

spreadtab

v0.1

User's manual

Christian TELLECHEA
unbonpetit@gmail.com

November 3rd 2009

Abstract

This package provides spreadsheet features for L^AT_EX 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	2
2	Basic features	3
2.1	Absolute references	3
2.2	Relative references	3
2.3	Text cells	4
2.4	Mixed cells	4
3	Features for formatting the table	5
3.1	End of lines and horizontal rules	5
3.2	Hide a row or column	6
3.3	Save the result of a cell	6
3.4	The use of <code>\multicolumn</code>	7
3.5	Number formatting and the <code>fp</code> package	8
4	Macro-functions	8
4.1	Mathematical macro-functions	8
4.1.1	Sum cells	8
4.1.2	The <code>fact</code> macro	9
4.1.3	The <code>sumprod</code> macro	9
4.1.4	Random numbers	9
4.2	Tests	10
4.3	Macro-functions manipulating dates	11
4.3.1	Date to number with <code>engshortdatetotnum</code>	11
4.3.2	From a number to a date	11
5	Particular care	12
5.1	Defining new commands with <code>\hline</code>	12
5.2	The use of <code>\multicolumn</code> and <code>\SThidecol</code>	13
5.3	Messages delivered by <code>spreadtab</code>	14
6	Examples	14
6.1	The Pascal's triangle again!	15
6.2	The convergence of a series	15
6.3	Convergence on the golden ration	16
6.4	A billing table	17
6.5	A magic square	17
6.6	A pyramid of additions	18

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 requires ε -TeX, L^AT_EX 2_ε and the **fp** package, which performs arithmetic on cell values. Also, the **xstring** package is needed in its v1.5c [2009/06/05] version or later.

The package is compatible with *all* tabular environments, and assumes that ‘&’ is used to delimit columns and ‘\’ to end the lines. 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 table ‘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 done by the **fp** package;
- 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.

Here is the syntax:

```
1 \begin{spreadtab}{<name of table environment><parameters of environment>}
2   table with formulas and numbers
3 \end{spreadtab}
```

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

```
1 \begin{<name of table environment><parameters of environment>
2   table with numbers
3 \end{<name of table environment>}
```

Although having features resembling those of a spreadsheet with L^AT_EX is appreciable, the 3 stages described above take time, and above all, **fp** is slow in its calculations. 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 difficulty¹ for big or complex tables. The syntax of **spreadtab** is also another difficulty. In fact, the advantage of this package is that it makes possible to write *in the L^AT_EX code* tables involving calculation when these tables are usually exported² from a spreadsheet program to L^AT_EX 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 formulas 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 L^AT_EX packages to imitate – even a little – the behaviour of spreadsheet programs. I found the challenge interesting and I started writing in this package which is only a good programming exercise.

¹I certify that, with the use, this discomfort tends to disappear (if you do not work with huge tables, of course).

²I mention the main two exportation programs: **cacl2latex** for ‘calc’ (Open Office), and **excel2latex** for ‘excel’ (Microsoft Office).

Indeed, the main algorithm contained in `spreadtab` that determines the order in which the cells must be calculated is quite general and not specific to \LaTeX . Other \LaTeX oriented algorithms analyze cells and formulas to find references to other cells. The exercise is interesting because for programming, \TeX does not provide anything resembling a pointer, useful to build linked lists of cells to calculate. Nothing is available either to attach to a ‘variable’ ‘chils fields’ which you can choose the type.

The road was long before reaching this first 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 Huu Dien Khue LE for the Vietnamese translation of this manual.

2 Basic features

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

2.1 Absolute references

A table cell is identified by the pair `<colref><rowref>`³, where:

- `<colref>` is a letter from `a` to `z`, and `a` 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;
- `<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`.

