numerica-tables

version 3.1.0

Andrew Parsloe
(ajparsloe@gmail.com)

September 6, 2023

Abstract

The numerica-tables package defines a command \nmcTabulate which enables the creation of multi-column tables of mathematical function values. Key–value assignments allow presentation in a wide variety of table styles within the 'formal table' framework of the booktabs package. numerica-tables requires the numerica package to be loaded.

- This document applies to version 3.1.0 of `numerica-tables`.
- Version 3 of `numerica` needs to be loaded before `numerica-tables`; (`numerica` requires `amsmath`, `mathtools` and a reasonably recent LaTeX system).
- The `booktabs` package is required.
- I refer many times in this document to *Handbook of Mathematical Functions*, edited by Milton Abramowitz and Irene A. Stegun, Dover, 1965, abbreviated to *HMF*, and often followed by a reference, like Table 1.2, to a specific table.
- Version 3.1.0 of `numerica-tables`
  - adds an index to the documentation;
  - adds the ability to round table entries to different values depending on row or column (or both) in the table;
  - fixes two minor presentation bugs.
- Version 3.0.0 of `numerica-tables`
  - adds the ability to use numbers or expressions from
    - a comma list for row variable values, either as values or verbatim;
    - a macro containing a comma list for row variable values, either as values or verbatim;
    - a file for row variable values, either as values or verbatim;
  - adds the ability to use functions of a stepped variable to generate varyingly stepped row variable values;
  - adds the ability to suppress the header row;
  - is compatible with the additional features of `numerica` version 3.0.0,
    - including the decimal comma if the `comma` package option is used with `numerica`; and
    - fraction-form row variables, or fraction-form output if the `/` or `//` 'triggers' are used in the number-format option;
  - amends and adds to documentation.

Contents

1 Introduction **4**

1.1 Table structure . . . 4

1.2 Shared syntax . . . 5

2 \nmcTabulate settings **7**

2.1 Row-variable settings . . . 7

2.1.1 Row-variable specification: uniform case . . . 7

2.1.2 Row variable specification: non-uniform case . . . 9

2.1.2.1 rfunc . . . 10

2.1.2.2 rdata, rfile, rverb . . . 10

2.1.3 Formatting row variable values & column . . . 12

2.1.3.1 Rounding: rround . . . 12

2.1.3.2 Font: rfont . . . 13

2.1.3.3 Alignment: ralign . . . 13

2.1.3.4 Row-variable header: rhead . . . 14

2.1.3.5 Nudging the header: rhnudge . . . 15

2.1.3.6 Position in the table: rpos . . . 15

2.1.3.7 rvar', rhead', rhnudge' . . . 16

2.1.3.8 Fraction-form values: rfrac . . . 16

2.1.4 Adjoined multi-function tables . . . 17

2.2 Column-variable settings . . . 17

2.2.1 Column header formatting . . . 19

2.2.1.1 Single-column header . . . 19

2.2.1.2 Multi-column header: chstyle . . . 20

2.2.1.3 User-defined header: chead . . . 21

2.2.1.4 Alignment: calign . . . 22

2.2.1.5 Nudging header entries: chnudge . . . 22

2.2.1.6 Rounding: chround . . . 23

2.3 Multiple functions in a single table . . . 24

2.4 Whole-of-table formatting . . . 25

2.4.1 Title for function-value columns: ctitle . . . 25

2.4.2 Between header & title: csubttl . . . 27

2.4.3 Suppress/show header row . . . 28

2.4.4 Footer row: foot . . . 28

2.4.4.1 Footer functions 28
2.4.5 Horizontal rules: `rules` 28
2.4.6 Second row variable column: `rpos=4` 30
2.4.7 Separating blocks of rows: `rbloc` 31
2.4.7.1 Adjusting the extra space `rblocsep` 32
2.4.8 Table placement 33
2.4.8.1 Vertical alignment 33
2.5 Function value formatting 34
2.5.1 Trailing optional argument 34
2.5.1.1 Fraction-form output 34
2.5.1.2 Scientific notation 35
2.5.2 The `t` option 36
2.5.2.1 Padding the exponent: `(pad)` 36
2.5.3 Indicating signs outside the t-notation 38
2.5.4 Rounding to varying values 39
2.5.5 Differences: `diffs` 40
2.5.6 Formatting special values: `Q?` and `A!` 41
2.5.6.1 Star option: `\nmcTabulate*` 42
2.6 Other matters 44
2.6.1 Nesting 44
2.6.2 Saving tables to file 45
2.6.3 Viewing the LaTeX form 46

3 Reference summary 48
3.1 Commands defined in `numerica-tables` 48
3.2 Settings for `\nmcTabulate` 48

Index 51

Chapter 1

Introduction

Entering

```
\usepackage[<options>]{numerica}
\usepackage{numerica-tables}
```

in the preamble of a document gives access to a command for creating tables of function values in a wide variety of styles. Contrary to previous practice, for version 3.0.0 of `numerica-tables`, the `numerica` package is not loaded automatically but must be loaded explicitly (as above), with options if desired, *before* `numerica-tables`. It is *essential* that the version of `numerica` loaded is version 3.

All tables are 'formal tables' in the sense of the `booktabs` package, which is loaded automatically. Such tables have no vertical rules and few horizontal rules.

