others, since it returns nothing when put in a formula and disappears without causing any effect on the mathematical result. We can write ‘\tag{<nom>}’ anywhere in the numeric field of a cell. The <name> can be any string of alphanumeric characters, but it is not advisable to put a letter and a number that could be understood as a reference to a cell, and would therefore be modified at a copy operation with \STcopy. This macro function has an additional action, it saves via a \def the numeric value of the cell in which it is located in order to be able to use later \tag{<name>} outside the table via the purely expandable command \STtag{<name>}.

Later in the table, instead of writing the coordinates of the cell, we can write ‘\cell(<name>)’, which is a macro function that returns the coordinates of the cell named <name>. For example, if ‘\tag{<name>}’ is written in the cell ‘B3’ and in a further cell, we write ‘\cell(<name>)’, this macro function returns B3.

Here is an example where we add cells and the name ‘foo’ is given to the first number and the name ‘bar’ to the last. One can see that \tag{foo} is between ‘1’ and ‘9’ but at the end, since this macro function disappears, the numeric field of the cell will be ‘19’:

\begin{spreadtab}{{tabular}{r@{}r}}
  \hline
  2 & 15\tag{foo} \\
  3 & + 37 \\
  4 & + 13 \\
  5 & + 48 \\
  6 & + 1\tag{bar}9 \\
  \hline
\end{spreadtab}

\begin{spreadtab}{{tabular}{r@{}r}}
  \hline
  7 & sum(\cell(foo):\cell(bar))\tag{baz} \\
  8 & \hline
\end{spreadtab}

\begin{spreadtab}{{tabular}{c}}
  9 \\
  10 \\
  11 \text{foo=15, bar=19, baz=132}
\end{spreadtab}

To transmit values between tables computed by \spreadtab, it is possible to tag the cell in the first array using the macro function \tag{<name>}, then in the 2nd array, to call up the previously tagged value with \value{<name>}.

\begin{spreadtab}{{tabular}{cc}}
  \hline
  2 & 100 & \text{a1+1\tag{abcd}} \\
  3 & \end{spreadtab}
\end{spreadtab}

\begin{spreadtab}{{tabular}{c}}
  \hline
  4 & \value{abcd} \\
  5 & \end{spreadtab}
\end{spreadtab}

When the environment \spreadtab is nested in another environment, the assignments made by the macro function \tag are local in this environment and could not be accessed outside of this environment via \STtag. In the example below, the table made with \spreadtab is in a center environment and we must use \STmakegtag{<name>} to make global the previous saving of the numerical value contained in the cell marked by \tag{<name>}:

\begin{spreadtab}{{tabular}{c}}
  \hline
  1 \begin{spreadtab}{{tabular}{c}}
    \hline
    2 & 100 \\
    3 & 101 \\
  \end{spreadtab}
\end{spreadtab}

\begin{spreadtab}{{tabular}{c}}
  \hline
  4 \begin{spreadtab}{{tabular}{c}}
    \hline
    5 & \value{abcd} \\
  \end{spreadtab}
\end{spreadtab}

\begin{spreadtab}{{tabular}{c}}
  \hline
  6 \end{spreadtab}
\end{spreadtab}

\begin{spreadtab}{{tabular}{c}}
  \hline
  7 \end{spreadtab}
\end{spreadtab}

---

\textsuperscript{12} As requested on tex.stackexchange
The argument of \STmakegtag can be made of several names, separated with commas. Although at first sight less useful, spreadtab also provides the macro functions 'row(<name>)' and 'col(<name>)' that return the number of the row or column of the cell tag(<name>). Here is an example of how to calculate the average of a number of values; the first and last values are tagged 'first' and 'last' and therefore, the number of values is row(last)-row(first)+1:

\begin{spreadtab}{{tabular}\[b\]{r}}
\begin{tabular}{cccc}
\hline
7 & 9 & 15 & 6 \\
6 & 20 & 13 & 11 \\
\hline
\end{tabular}
\end{spreadtab}

average = \begin{spreadtab}{{tabular}\[b\]{r}}
\begin{tabular}{cccc}
\hline
\STtag(first) & 9 & 15 & 6 & 20 & 13 & 11 \\
\hline
\STtag(last) & row(last)-row(first)+1
\end{tabular}
\end{spreadtab}

5 Particular care

5.1 Defining new commands with \hline

It may be useful to define a new command to produce, for example, a double horizontal line:

\newcommand\dline{\hline\hline}

and then try to use it in a table as in this simple example that computes the Fibonacci sequence in the second line:

\begin{spreadtab}{{tabular}\[c\]{*7c}}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\dline
1 & 1 & a2+b2 & b2+c2 & c2+d2 & d2+e2 & e2+f2
\end{spreadtab}

In the log file, you can read that \FPeval fails and complains:

! Improper alphabetic constant.