This example calculates the sum each row and each column and then calculates the grand total:

<code>1 \begin{spreadtab}{\{tabular\}{rr r}}</code>			
<code>2 22 & 54 & a1+b1 \\\</code>	22	54	76
<code>3 43 & 65 & a2+b2 \\\</code>	43	65	108
<code>4 49 & 37 & a3+b3 \\\</code>	49	37	86
<code>5 \hline</code>			
<code>6 a1+a2+a3 & b1+b2+b3 & a4+b4</code>	114	156	270
<code>7 \end{spreadtab}</code>			

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

<code>1 \begin{spreadtab}{\{tabular\}{ccccc}}</code>					
<code>2 1 & & & & \\\</code>	1				
<code>3 a1 & a1 & & & \\\</code>	1	1			
<code>4 a2 & a2+b2 & b2 & & \\\</code>	1	2	1		
<code>5 a3 & a3+b3 & b3+c3 & c3 & \\\</code>	1	3	3	1	
<code>6 a2 & a4+b4 & b4+c4 & c4+d4 & d4</code>	1	4	6	4	1
<code>7 \end{spreadtab}</code>					

2.2 Relative references

To refer to a cell, it may be convenient to specify its position relatively 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.

³Note: this is the opposite to the standard matrix convention

Here is the same table as above but the references are relative and the `matrix` environment of the `amsmath` is used:

<pre> 1 \$ 2 \begin{spreadtab}{{matrix}}{} 3 1\\ 4 [0,-1] & [-1,-1]\\ 5 [0,-1] & [-1,-1]+[0,-1] & [-1,-1]\\ 6 [0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]\\ 7 [0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1]+[0,-1] & [-1,-1] 8 \end{spreadtab} 9 \$ </pre>	<pre> 1 1 1 1 2 1 1 3 3 1 1 4 6 4 1 </pre>
--	--

We note that relative references are more appropriate here, since only 2 different references are used: `[0,1]` which refers to the cell below and `[-1,1]` which refers to the cell located at the SW 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 character ‘@’ with its usual catcode 12. The cell will be ignored by `spreadtab` which will consider it as an inert cell impossible to refer⁴ elsewhere in the table.

Example:

<pre> 1 \begin{spreadtab}{{tabular}{ r ccc }} 2 \hline 3 @ values of de \$x\$ & -5 & & -1 & & 4 \\ 4 @ \$f(x)=2x\$ & & 2*[0,-1] & & 2*[0,-1] & & 2*[0,-1] \\ 5 \end{spreadtab} </pre>	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px 10px;">values of x</td> <td style="padding: 2px 10px;">-5</td> <td style="padding: 2px 10px;">-1</td> <td style="padding: 2px 10px;">4</td> </tr> <tr> <td style="padding: 2px 10px;">$f(x) = 2x$</td> <td style="padding: 2px 10px;">-10</td> <td style="padding: 2px 10px;">-2</td> <td style="padding: 2px 10px;">8</td> </tr> </table>	values of x	-5	-1	4	$f(x) = 2x$	-10	-2	8
values of x	-5	-1	4						
$f(x) = 2x$	-10	-2	8						

The control sequence `\STtextcell` expands to the character ‘@’. It is possible to redefine it, for example, after `\renewcommand\STtextcell{\celltext}`, a cell containing `\celltext` 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 `fp` 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 field>:={numeric field}<end of text field>`

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

⁴There is an exception, see page 11.

```
\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 `\textbf{:= a_1+1 }` and if the numeric value of the cell `a1` is 5, then finally, the cell will contain `\textbf{6}`

Here is a simple example:

```
1 \begin{spreadtab}{{\tabular}{|c|c||c|}}\hline
2 value 1 : := ${50}$  & value 2 : := ${29}$  & average : \textbf{:= ${(a_1+b_1)/2}$ }}\hline
3 \end{spreadtab}
```

value 1 : 50	value 2 : 29	average : 39.5
--------------	--------------	-----------------------

If ‘:=’ has an empty argument like this ‘:=`{}`’, then the cell is understood as a text cell. It is a synonym of ‘@’.

3 Features for formatting the table

3.1 End of lines and horizontal rules

`spreadtab` recognizes the usual line breaks and horizontal rules `\` and `\hline`. Also, it may use the optional argument in line break: `\`[<dimension>].

For horizontal rule, it is possible to use:

- `\hline`;
- `\cline{x-y}` where `x` and `y` define the start and the end of the rule;
- `\hhline{<type>}` where `<type>` is the type of rule (read the manual of the `hhline` package).

Example:

```
1 \begin{spreadtab}{{\tabular}{*5c}}
2 [0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]+[0,1] & [-1,1]\
```

1	4	6	4	1
1	3	3	1	
1	2	1		
<hr/>				
1	1			
<hr/>				
1				

3.2 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, `spreadtab` provides two control sequences `\SThiderow` and `\SThidecol` which, when placed in a cell, hide the row or column that contains the cell.

An example:

```

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

```

values of x	-1	2	3
$f(x) = 2x - 1$	-3	3	5
$h(x) = 1 - x$	2	-1	-2

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

You must be remember that the hidden rows and columns are *invisible* to the tabular environment chosen by the user. Thus, only 4 columns have been defined (`|r|ccc|`) and not 5 as seen by `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:

values of x	-1	0	2	3
$f(x) = 2x - 1$	-3	-1	3	5
$g(x) = x - 10$	-11	-10	-8	-7
$h(x) = 1 - x$	2	1	-1	-2

3.3 Save the result of a cell

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

`\STsavecell{<control sequence>}{<absolute reference>}`

With a `\global\def`⁵, this command globally saves in `<control sequence>` the result of the formula contained in the cell `<absolute reference>`.

Only absolute references can be used since this command must be placed in the optional argument of the `spreadtab` environment.

Example:

```

1 \begin{spreadtab}[\STsavecell\result{c1}]{tabular}{|c|c|c|c|c|}
2 \hline
3 10 & a1+10 & b1+10 & a1+b1+c1 & @cell c1 : \result \\ \hline
4 \end{spreadtab}
5 \par\medskip
6 Here is the cell c1 : \result

```

10	20	30	60	cell c1 : 30
----	----	----	----	--------------

Here is the cell c1 : 30

⁵The `\def` command does not check if the macro it defines already exists.

In order to save several cells, the command `\STsavecell` can be put several times in the optional argument.

Example:

```

1 \begin{spreadtab}[\STsavecell\hhh{b3}\STsavecell\mmm{c3}\STsavecell\sss{d3}]{\tabular
2   }{|rc|}\hline
3 @Speed (km/h) &\SThidecol&\SThidecol&\SThidecol& 35 \\
4 @distance (km) & & & & 180\\\hline
5 @Time (h min s) & trunc(e2/e1,0) & trunc(60*(e2/e1-b3),0) & trunc(3600*(e2/e1-b3)-60*
6   c3,1) &\hhh\ h \mmm\ min \sss\ s\\\hline
7 \end{spreadtab}\par\medskip
8 It lasts more than \hhh\ hours.

```

Speed (km/h)	35
distance (km)	180
Time (h min s)	5 h 8 min 34.2 s

It lasts more than 5 hours.

3.4 The use of `\multicolumn`

`spreadtab` is compatible with the syntax `\multicolumn{<number>}{<type>}{<content>}` which merges `<number>` cells in unique cell whose type and content are specified in the arguments.

Even when using `\multicolumn`, `spreadtab` maintains some consistency in the references. In this table where `\multicolumn` is used, the absolute references are displayed:

a1	b1	c1	d1	e1	f1	g1
a2	b2		d2	e2	f2	g2
a3			d3	e3		g3

Thus, whatever be the number of merged cells, the next cell has a column number that take into account the number of merged cells.

In the last line, cells a3, b3 and c3 are merged and consequently, the cells b3 and c3 *do not exist* for `spreadtab`: it is not possible to refer to b3 or c3 anywhere in the table.

In this example, every number in the top line is the product of the 2 numbers below:

```

1 \newcolumntype{K}[1]{@{}>\centering\arraybackslashp{#1cm}@{}}
2 \begin{spreadtab}{\tabular}{*6{K{0.5}}}}
3 \cline{2-5}
4 &\multicolumn{2}{|K{1}|}{:={a2*c2}} & \multicolumn{2}{|K{1}|}{:={c2*e2}} & \\\hline
5 \multicolumn{2}{|K{1}|}{:=8}&\multicolumn{2}{|K{1}|}{:=7}&\multicolumn{2}{|K
6   {1}|}{:=6}\\\hline
7 \end{spreadtab}

```

	56	42	
8	7	6	

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

3.5 Number formatting and the fp package

All calculations are made by the `\FPeval` macro⁶ of the `fp` package. This package provides all necessary along with various scientific and trigonometric functions. Calculations are made with 18 decimal digits of precision, and `fp` displays *all* the decimals!

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 trunc 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.

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