1.1 Table structure

I take as my source of models of mathematical tables those presented in *Handbook of Mathematical Functions*, edited by Milton Abramowitz and Irene A. Stegun, Dover, 1965, not because the typesetting is elegant (it often isn't) but because *HMF* displays a wide variety of table styles. The editors of that volume were faced with a host of different problems requiring a host of different solutions. The `\nmcTabulate` command of `numerica-tables` aims to reproduce most of those different solutions, within `booktabs` elegance.

To create a table we need to specify a function or functions to tabulate. The values a function takes will generally depend on a primary parameter and, possibly, a number of secondary parameters (which is where much of the complexity comes from). Mathematical tables are structured in *columns*. We (nearly always) read *down* a column as the primary parameter is incremented, generally in regular steps. We need to decide on the range of values the primary parameter will take and how fine-grained the tabulation will be – what the step size of its increments will be. Assigning different values to a second parameter generates a second, third,… column. Sometimes rather than a second parameter, a second, third, … function of the first parameter is tabulated in the successive columns.

In this document the first parameter is called the *row variable* – its value determines which row we are in; the second parameter, if present, is called the *column variable* – its value determines which column we are in. A table generally (but not always) presents the values of the row variable in the first column, the *row variable column*, sometimes in distinctive type (e.g. bolded). The values of the column variable are presented in a *header row* above the table body of function values. Above the header row there may be a *title row* and perhaps a *subtitle row* where other explanatory material can be displayed. Sometimes there is a *footer row* beneath the table body. Vertical rules are absent, horizontal rules used sparingly – for example, at the top and bottom of the table, or under the header row, but not in the body of the table.

1.2 Shared syntax

The `\nmcTabulate` command (short-name form `\tabulate`) shares the syntax of `\nmcEvaluate` (see `numerica.pdf`). When all options are used the command looks like

```
\nmcTabulate*[settings]{expr.}[vv-list][num. format]
```

1. `*` optional switch; if present ensures a single number output with no formatting, or an appropriate error message if the single number cannot be produced; see §2.5.6.1;

2. `[settings]` comma-separated list of *key=value* settings; this option is at the heart of creating a table of function values; see Chapter 2;

3. `{expr.}` mandatory argument specifying the mathematical expression or expressions in LaTeX form to be tabulated;

4. `[vv-list]` comma-separated list (or semicolon-separated list if the `comma` package option is used with `numerica`) of *variable=value* items, in particular containing the initial value of the row variable, and column variable if one is used;

5. `[num. format]` optional format specification for presentation of the numerical results (rounding, padding with zeros, scientific notation, fraction-form output); see §2.5.1.

Unlike `\nmcEvaluate` (the main command in `numerica`), for `\nmcTabulate` the two apparently optional arguments straddling the main argument (`settings` and `vv-list`) are *essential*. Although both are delimited by square brackets, that is in order to draw on the `numerica` code. Each argument contains items *necessary* for the construction of any table of function values.

Should `numerica` be loaded with the `comma` package option, numbers in tables will be displayed with a decimal comma. In this case items in the vv-list *must* be separated by a semicolon since the decimal comma is likely to feature there. Similarly, n-ary functions – `\max`, `\min` and `\gcd` – must use the semicolon as their argument separator.

Math environments are significant for `\nmcEvaluate`. They are essentially irrelevant for `\nmcTabulate`, although an external environment may determine how a table is positioned on the page (`\[ <table> \]` for instance will centre the table between the margins). Environments specified in the main argument (the formula) should be avoided. `$ $`, `\( \)` or `\[ \]` will not cause error, but a `\begin`–`\end` environment there will.

Just as the syntax is shared, so the settings available to the command `\nmcEvaluate` in `numerica` are also available to `\nmcTabulate`, although not all will be relevant. See the associated document `numerica.pdf` for a list of these and associated discussion. I will point out instances of their use in the following examples.

Chapter 2

\nmcTabulate settings

In addition to the shared settings, \nmcTabulate has many settings specific to it. They are discussed in groups in subsequent sections, some in more than one place. For the main discussion of row variable settings, see §2.1; for column variable settings see §2.2; for whole-of-table formatting see §2.4; for formatting the function values in table cells see §2.5.

2.1 Row-variable settings

2.1.1 Row-variable specification: uniform case

Deciding on a function to tabulate (entered in the main or mandatory argument of \nmcTabulate) will inevitably also mean deciding on the tabulation variable, the *row* variable, **rvar**, what value to start tabulating from – *that* is specified in the vv-list and so does not need a specific key – what value to tabulate to, **rstop**, and how fine-grained the tabulation is to be, the step size **rstep**. In the uniform case (which makes up the overwhelming majority of cases in *HMF* for instance) the step size is constant. It does not change as the value of the row variable changes. (The non-uniform case, available from version

Table 2.1: Row-variable specification

key	type	meaning	comment
rvar	token(s)	row variable	
rstep	real num.	step size	
rstop	real num.	stop value	use only one of rstop
rows	int	number of rows	or rows
rspec	comma list	{**rvar**, **step**, **rows**}	short form spec.

3.0.0 of `numerica-tables`, is discussed in §2.1.2 below. Quite different keys are required.)

The two tables in the example below tabulate $\sin x$ and $\cos x$ between 0 and 1 in increments of 0.2. Note that the start value of the tabulation variable is entered in the vv-list. The reason for placing it there is that for more complicated functions other parameters in the function and therefore in the vv-list may depend on the row variable. Although it will often be the first entry in the vv-list, it does not need to be. The initial value of the row variable may depend on other quantities which must necessarily precede it – lie to the right of it – in the list.