The reason is simple, \dline in line 4 is not recognized by spreadtab as a horizontal rule and therefore, it is placed in the cell in the next line. For spreadtab, the cell b1 contains:

\dline 1

Since there is no \ or :={(<formula>)}, spreadtab considers that the whole cell is a numeric field and \FPeval tries valiantly to calculate this content and obviously fails.

To compile without error, the cell a2 must contain a numeric field marker:
5.2 The use of \texttt{\textbackslash multicolumn} and \texttt{\textbackslash SThidecol}

Firstly, in normal use, joint use of \texttt{\textbackslash multicolumn} and \texttt{\textbackslash SThiderow} should not happen, and most users should not encounter this situation and should not read this section.

For the brave here is the problem: first, a hidden column must not contain a cell with the command \texttt{\textbackslash multicolumn}! But what happens if a hidden column hides cells merged with \texttt{\textbackslash multicolumn}?

In general, there is no compilation error or error messages, but there are some subtleties about the references that are a bit turned upside down in the line after the \texttt{\textbackslash multicolumn} command...

Let’s take an example, and let’s say that, in the following table, we want to merge the cell b2 to h2 and we also want to hide the columns c, d and f, here in gray:

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
a1 & b1 & c1 & d1 & e1 & f1 & g1 & h1 & i1 & j1 \\
\hline
\end{array}
\]

There are 4 visible merged cells, so we write \texttt{\textbackslash multicolumn(4)} because hidden columns are never taken into account when counting the number of \texttt{\textbackslash multicolumn}.

Then we count 4 letters from b (this letter included): we obtain the letter e. In the range b–e, let’s count: 2 gray hidden columns are included (c and d) and 1 hidden column is not included (f). These numbers are important to understand the following, also let’s call them x and y in the general case.

The rule is:

\begin{itemize}
  \item it is necessary to add y signs \& after \texttt{\textbackslash multicolumn} (in the example above, it would be 1);
  \item references to columns of cells after the \texttt{\textbackslash multicolumn} will be shifted x to the beginning of the alphabet.
\end{itemize}

For the example given, if we want to refer to the cell i2, we should write g2 instead of i2.

Here is an example with a similar structure to the previous (x = 2 and y = 1) with simple formulas: add 1 to the number above.

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
a2 & b2 & \SThiderow3 & \SThiderow4 & \SThiderow5 & \SThiderow6 & \SThiderow7 & \SThiderow8 & \SThiderow9 & \SThiderow10 \\
\hline
a1 & \texttt{\textbackslash multicolumn(1)} & \& & & & & & & & \\
\hline
\end{array}
\]

And another example with x = 1 and y = 0 where a single column is hidden d:
5.3 Messages delivered by \texttt{spreadtab}

The package delivers error messages and aborts compilation in these cases:

- a circular reference is found in a cell. In this case, the dependent cells are displayed;
- a cell refers to an empty cell or a text cell when a non-empty numeric field is expected;
- a cell refers to an undefined cell (outside the table);
- a cell refers to a cell merged by a \texttt{\multicolumn} command;
- a relative reference has bad syntax.

The package can deliver informative messages (in the log file), which it does by default. If the user wants or not the delivery of informative messages, the syntax is \texttt{\STmessage{true}} or \texttt{\STmessage{false}}.

To understand the meaning of these messages, let’s take a simple table:

\begin{spreadtab}{{tabular}{|cccc|}}
\hline
1 & 2 & 3 & \SThidecol4 \\
\hline
a1+1 & b1+1 & i1+1 & j1+1 \\
\hline
a2+1 & b2+1 & i2+1 & j2+1 \\
\hline
\end{spreadtab}

\begin{tabular}{|cccc|}
\hline
1 & 2 & 3 & 5 \\
9 & 10 & 11 & 12 \\
\hline
\end{tabular}

Here are the messages delivered by \texttt{spreadtab}:

\begin{spreadtab}{{tabular}{|cccc|}}
\hline
b1+1 & c1+1 & d1+1 & 10 \\
\hline
\end{spreadtab}

\begin{itemize}
\item \texttt{\STmessage{true}} already set by default
\item \texttt{\STmessage{true}} New \texttt{spreadtab \{tabular\}}{\{cccc\}}
\item * reading tab: ok
\item * computing formulas:
\item \texttt{cell A1-B1-C1}
\item \texttt{cell B1}
\item \texttt{cell C1}
\item \texttt{cell D1}
\item \texttt{cell E1}
\item * building tab: ok
\item \texttt{\STmessage{true}} End of \texttt{spreadtab}
\end{itemize}

Preceded by a star, we recognize the 3 steps necessary for \texttt{spreadtab} to complete its task: reading the table, calculation of the formulas and building the final table.