```

1 \STautoround{6}
2 \begin{spreadtab}{\tabular}{|l|*7>{\centering\arraybackslash}m{1.45cm}|}}
3 \hline
4 @$$x$ & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \\\hline
5 @$x^{-1}$ & 1/b1 & 1/c1 & 1/d1 & 1/e1 & 1/f1 & 1/g1 & 1/h1\\\hline
6 \end{spreadtab}

```

x	1	2	3	4	5	6	7
x^{-1}	1	0.5	0.333333	0.25	0.2	0.166667	0.142857

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 available⁷. 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]`;
- either 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 the ranges of cell, if a cell does have a numeric field (empty cell or text cell or merged cell with `\mutlicolumn`), it is seen as 0 by `sum`.

⁶This macro accepts infix or postfix notation. Consequently, both can be used to write formulas in a cell. For example the infix formula `'a1+b1'` is equivalent to the postfix ones `'a1 b1 add'` or `'a1 b1 +'`.

⁷Many others should be written soon and be available in future versions of the package.

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

1	4	6	4	1
1	3	3	1	
1	2	1		
1	1			
1				

4.1.2 The fact macro

The macro-function `fact(<number>)` computes the factorial of its argument, assumed it is an *integer* between 0 and 18 to avoid overflows⁸. The `<number>` can also be a reference to a cell whose numeric field contains an integer.

Here are the factorial from 0 to 8:

0	1	2	3	4	5	6	7	8
1	1	2	6	24	120	720	5040	40320

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:

Ages	10	11	12	13	14	15
Number	5	8	20	55	9	3
Average	12.64					

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` return a random number depending of its argument.

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 times which minutes differ. If randoms 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:

⁸Indeed, for `fp` the greatest integer is $10^{18} - 1$. The factorial of 19 is greater than it.

```

1 \makeatletter
2 \renewcommand\ST@seed{}% redefines the private macro
3 \makeatletter
4 \FPseed=27% gives a seed (any integer) to fp.

```

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

The macro-function `rand()` returns a random number between 0 and 1.

```

1 \STautoround{6}
2 \begin{spreadtab}{{\tabular}{|l|cccc|}}\hline
3 @numbers in [0;1] & \rand() & \rand() & \rand() & \rand() & \\
4 @numbers in [-5;5] & \randint(-5,5) & \randint(-5,5) & \randint(-5,5) & \randint(-5,5) & \\
5 @numbers in [0;20] & \randint(20) & \randint(20) & \randint(20) & \randint(20) & \\
6 \hline
7 \end{spreadtab}

```

numbers in [0;1]	0.173889	0.549332	0.629985	0.160506
numbers in [-5;5]	1	-2	-1	-5
numbers in [0;20]	5	0	18	9

4.2 Tests

Three macro-functions provide tests:

`ifeq(number1,number2,number3,number4)`

`ifgt(number1,number2,number3,number4)`

`iflt(number1,number2,number3,number4)`

`number1` and `number2` are compared:

- for `ifeq`, is `number1 = number2`?
- for `ifgt`, is `number1 > number2`?
- for `iflt`, is `number1 < number2`?

If the test is positive, `number3` is returned, otherwise it is `number4`.

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

```

1 \begin{spreadtab}{{\tabular}{|*2c|}}\hline
2 @$x$ & @$f(x)$ & \\
3 -0.5 & \iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\ \\
4 [0,-1]+0.5 & \iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\ \\
5 [0,-1]+0.5 & \iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\ \\
6 [0,-1]+0.5 & \iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\ \\
7 [0,-1]+0.5 & \iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\ \\
8 [0,-1]+0.5 & \iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\ \\
9 [0,-1]+0.5 & \iflt([-1,0],1,10,ifeq([-1,0],1,0,-10))\ \\
10 \end{spreadtab}

```

x	$f(x)$
-0.5	10
0	10
0.5	10
1	0
1.5	-10
2	-10
2.5	-10

4.3 Macro-functions manipulating dates

4.3.1 Date to number with engshortdatetotnum

The macro `engshortdatetotnum` 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 0⁹. It is important to note that this macro-function requires a *textual* 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 T_EX counters `\year`, `\month` and `\day` which contain the numbers of the current year, month and day.