In the vv-list, the start value of the row variable may be a LaTeX expression. Both `rstep` and `rstop` may also be LaTeX expressions. They are evaluated *after* the vv-list is evaluated and so may depend on the values of variables in the vv-list, including the *initial* value of the row variable. Thus setting `rstep=1/x`, where `x` is the row variable, will give a *constant* step size equal to the reciprocal of the *initial* value of the row variable.

The difference in appearance of the two tables results from padding with zeros in the second (the asterisk in the trailing optional argument has the same effect in `\nmcTabulate` as in `\nmcEvaluate`). As you can see, padding applies not only to the values of the function but also to the values of the row variable, although that has been padded to only 1 decimal place rather than the 6 of the function values. Padding makes an obvious improvement to the appearance. (How many digits to pad to in the row variable column is discussed in §2.1.3.)

```
\tabulate[rvar=x,rstep=0.2,rstop=1]
  { \sin x }[x=0]\qquad
\tabulate[rvar=x,rstep=0.2,rstop=1]
  { \cos x }[x=0][*]
```

$\Longrightarrow$

x	$\sin x$
0	0
0.2	0.198669
0.4	0.389418
0.6	0.564642
0.8	0.717356
1	0.841471

x	$\cos x$
0.0	1.000000
0.2	0.980067
0.4	0.921061
0.6	0.825336
0.8	0.696707
1.0	0.540302

Sometimes (perhaps often) it may prove more convenient to specify the number of rows, `rows`, to tabulate rather than a stop value. Only one of `rows` and `rstop` should be given, but if both (inadvertently) are present, it is the value of `rows` that prevails. The first of the following three tables shows an example where `rows` is specified.

The second and third tables use an abbreviated form of the row variable specification, `rspec`. This is a three-element comma list, `{rvar,rstep,rows}`. The second table gives a straightforward example. In the third table a LaTeX expression has been inserted for `rows` in the `rspec` comma list. Like `rstep` and `rstop`, `rows` can be a LaTeX expression, but it is evaluated *before* the vv-list and therefore, unlike `rstep` and `rstop`, cannot depend on quantites specified there like the initial row variable value.

```
\tabulate[rvar=x,rstep=0.2,rows=6]
  { \sin x/\cos x }[x=0][*] \qquad
\tabulate[rspec={x,0.2,6}]
  { \tan x }[x=0][*] \qquad
\tabulate[rspec={x,0.2,1+(1/0.2)}]
  { \sqrt{\sec^2 x - 1} }[x=0][*]
```

$\Longrightarrow$

x	$\sin x/\cos x$
0.0	0.000000
0.2	0.202710
0.4	0.422793
0.6	0.684137
0.8	1.029639
1.0	1.557408

x	$\tan x$
0.0	0.000000
0.2	0.202710
0.4	0.422793
0.6	0.684137
0.8	1.029639
1.0	1.557408

x	$\sqrt{\sec^2 x - 1}$
0.0	0.000000
0.2	0.202710
0.4	0.422793
0.6	0.684137
0.8	1.029639
1.0	1.557408

2.1.2 Row variable specification: non-uniform case

Occasionally one wants to form a table in which the row variable does not increase or decrease in regular steps; for examples, see *HMF* Tables 1.1 and 3.1. (Tables 9.7, 10.5, 10.10 use two step values and could also be handled with these settings.) For instance, one might want a table of values of simple functions of a list of constants, or a table of function values at π, $\pi/2$, $\pi/3$, $\pi/4$, …, or at 1, 10, 100, 1000, …, or a table of function values at thoroughly irregular, perhaps experimentally determined, values.

`numerica-tables` provides two means of specifying such row variables, either by means of a row variable function (`rfunc`), when the row variable values

Table 2.2: Non-uniform row variable specification

key	type	meaning	comment
`rdata`	comma list	list of row-var. values	comma list may be stored in a macro
`rfile`	chars	filepath/name	file of comma separated values
`rverb`	int (0/1)	display `rdata` or `rfile` values verbatim	initialized to `0`
`rfunc`	token(s)	function of a stepped variable	

change in a non-uniform but formulaic way, or by explicitly listing the row variable values in a comma list (`rdata`, `rfile`). In this latter case, a setting `rverb` enables the row variable column to be displayed either as a sequence of *values* or verbatim as a sequence of *expressions* – like fractions of π.

2.1.2.1 `rfunc`

Suppose – perhaps with an interest in the distribution of prime numbers – that we want to create a small table of values of $n/\ln n$ for, say, $n = 10$, 100, 1000, 10000, ... The prospective row variable n is not increasing in constant steps, although clearly in a regular manner. We handle such cases with the `rfunc` setting; in the present instance `rfunc=10^n` where now `n` does increment by a constant amount:

```
\tabulate[rpos=1,rfunc=10^n,rvar=n,rstep=1,
            rows=7,rround=0]
 { n/\ln n }[n=1][0]
```

$\Longrightarrow$

n	$n/\ln n$
10	4
100	22
1000	145
10000	1086
100000	8686
1000000	72382
10000000	620421

The variable `n` has two different meanings here. Initially it is the variable of a simple step function incrementing by 1 at each step. The function takes values `10^n`. Once the table is compiled however `n` is used to denote these successive function values, $10, 100, \dots, 10000000$. To the *reader* of the table, only this latter meaning is evident. Yes, it would be possible to add further keys to specify a distinct step function variable and its start, step and stop values, but the list of keys to specify a table is already large. The 'double usage' implemented, perhaps confusing initially, economizes on keys and is invisible to the reader of the table. The initial value `n=1` in the vv-list applies to the row variable function `10^n`, not to the function being tabulated (so the error-producing expression `1/\ln 1` does not arise).