For the second step, cells are evaluated from top to bottom, left to right: at line 4 in the code above, \texttt{spreadtab} says that it begins by trying to calculate the first cell \texttt{A1}. After a dash, we see that for this, it must first compute the cell \texttt{B1}, which itself requires that the cell \texttt{C1} is calculated: the latter can be calculated since it depends only on \texttt{D1} which is a cell containing the number 10.

In the following (lines 5 to 8), there is only one cell per line which means that when \texttt{spreadtab} tries to evaluate the cell, either it contains a number or dependent cells are already calculated.
5.4 Debug mode

To ease the use of spreadtab, a debug mode is available. It is activated when the command \STdebug is written in the optional argument of the spreadtab environment. This command changes the behavior of spreadtab which, instead of displaying the final table, displays one (or more) table(s) containing debugging information. This display is done just after spreadtab has read all the cells, and no calculating of formulas has yet taken place. There are as many tables as commands \STdebug, provided that their argument is different. Only 3 arguments are possible:

- \STdebug{formula}: displays all the numeric fields and the ends of lines;
- \STdebug{text}: displays all the textual fields;
- \STdebug{code}: displays the internal code of the cells. Indeed, spreadtab assigns a code to every cell when it reads the table. Here are the possible values of this code:
  - −1 if the cell is merged with \multicolumn;
  - 0 if the cell is a text cell or is empty;
  - 1 if the numeric field of the cell contains a formula which will be computed later;
  - 2 if the numeric field of the cell contains a number.

When the 'debug mode' is activated, the final table is not displayed.

Here is a table which will be used for the next example:

\begin{spreadtab}{\{tabular\}{|r|r|r|}}
\hline
\$x$ & @$y$ & @$x+y$ \\
\hline
22 & 54 & \STcopy{v3}{a2+b2} \\
43 & 65 & \\
49 & 37 & \\
\hline
\end{spreadtab}

\begin{tabular}{|r|r|}
\hline
\multicolumn{2}{|r|}{$Sx=:={a2+a3+a4}$} & @\multicolumn{1c}{} \\
\hline \\
\multicolumn2{|r|}{$Sy-Sx=:={b5-a5}$} & @\\
\end{tabular}

\bigskip

Let’s ask spreadtab to show the 3 possible debugging tables for the table above. To do this, just change line 1 in the code above to:

\begin{spreadtab}\{\STdebug{text}\STdebug{formula}\STdebug{code}\}\{tabular\}{|rr|r|}\
\hline
\multicolumn{2}{|r|}{$Sx=:={a2+a3+a4}$} & @\multicolumn{1c}{} \\
\multicolumn2{|r|}{$Sy-Sx=:={b5-a5}$} & @\\
\end{spreadtab}

\begin{tabular}{|r|r|}
\hline
\multicolumn{2}{|r|}{$Sx=:={a2+a3+a4}$} & @\multicolumn{1c}{} \\
\hline \\
\multicolumn2{|r|}{$Sy-Sx=:={b5-a5}$} & @\\
\end{tabular}

\bigskip

\texttt{\begin{spreadtab}\{\STdebug{text}\STdebug{formula}\STdebug{code}\}\{tabular\}{|rr|r|}\
\hline
\multicolumn{2}{|r|}{$Sx=:={a2+a3+a4}$} & @\multicolumn{1c}{} \\
\end{spreadtab}}
These 3 debugging tables may help to better understand what happens behind the scene when spreadtab works. We can observe that all cells with a numeric field (see table 2) have an internal code of 1 or 2 (see table 3) and an associated numeric field marker ‘:=’ in table 1. This marker represents the location where (by substitution) the result of the numeric field will be inserted. So from the contents of text fields in table 1 and, once calculated, the numeric fields, by a simple substitution, the cells are reconstituted to give those of the final table.

In the debugging tables, cells containing the coordinates are grayed if the package colorbl has been loaded; they are left white otherwise.

6 Examples

In the examples of this section, the numbers entered by the user are in red and the calculated numbers are in black

In these tables, lots of tricks (struts, \multicolumn commands) and packages (including the numprint package and its columns ‘N’, which aligns the decimal points) were used to obtain a satisfactory result. The code is sometimes cumbersome and difficult to read, but these tables are not basic but well-groomed examples!