```

1 \begin{spreadtab}{{tabular}{cc}}
2 @1789/7/14 & & engshortdatetotnum(a1)\
3 2001/1/1 :={} & & engshortdatetotnum(a2)\
4 engshortdatetotnum(0/3/1) & engshortdatetotnum(\number\year/\number\month/\number\day)
5 \end{spreadtab}

```

1789/7/14	653554
2001/1/1	730791
0	734019

Another macro-function provides the same feature but with a long date like 'December 25, 1789' or the string contained in `\today`:

```

1 \begin{spreadtab}{{tabular}{cc}}
2 englongdatetotnum(February 13, 2005) & englongdatetotnum(\today)\
3 @July 1, 1970 & & englongdatetotnum(a2)
4 \end{spreadtab}

```

732295	734019
July 1, 1970	719649

4.3.2 From a number to a date

Several macro-functions enable to translate a number into a date. All these macro-functions have in common that their result is *text*. Therefore, the cells containing such results *become cells containing text*: the numeric field becomes empty and the text field becomes this result.

This macro-functions are:

- `numtoengshortdate` translate a number into a short date like '1789/7/14';
- `numtoenglongdate` translate a number into a long date like 'July 14, 1789';
- `numtoengmonth` given a number representing a date, it finds the name of the month;
- `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.

⁹This year 0 does not exist but this should not be a problem with recent dates.

```

1 \begin{spreadtab}{{\tabular}{cc}}                                \hline
2 \multicolumn{2}{|c|}{@2009/6/1}                               \\ \hline \hline
3 1000 & \numtoengshortdate(engshortdatetotnum(a1)+[-1,0]) \\
4 1000 & \numtoenglongdate(engshortdatetotnum(a1)+[-1,0]) \\
5 1000 & \numtoengmonth(engshortdatetotnum(a1)+[-1,0]) \\
6 1000 & \numtoengday(engshortdatetotnum(a1)+[-1,0]) \\
7 -1000 & \numtoengshortdate(engshortdatetotnum(a1)+[-1,0]) \\
8 -1000 & \numtoenglongdate(engshortdatetotnum(a1)+[-1,0]) \\
9 -1000 & \numtoengmonth(engshortdatetotnum(a1)+[-1,0]) \\
10 -1000 & \numtoengday(engshortdatetotnum(a1)+[-1,0]) \\
11 \end{spreadtab}

```

2009/6/1	
1000	2012/2/26
1000	February 26, 2012
1000	February
1000	sunday
-1000	2006/9/5
-1000	September 5, 2006
-1000	September
-1000	tuesday

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 Fibonacci sequence in the second line:

0	1	2	3	4	5	6
1	1	2	3	5	8	13

But, if you write the following code, there is a problem when compiling:

```

1 \newcommand\dline{\hline\hline}
2 \begin{spreadtab}{{\tabular}{*7c}}
3 0 & 1 & 2 & & 3 & & 4 & & 5 & & 6 & \\ \dline
4 1 & 1 & a2+b2 & b2+c2 & c2+d2 & d2+e2 & e2+f2
5 \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 an horizontal rule and therefore, *it is placed in the cell 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* contains a numeric field marker:

```

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

```

0	1	2	3	4	5	6
1	1	2	3	5	8	13

5.2 The use of `\multicolumn` and `\SThidecol`

Firstly, in normal use, joint use of `\multicolumn` and `\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 `\multicolumn`! But what happens if a hidden column hides cells merged with `\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 `\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:

a1	b1	c1	d1	e1	f1	g1	h1	i1	j1
a2	b2							i2	j2

There are 4 visible merged cells, so we write `\multicolumn{4}` because hidden columns are never taken into account when counting the number of `\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 a and b in the general case.