2.1.2.2 `rdata`, `rfile`, `rverb`

One source of difficulty in reading the previous table is working out just how many zeros there are in the larger numbers in the left column. It would be more readable 'at a glance' if we could write those in scientific notation. To do that we use the `rdata` and `rverb` keys. In the first of the examples below, `rverb` is absent (corresponding to the default `rverb=0`); in the second `rverb=1`, the effect of which is to use verbatim the row variable values provided in the `rdata` comma list:

```
\tabulate[rdata={10,100,1000,10^4,10^5,10^6,10^7},
    rvar=n,rround=0,ralign=l]
 { n/\ln n }[0]\qquad
\tabulate[rdata={10,100,1000,10^4,10^5,10^6,10^7},
    rverb=1,rvar=n,rround=0,ralign=l]
 { n/\ln n }[0]
```

$\Longrightarrow$

n	$n/\ln n$
10	4
100	22
1000	145
10000	1086
100000	8686
1000000	72382
10000000	620421

n	$n/\ln n$
10	4
100	22
1000	145
10^4	1086
10^5	8686
10^6	72382
10^7	620421

`rverb` is particularly useful if you want to make a table of simple functions of constants. In the following example, with its initial setting (`rverb=0`), the constants would be listed in the first column as numbers and repeated in the second column (perhaps to a different number of decimal places), which would all be rather pointless. With `rverb=1`, the constants are listed verbatim in the first column against their numerical values in the second. In the example, the data has been stored in macros, `\mydatai` and `\mydataii`, prior to calling the `\tabulate` command. `\mydataii` uses the `\sfrac` command from the `xfrac` package. By setting `rdata` equal to these macros, the `\tabulate` command gains access to the values stored in them.

```
\def\mydatai{\tfrac12\pi,\tfrac13\pi,\tfrac23\pi,
             \tfrac14\pi,\tfrac34\pi}
\renewcommand\arraystretch{1.2}
\tabulate[rdata=\mydatai,rverb=1,rvar=k,headless=1]
  { k }[6*]
\renewcommand\arraystretch{1} \qquad
\def\mydataii{\sfrac\pi2,\sfrac\pi3,\sfrac{2\pi}3,
              \sfrac\pi4,\sfrac{3\pi}4}
\tabulate[rdata=\mydataii,rverb=1,rvar=k,headless=1]
   { k }[6*]
```

$\Longrightarrow$

$\tfrac12\pi$	1.570796
$\tfrac13\pi$	1.047198
$\tfrac23\pi$	2.094395
$\tfrac14\pi$	0.785398
$\tfrac34\pi$	2.356194

${}^{\pi}/_{2}$	1.570796
${}^{\pi}/_{3}$	1.047198
${}^{2\pi}/_{3}$	2.094395
${}^{\pi}/_{4}$	0.785398
${}^{3\pi}/_{4}$	2.356194

The setting `headless=1` (see §2.4.3) means the tables have no header row. To accommodate the tower of `\tfrac`-s in the first column of the first table, `\arraystretch` has been used to add more space between the rows (and re-set afterwards). Through the virtues of `\sfrac` it is unnecessary in the second table which is neater and more readable to my eye. But even better is the example at §2.1.3.8.

Note that the row variable can be chosen arbitrarily – I earlier used `n` and have used `k` here but it could be anything. Nor does the function tabulated need to be the identity – see the earlier $n/\ln n$ example or a later example in a multi-column setting in §2.2.1.2. The identity function was appropriate for tables showing the decimal values of symbolic constants, as was the `rverb=1` setting. The `rverb` setting applies *only* to the `rdata` and `rfile` keys. It has no effect otherwise.

In addition to a comma list or macro, data for the row variable can also be stored in a file of comma-separated values – say `mydata.txt`. If `mydata.txt` is placed in the directory of the current document and `rfile=mydata.txt` entered in the settings option of the `\tabulate` command, the file will be found and the values used for the row variable. Alternatively, the file could be placed in your `texmf` tree and your TeX distro alerted to its presence (by refreshing the filename database). Again `rfile=mydata.txt` in the settings option will ensure the file is found and the contents used for the row variable values. Or, the file could be stored elsewhere and the `rfile` key equated to the full path and filename – something like `rfile=e:/mydocs/mydatafiles/mydata.txt`. This ensures the file will be found and the contents used for the row variable values. Note that even in Windows systems (where file paths use backslashes) the path requires that forward slashes only be used.

2.1.3 Formatting row variable values & column

The padding option (*) of the trailing optional argument is one way of formatting the row variable values, but to how many decimal places? Aligned left or right or centred? Under what heading – the example tables so far have simply used the row variable for the header? And should the row variable column be at the left of the table, or the right – or both? These and related questions are answered by assigning values to the keys listed in Table 2.3.

2.1.3.1 Rounding: `rround`

After studying some of the previous tables, we might decide to adjust the step size, say from 0.2 to 0.25. But changing `rstep` to the new value gives a disconcerting result. `numerica-tables` uses a default rounding value of 1 for the row variable and has rounded 0.25 down to 0.2, then $0.2 + 0.25 = 0.45$ down to 0.4, then $0.4 + 0.25 = 0.65$ down to 0.6, then $0.6 + 0.25 = 0.85$ down to 0.8, at which point it stops since $0.8 + 0.85 > 1$ which is the stopping value. The rounded-down values are the values used for caclulating the sines. The second table corrects matters by adjusting the row variable rounding with `rround=2`.