6.1 Pascal’s triangle again!

\begin{spreadtab}{{tabular}{*7r}}
\end{spreadtab}

\begin{spreadtab}{{tabular}{cc}}
\end{spreadtab}

6.2 The convergence of a series

For people familiar with maths, here is the series of the exponential. Indeed,

$$\forall x \in \mathbb{R}, \quad e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

and this table shows the speed of convergence for $x = 0.5$

\begin{spreadtab}{{tabular}{cc}}
\end{spreadtab}
6.3 Convergence to the golden ratio

In mathematical terms, the sequence \( F_n \) of Fibonacci numbers is defined by the recurrence relation:

\[
F_0 = 1 \quad F_1 = 1 \quad F_{n+2} = F_{n+1} + F_n
\]

The golden ratio is the limit of the ratios of successive terms of the Fibonacci sequence. We show here that the quotients \( F_{n+1}/F_n \) approximate the golden ratio \( \varphi = \frac{1 + \sqrt{5}}{2} \) alternately lower and higher than \( \varphi \).
6.4 A billing table

Here is a billing table where the decimal points are aligned in columns with the column specifier 'N' of the package numprint.

This table is generated by the environment \texttt{tabularx} stretched to fit 80\% of the width of the line. The command \texttt{\multicolumn} has been widely used for formatting:

\begin{spreadtab}{{\tabularx}{0.8\linewidth}{|\textbf{X}|\textbf{c N42}|\textbf{c N42}|\textbf{c <\PC|}}\hline
\item \textbf{Item} & \textbf{Price/U} & \textbf{Qty} & \textbf{Price} & \textbf{Reduction} & \textbf{Net} \\
\hline
\item Item 1 & 5.99 & 20 & 119.80 & –20\% & 95.84 \\
\item Item 2 & 12.00 & 7 & 84.00 & –10\% & 75.60 \\
\item Item 3 & 4.50 & 40 & 180.00 & –35\% & 117.00 \\
\item Item 4 & 650.00 & 2 & 1 300.00 & –15\% & 1 105.00 \\
\hline
\item \multicolumn{6}{c}{\textbf{Total}} & 1 683.80 & –17\% & 1 393.44 \\
\end{spreadtab}
6 EXAMPLES

\begin{spreadtab}{|*3{>{\hfill\rule[-0.4cm]{0pt}{1cm}$}m{0.7cm}<{$\hfill\null}|}}}
\hline
\color{red}:=2 & 5*b2-4*a1 & 3*a1-2*b2 \\
\hline
2*a1-b2 & \color{red}:={-1} & 3*b2-2*a1 \\
\hline
4*b2-3*a1 & 4*a1-3*b2 & 2*b2-a1 \\
\hline
\end{spreadtab}

\begin{tabular}{|c|c|c|}
\hline
2 & -13 & 8 \\
\hline
5 & -1 & -7 \\
\hline
-10 & 11 & -4 \\
\hline
\end{tabular}

2 –13 8
5 –1 –7
–10 11 –4

6.6 A pyramid of additions

Each number is the sum of two numbers located below it.

\begin{spreadtab}{*{8}{@{}p{.5\cellsize}@{}}}
\cline{4-5}
\multicolumn{2}{|K|}{:={\[-1,1\]+[1,1]}}&&&\
\cline{3-6}
\multicolumn{2}{|K}{:={\[-1,1\]+[1,1]}}&\multicolumn{2}{|K|}{:={\[-1,1\]+[1,1]}}&&&\
\cline{2-7}
\multicolumn{2}{|K}{:={\[-1,1\]+[1,1]}}&\multicolumn{2}{|K}{:={\[-1,1\]+[1,1]}}&\multicolumn{2}{|K}{:={\[-1,1\]+[1,1]}}&\
\cline{3-7}
\multicolumn{2}{|K}{\color{red}:={-5}}&\multicolumn{2}{|K}{\color{red}:={3}}&\multicolumn{2}{|K}{\color{red}:={-2}}&\multicolumn{2}{|K|}{\color{red}:={-3}}\
\hline
\end{spreadtab}

\begin{tabular}{|c|c|c|}
\hline
-5 & 1 & -5 \\
\hline
-2 & 1 & -5 \\
\hline
-5 & 3 & -2 & -3 \\
\hline
\end{tabular}

\*
\*
\*

That’s all. I hope you will find this package useful!

I thank you in advance for sending by email any bug you find, any macro-function or improvement you would like to be implemented, assuming that it must be realistic. This package has to be modest and spreadtab is not excel or calc: it is impossible to implement some advanced features of these spreadsheets.

Christian Tellechea