The rule is:

- it is necessary to add b signs `&` after `\multicolumn` (in the example above, it would be 1);
- references to columns of cells after the `\multicolumn` will be shifted a to the beginning of the alphabet. 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 than the previous ($a = 2$ and $b = 1$) with simple formulas: add 1 to the number above.

```

1 \begin{spreadtab}{\tabular}{|*{7}{c|}}
2 \hline
3 1 & 2 & & \SThidecol3 & \SThidecol4 & 5& \SThidecol6 & 7& 8& 9 & 10 \\ \hline
4 a1+1& \multicolumn4{1}|{:=b1+1}} & & & & & i1+1 & j1+1 \\ \hline
5 a2+1& b2+1 & & & & & & & g2+1 & h2+1 \\ \hline
6 \end{spreadtab}

```

1	2	5	7	8	9	10
2	3				10	11
3	4				11	12

And another example with $a = 1$ and $b = 0$ where a single column is hidden `d`:

```

1 \begin{spreadtab}{\tabular}{|*{9}{c|}}
2 \hline
3 1 & 2 & & 3 & & \SThidecol4 & 5 & 6 & 7 & 8 & & 9 & & 10 \\ \hline
4 a1+1& \multicolumn6{1}|{:=b1+1}} & & & & & & i1+1 & j1+1 \\ \hline
5 a2+1& b2+1 & & & & & & & & & & h2+1 & i2+1 \\ \hline
6 \end{spreadtab}

```

1	2	3	5	6	7	8	9	10
2	3						10	11
3	4						11	12

5.3 Messages delivered by spreadtab

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

- a circular reference is found in a cell. In this case, the dependant 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 merged cell by a `\multicolumn` command;
- a relative reference has a bad syntax.

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

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

```

1 \STmessage{true}% already set by default
2 \begin{spreadtab}{\tabular}{|cccc|c|}\hline
3 b1+1 & c1+1 & d1+1 & 10 & a1+b1+c1+d1\\\hline
4 \end{spreadtab}

```

13	12	11	10	46
----	----	----	----	----

Here are the messages delivered by spreadtab:

```

1 [spreadtab] New spreadtab {tabular}{|cccc|c|}
2 * reading tab: ok
3 * computing formulas:
4   cell A1-B1-C1
5   cell B1
6   cell C1
7   cell D1
8   cell E1
9 * building tab: ok
10 [spreadtab] End of spreadtab

```

Preceded by a star, we recognize the 3 steps necessary to 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, spreadtab says that it begins by trying to calculate the first cell A1. After a dash, we see that for this, it must first compute the cell B1, which itself requires that the cell C1 is calculated: the latter can be calculated since it depends only on 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 spreadtab tries to evaluate the cell, either it contains a number or dependant cells are already calculated.

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!

Convergence for $x = 0,5$

$$n \quad e^{0,5} - \sum_{k=0}^n \frac{0,5^k}{k!}$$

0	0,648 721 270 700 128
1	0,148 721 270 700 128
2	0,023 721 270 700 128
3	0,002 887 937 366 795
4	0,000 283 770 700 128
5	0,000 023 354 033 461
6	0,000 001 652 644 572
7	0,000 000 102 545 366
8	0,000 000 005 664 166
9	0,000 000 000 281 877

6.3 Convergence on the golden ration

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 ration $\varphi = \frac{1+\sqrt{5}}{2}$ alternately lower and higher than φ .

```

1 \STautoround{9}
2 $
3 \begin{spreadtab}{{matrix}}{
4 @n      & @F_n      & & @\dfrac{F_n}{F_{n-1}} & @\varphi-\dfrac{F_n}{F_{n-1}}
5 -1}}\hline
6 \color{red}:=1 & \color{red}:=1 & & & \\\
7 [0,-1]+1 & [0,-1]+1 & & [-1,0]/[-1,-1] & (1+5^0.5)/2-[-1,0] \\\
8 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
9 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
10 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
11 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
12 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
13 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
14 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
15 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
16 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
17 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
18 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
19 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
20 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
21 [0,-1]+1 & [0,-1]+[0,-2] & & [-1,0]/[-1,-1] & d3+1-[-1,0] \\\
22 \end{spreadtab}
23 $

```

n	F_n	$\frac{F_n}{F_{n-1}}$	$\varphi - \frac{F_n}{F_{n-1}}$
1	1		
2	1	1	0.618033989
3	2	2	-0.381966011
4	3	1.5	0.118033989
5	5	1.666666667	-0.048632678
6	8	1.6	0.018033989
7	13	1.625	-0.006966011
8	21	1.615384615	0.002649374
9	34	1.619047619	-0.00101363
10	55	1.617647059	0.00038693
11	89	1.618181818	-0.000147829
12	144	1.617977528	0.000056461
13	233	1.618055556	-0.000021567
14	377	1.618025751	0.000008238
15	610	1.618037135	-0.000003146
16	987	1.618032787	0.000001202
17	1597	1.618034448	-0.000000459