Table 2.3: Formatting the row variable column

key	type	meaning	initial
`rround`	int	rounding	`1`
`rfont`	chars	font (`\math<chars>`)	
`ralign`	char (`r/c/l`)	horizontal alignment	`r`
`rhead`	tokens	header	`rvar`
`rhnudge`	int	nudge header `<int>` mu	`0`
`rpos`	int (0. . . 4)	column placement	`1`
`rvar'`	tokens	2nd row variable col. spec.	`rvar`
`rhead'`	tokens	header of 2nd rv col. (if it exists)	`rvar'`
`rhnudge'`	int	nudge 2nd rv col. header `<int>` mu	`0`
`rfrac`	int (0. . . 5)	fraction form	`0`

```
\tabulate[rvar=x,rstep=0.25,rstop=1]
  { \sin x }[x=0][*] (Eh???) \quad
\tabulate[rvar=x,rstep=0.25,rstop=1,rround=2]
  { \sin x }[x=0][*]
```

$\Longrightarrow$

x	$\sin x$
0.0	0.000000
0.2	0.198669
0.4	0.389418
0.6	0.564642
0.8	0.717356

(Eh???)

x	$\sin x$
0.00	0.000000
0.25	0.247404
0.50	0.479426
0.75	0.681639
1.00	0.841471

2.1.3.2 Font: `rfont`

In the second table below bolding by means of the setting `rfont=bf` has been applied to emphasize the distinction between the row variable values and the function values. Possible values for this key are those characters that can be adjoined to `\math` to give a meaningful result. Thus other valid values are `it` (italic), `sf` (sans serif), `tt` (typewriter); `frak` (Fraktur); also `rm` (roman) is available, but that is the default.

2.1.3.3 Alignment: `ralign`

By default, the alignment of all columns is to the right, as in previous examples. This lends itself to neat output when padding with zeros is activated (the `*` in the trailing argument) and when some values are negative – the minus signs interfere with neat output in left or centred alignments. But in a case like the second table in the last example, you might prefer to centre the headers for both the row and function-value columns. These alignments are independently set. For the row variable column the default alignment is to the right `ralign=r`; `ralign=l` (lowercase L) aligns entries in the row variable column to the left, and `ralign=c` centres entries in the row variable column. The tables of the next example use a `c` alignment to centre the row variable column header. The third of those tables shows how minus signs spoil the effect.

2.1.3.4 Row-variable header: `rhead`

In the second and third tables, the header for the row variable column has also been bolded. The default header is the row variable symbol. That can be replaced by giving a value to the key `rhead`. I have used `rhead=\boldsymbol{x}` (rather than `\mathbf{x}`) in order to get an italicized bold symbol. Note that you do not need to include math delimiters in the specification. It is assumed that `rhead` will sit between `$ $` delimiters which are inserted automatically by `numerica-tables`.

```
\tabulate[rvar=x,rstep=0.25,rstop=1,
          rround=2,ralign=c]
  { \sin x }[x=0][*]\qquad
\tabulate[rvar=x,rstep=0.25,rstop=1,rround=2,
          ralign=c,rfont=bf,rhead=\boldsymbol{x}]
  { \sin x }[x=0][*]\qquad
\tabulate[rvar=x,rstep=0.25,rstop=0.5,rround=2,
          ralign=c,rfont=bf,rhead=\boldsymbol{x}]
  { \sin x }[x=-0.5][*]
```

$\Longrightarrow$

x	$\sin x$
0.00	0.000000
0.25	0.247404
0.50	0.479426
0.75	0.681639
1.00	0.841471

$\boldsymbol{x}$	$\sin x$
0.00	0.000000
0.25	0.247404
0.50	0.479426
0.75	0.681639
1.00	0.841471

$\boldsymbol{x}$	$\sin x$
−0.50	−0.479426
−0.25	−0.247404
0.00	0.000000
0.25	0.247404
0.50	0.479426

In these tables the row variable column has been given a centred alignment. The third table shows what goes wrong when *some* values in the row variable column are negative. Better then is to use padding, a right alignment (the default), and to use a phantom in the header. The first table below does this. The second table incorporates kerning into the header to achieve the same effect:

```
\tabulate[rvar=x,rstep=0.25,rstop=0.5,rround=2,
          rfont=bf,rhead=\boldsymbol{x}\hphantom{0}]
  { \sin x }[x=-0.5][*]\qquad
\tabulate[rvar=x,rstep=0.25,rstop=0.5,rround=2,
          rfont=bf,rhead=\boldsymbol{x}\mkern 9 mu]
  { \sin x }[x=-0.5][*]
```

$\Longrightarrow$

$\boldsymbol{x}$	$\sin x$
$-\mathbf{0.50}$	-0.479426
$-\mathbf{0.25}$	-0.247404
$\mathbf{0.00}$	0.000000
$\mathbf{0.25}$	0.247404
$\mathbf{0.50}$	0.479426

$\boldsymbol{x}$	$\sin x$
$-\mathbf{0.50}$	-0.479426
$-\mathbf{0.25}$	-0.247404
$\mathbf{0.00}$	0.000000
$\mathbf{0.25}$	0.247404
$\mathbf{0.50}$	0.479426

(To my eye, aligning the $\boldsymbol{x}$ above the first column of digits after the decimal point gives a better result than truly centring it in the column; compare these examples with the first two tables of the previous example.)

2.1.3.5 Nudging the header: `rhnudge`

However, you might prefer to avoid inserting positioning commands into the actual row variable header, obscuring its true content. You can avoid doing this by setting the key `rhnudge`.

The first table below reverts to the default right alignment, avoids any positioning commands in the row variable header, but instead nudges it into position with the setting `rhnudge=9`. For positive nudge values, nudging works in the *opposite* sense to the alignment. The units for nudging are mu (math units, 18 to a quad), but only a number – generally an integer – should be specified; the 'mu' is supplied by `numerica-tables`.

In the second table below the row variable takes single digit integer values, while the row variable name now occupies more than one character. With a right alignment the header would protrude out to the left. By giving `rhnudge` a *negative* value (`rhnudge=-12` in the example) it is brought back to a centred position in the row variable column.

```
\tabulate[rvar=x,rstep=0.25,rstop=0.5,rround=2,
          rfont=bf,rhead=\boldsymbol{x},rhnudge=9]
  { \sin x }[x=-0.5][4*]\qquad
\tabulate[rvar=x_{\text{int}},rstep=1,rstop=4,
          rround=0,rfont=bf,rhnudge=-12,
          rhead=\boldsymbol{x_{\text{int}}}]
  { \sin x_{\text{int}} }[x_{\text{int}}=0][4*]
```

$\Longrightarrow$

$\boldsymbol{x}$	$\sin x$
$-\mathbf{0.50}$	-0.4794
$-\mathbf{0.25}$	-0.2474
$\mathbf{0.00}$	0.0000
$\mathbf{0.25}$	0.2474
$\mathbf{0.50}$	0.4794

$\boldsymbol{x}_{\text{int}}$	$\sin x_{\text{int}}$
$\mathbf{0}$	0.0000
$\mathbf{1}$	0.8415
$\mathbf{2}$	0.9093
$\mathbf{3}$	0.1411
$\mathbf{4}$	-0.7568

2.1.3.6 Position in the table: `rpos`

By default, the row variable column is the *first* column of the table. Its position is determined by the value of the key `rpos`:

- `rpos=0`, suppressed (no row variable column);
- `rpos=1`, first column (the default);
- `rpos=2`, last column;
- `rpos=3`, first and last columns;
- `rpos=4`, first and last columns, with the values in the last column a user-defined function of the first; see §2.4.6;
- Any other integer acts like `rpos=1`.

An example with `rpos=3` is given shortly below, §2.3.

2.1.3.7 `rvar', rhead', rhnudge'`

These settings become relevant only when `rpos=4`; see §2.4.6.

2.1.3.8 Fraction-form values: rfrac

By giving the setting `rfrac` an integer value between 1 and 5 inclusive it is possible to render the row variable values as fractions with no more than `rround` digits in the denominator. Initially `rfrac=0`, which gives decimal output.

- `rfrac=1` produces a slash fraction like 2/3;
- `rfrac=2` produces a scalable `\frac`-tion like $\frac{2}{3}$ in textstyle and $\dfrac{2}{3}$ in displaystyle;
- `rfrac=3` produces a non-scalable `\tfrac` like $\tfrac{2}{3}$;
- `rfrac=4` produces a non-scalable `\dfrac` like $\dfrac{2}{3}$;
- `rfrac=5` produces a slash fraction like ${}^{2}\!/_{3}$ (by means of the `\sfrac` command) in text- and displaystyles if the `xfrac` package is loaded and like $_{2/3}$ if it isn't; when used as a super- or sub-script, the fractions reduce in size like $e^{{}^{2}\!/_{3}}$ or $e^{2/3}$.

The following table repeats an earlier table from §2.1.2.2, but more neatly, through using `\sfrac` (with the `rfrac=5` call) instead of `\tfrac`, and moving π from the row variable column into the formula:

```
\def\mydataii{1/2,1/3,2/3,1/4,3/4}
\tabulate[rdata=\mydataii,rpos=1,
          rvar=k,rfrac=5,rhnudge=6,chnudge=18]
  { k\pi }[k=0][6*]
```

$\Longrightarrow$

k	$k\pi$
$^1/_2$	1.570796
$^1/_3$	1.047198
$^2/_3$	2.094395
$^1/_4$	0.785398
$^3/_4$	2.356194

2.1.4 Adjoined multi-function tables

How might one tabulate multiple functions simultaneously? *HMF* has many, many examples where multiple functions (like the trigonometric or the hyperbolic functions) are tabulated in separate columns of the same table.

With the settings described so far, one way is to adjoin single column tables. In the tables below, which display as a single multi-columned table, I have used three different `rpos` settings (`rpos=1` is implicit in the first). This is one way to build a table that displays as multi-column. If you use this method, note that the `%` comment characters are essential at the end of the last argument of the `\tabulate` commands if you want the tables to abut exactly. Omitting them results in a space between the tables.

```
\tabulate[rspec={x,0.2,6}]
  { \sin x }[x=0][*]%
\tabulate[rpos=0,rspec={x,0.2,6}]
  { \cos x }[x=0][*]%
\tabulate[rpos=2,rspec={x,0.2,6}]
  { \tan x }[x=0][*]
```

$\Longrightarrow$

x	$\sin x$	$\cos x$	$\tan x$	x
0.0	0.000000	1.000000	0.000000	0.0
0.2	0.198669	0.980067	0.202710	0.2
0.4	0.389418	0.921061	0.422793	0.4
0.6	0.564642	0.825336	0.684137	0.6
0.8	0.717356	0.696707	1.029639	0.8
1.0	0.841471	0.540302	1.557408	1.0

However, tabulating more than one function at a time is too common a need to have to resort to a fudge like adjoining tables. `numerica-tables` offers a systematic way of doing this; see §2.3.