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 `tabularx` stretched to fit 80% of the width of the line. The command `\multicolumn` has been widely used for formatting:

```

1 \nprounddigits2
2 \let\PC\%
3 \begin{spreadtab}{{tabularx}{0.8\linewidth}{|>\rule[-1.2ex]{0pt}{4ex}}X>{{\color{red}
   }}N42>{\color{red}}cN42>{\color{red}}cN42|}}
4 \hline
5 @Items & @\multicolumn{1}{c}{Price/U} & @\multicolumn{1}{c}{Qty} & @\multicolumn{1}{c}{
   Price} & @\multicolumn{1}{c}{Reduction} & @\textbf{Net} \\ \hline
6 @Item 1 & 5.99 & 20 & [-2,0]*[-1,0] & $-:={20}\PC$ & [-2,0]*(1-[-1,0]/100) \\
7 @Item 2 & 12 & 7 & [-2,0]*[-1,0] & $-:={10}\PC$ & [-2,0]*(1-[-1,0]/100) \\
8 @Item 3 & 4.50 & 40 & [-2,0]*[-1,0] & $-:={35}\PC$ & [-2,0]*(1-[-1,0]/100) \\
9 @Item 4 & 650 & 2 & [-2,0]*[-1,0] & $-:={15}\PC$ & [-2,0]*(1-[-1,0]/100) \\ \hline
10 @\multicolumn{6}{c}{\vspace{-1.5ex}} \\ \cline{4-6} % empty line and raise it a little
11 @\multicolumn{1}{c}{\rule[-1.2ex]{0pt}{4ex}} & @\multicolumn{2}{r|}{\textbf{Total}} & @
   sum(d2:[0,-2]) & @\multicolumn{1}{c}{$:={round((([1,0]/[-1,0]-1)*100,0)}\PC$} & @\
   fontseries{b}\selectfont:={sum(f2:[0,-2])} \\
12 \cline{4-6}
13 \end{spreadtab}

```

Items	Price/U	Qty	Price	Reduction	Net
Item 1	5,99	20	119,80	-20%	95,84
Item 2	12,00	7	84,00	-10%	75,60
Item 3	4,50	40	180,00	-35%	117,00
Item 4	650,00	2	1 300,00	-15%	1 105,00
Total			1 683,80	-17%	1 393,44

6.5 A magic square

```

1 \begin{spreadtab}{{tabular}{|*3>{\hfill\rule[-0.4cm]{0pt}{1cm}}$}m{0.7cm}<{\hfill\
   null}|}}
2 \hline

```

```

3 \color{red}:=2 & 5*b2-4*a1 & 3*a1-2*b2 \\\hline
4 2*a1-b2 & \color{red}:=-1 & 3*b2-2*a1 \\\hline
5 4*b2-3*a1 & 4*a1-3*b2 & 2*b2-a1 \\\hline
6 \end{spreadtab}

```

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.

```

1 \newlength\cellsize
2 \setlength\cellsize{1.5cm}
3 \newcolumntype{K}{@{>\rule{0pt}{2.5ex}\centering\arraybackslash$}p{\cellsize}<${@{}}
4 \begin{spreadtab}{\tabular}{*{8}{@{}p{.5\cellsize}@{}}}
5 \cline{4-5}
6 &&&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&&&\cline{3-6}
7 &&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&&\cline{2-7}
8 &\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&\multicolumn{2}{|K|}{:=[-1,1]+[1,1]}&\hline
9 \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
10 \end{spreadtab}

```

-5			
-1		-4	
-2	1	-5	
-5	3	-2	-3

*
* *

That's all, I hope we will find this package useful!

Given the youth of this package, please, be tolerant if you find bugs: with comments from users, better versions should be available soon. Indeed, this version is just beginning and, though rather stable, many things are still imperfect. Especially, it is obvious that many other macro-functions need to be written.

I thank you in advance for sending by [email](#) any bug you find, any macro-function or improvement you would like to be implemented, assumed 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