2.2 Column-variable settings

When a function of *two* variables is being tabulated, we generally think of one variable as the primary variable and the other as a parameter. To tabulate such a function, one way to proceed, as we have seen, is to create and adjoin separate tables, one per parameter value, but that is clumsy. A more systematic

Table 2.4: Column-variable specification

key	type	meaning	comment
`cvar`	token(s)	column variable	
`cstep`	real num.	step size	
`cstop`	real num.	stop value	either `cstop`
`cols`	int	number of columns	or `cols`
`cspec`	comma list	`{cvar,cstep,cols}`	short form spec.

procedure is to specify, in addition to the row variable, a *column* variable and its start, step and stop values.

In the following example `cvar=k` is the column variable. I have chosen a step size `cstep=2` and a stop value `cstop=9`. As with the row variable, the start value (`k=3`) of the column variable is specified in the vv-list. Although in the example these values are numbers, all three values could be LaTeX expressions that evaluate to numbers. In particular, the expressions for step and stop values may include the row and column variables (in the example x and k) which are assigned their initial vv-list values. Note also the setting for `rhead` which shows the reader of the table that the numerical values displayed in the column headers are values of `k`. This usage occurs throughout *HMF*.

```
\tabulate[rspec={x,0.2,6},rhead=x\backslash k,
          cvar=k,cstep=2,cstop=9]
  { \sin kx }[k=3,x=0][*]
```

$\Longrightarrow$

$x\backslash k$	3	5	7	9
0.0	0.000000	0.000000	0.000000	0.000000
0.2	0.564642	0.841471	0.985450	0.973848
0.4	0.932039	0.909297	0.334988	−0.442520
0.6	0.973848	0.141120	−0.871576	−0.772764
0.8	0.675463	−0.756802	−0.631267	0.793668
1.0	0.141120	−0.958924	0.656987	0.412118

Again, as with the row variable, rather than using an explicit stop value `cstop`, you might prefer to specify the number of columns, `cols`, explicitly. I could have replaced `cstop=9` with `cols=4` to get the same result. Note that the number of columns specified here is the number of *function-value* columns; the row variable column is ignored for this count.

It is worth pointing out explicitly that if `cols` is specified, then it is possible to have a *zero* step size, `cstep=0`. (A similar comment applies to `rows` and `rstep`.)

And again, as with the row variable, it is possible to condense the specification into a comma list with the key `cspec`. This is a 3-element comma list of the form {cvar,cstep,cols}. Thus, for the preceding table I could also have written

```
\tabulate[rspec={x,0.2,6},rhead=x\backslash k,
          cvar=k,cstep=2,cols=4]
  { \sin kx }[k=3,x=0][*]
```

or more succinctly

```
\tabulate[rspec={x,0.2,6},rhead=x\backslash k,
          cspec={k,2,4}]
  { \sin kx }[k=3,x=0][*]
```

and produced the same table.

As with the row equivalents, cstep, cstop and cols can all be LaTeX expressions. Again like the row equivalents, the first two are evaluated *after* the vv-list and so may depend not only on numbers and constants but also the initial values of the row and column variables, which are given those values in the vv-list. cols is evaluated *before* the vv-list; it may be a LaTeX expression but cannot depend on the row or column variable.

2.2.1 Column header formatting

There are four built-in style settings for the header to the column variable (or function-value) columns (the 'ch' prefix evoking 'column header'). If these don't meet your needs or otherwise satisfy, then it is possible to define your own header to the function value columns using the key chead. First I discuss the built-in styles.

2.2.1.1 Single-column header

When there is only one column of function values, the function being tabulated is by default set as the header to the column. This corresponds to setting ctitle=* (see §2.4.1 below). You may want some other header. Then give ctitle some other value (although note that giving it the value ** will set both

Table 2.5: Formatting the column variable header

key	type	meaning	initial
chstyle	int (0. . . 4)	header style	0
ctitle	tokens	single col. alt. header	
chead	tokens	user-defined header	
calign	char (r/c/l)	column alignment	r
chnudge	int	nudge header int mu	0
chround	int	rounding	0

the function and the vv-list as the header; again see §2.4.1). Whatever value you set, it will be typeset between math delimiters (`$` signs) and can be nudged (see §2.2.1.5) left or right to fine-tune its position in the column. (If you want an asterisk as the header, you will need to place it between *two* pairs of braces, `ctitle={{*}}`, to prevent it being misinterpreted as the default setting.)

If you want some more complicated header, perhaps not constrained by the `$` delimiters, then give `chead` a value. This key I discuss below in §2.2.1.3. `chead` is entirely up to the user to specify, including any math delimiters and positioning (nudging) of elements.

If both `ctitle` and `chead` are given, the `chead` value prevails.

2.2.1.2 Multi-column header: `chstyle`

`chstyle=0` which is the default gives a header of the form displayed in the last example, with only the column variable *value* at the head of each column. This style generally requires the row variable header to indicate what the values denote, as in the example above where `rhead=x\backslash k`, the backslash separating row from column variable. *HMF* contains a multitude of instances of this style; see Tables 9.7, 17.5, 21.1, 24.3, 27.4, etc. for examples.

`chstyle=1` changes the header of the *first* function value column to the form *variable=value* – in the example below, to $k = 1$. This may be an appropriate choice when a small rounding value is being used and the resulting columns are narrow. I can find only one real instance in *HMF*, Table 26.7. In the example I have used the `rdata` setting to collect an assortment of nonsense values and for some weird reason wish to tabulate the sines of multiples of these oddballs. Note that the row variable setting `rhead` (producing $X\backslash k$) is no longer needed since the column variable is now explicitly indicated. (But the table is lacking a title – what on earth are we calculating with this strange group of numbers?)

```
\tabulate[rdata={-e^2,1.234e2,3.1416,\pi/\gamma,1/9},
          ^,rvar=X,rverb=1,cspec={k,1,3},chstyle=1]
  { \sin kX }[k=1][3*]
```

$\Longrightarrow$

X	$k = 1$	2	3
$-e^2$	-0.894	-0.802	0.175
$1.234e2$	-0.769	0.983	-0.486
3.1416	0.000	0.000	0.000
π/γ	-0.745	-0.994	-0.581
$1/9$	0.111	0.220	0.327

`chstyle=2` changes the header of all function-value columns to the form *variable=value*. In *HMF* examples are Tables 7.4, 7.9, 10.10, 16.6, etc. Again, the row variable setting `rhead` no longer needs the `\backslash k` part since the column variable is now explicitly indicated.

```
\tabulate[rspec={x,0.2,6},
          cspec={k,2,3},chstyle=2]
  { \sin kx }[k=3,x=0][3*]
```

$\Longrightarrow$

x	$k = 3$	$k = 5$	$k = 7$
0.0	0.000	0.000	0.000
0.2	0.565	0.841	0.985
0.4	0.932	0.909	0.335
0.6	0.974	0.141	−0.872
0.8	0.675	−0.757	−0.631
1.0	0.141	−0.959	0.657

`chstyle=3` fills each column variable header with the expression being tabulated but with the column variable replaced by its respective values. See *HMF* Tables 5.4, 8.1, 9.1, 19.1, etc. for examples. Note that if the column variable value is `1`, the `1` will be displayed:

```
\tabulate[rspec={x,0.2,6},
          cspec={k,2,3},chstyle=3]
  { \sin kx }[k=1,x=0][4*]
```

$\Longrightarrow$

x	$\sin 1x$	$\sin 3x$	$\sin 5x$
0.0	0.0000	0.0000	0.0000
0.2	0.1987	0.5646	0.8415
0.4	0.3894	0.9320	0.9093
0.6	0.5646	0.9738	0.1411
0.8	0.7174	0.6755	−0.7568
1.0	0.8415	0.1411	−0.9589

In this last example you may not want the `1` displayed. To achieve that effect put `chstyle=4`. This results in a header as for `chstyle=3` but if the column variable takes the value `1`, it has an empty replacement:

```
\tabulate[rspec={x,0.2,6},
          cspec={k,2,3},chstyle=4]
  { \sin kx }[k=1,x=0][4*]
```

$\Longrightarrow$

x	$\sin x$	$\sin 3x$	$\sin 5x$
0.0	0.0000	0.0000	0.0000
0.2	0.1987	0.5646	0.8415
0.4	0.3894	0.9320	0.9093
0.6	0.5646	0.9738	0.1411
0.8	0.7174	0.6755	−0.7568
1.0	0.8415	0.1411	−0.9589

2.2.1.3 User-defined header: `chead`

If the function in the last example were, for instance, $k + \sin kx$, then neither replacing k by `1` nor an empty replacement would be appropriate. In that case the only recourse is to use the `chead` key. Users can assign whatever value they like to `chead`. The assignment must contain the correct number of tab characters (`&`) for the *column variable columns only.* It is a header only to the function-value columns. The user will need to insert `$` signs or other math delimiters as appropriate. This differs from the practice for `rhead`, but `chead` is potentially far more complicated. Thus for $k + \sin kx$,

```
\tabulate[rspec={x,0.2,6},cspec={k,2,3},
          chead=$1+\sin x$&$3+\sin3x$&$5+\sin 5x$]
  { k+\sin kx }[k=1,x=0][4*]
```

$\Longrightarrow$

x	$1 + \sin x$	$3 + \sin 3x$	$5 + \sin 5x$
0.0	1.0000	3.0000	5.0000
0.2	1.1987	3.5646	5.8415
0.4	1.3894	3.9320	5.9093
0.6	1.5646	3.9738	5.1411
0.8	1.7174	3.6755	4.2432
1.0	1.8415	3.1411	4.0411

Non-empty content for the `chead` key overrides any `chstyle` setting and, in the case of a table with only a single function-value column, overrides any `ctitle` setting.

2.2.1.4 Alignment: `calign`

The function-value columns are aligned right, `calign=r`, by default. Also available are `calign=c` for centred alignment and `calign=l` (lowercase L) for left alignment. Using centred alignment with `chstyle=2` in a previous example table gives

```
\tabulate[rspec={x,0.2,6},ralign=c,
          cspec={k,2,3},chstyle=2,calign=c]
  { \sin kx }[k=3,x=0][*]
```

$\Longrightarrow$

x	$k = 3$	$k = 5$	$k = 7$
0.0	0.000000	0.000000	0.000000
0.2	0.564642	0.841471	0.985450
0.4	0.932039	0.909297	0.334988
0.6	0.973848	0.141120	−0.871576
0.8	0.675463	−0.756802	−0.631267
1.0	0.141120	−0.958924	0.656987

The first column of function values looks better, but the minus signs spoil the effect in the others. Handling signs in tables is discussed below; see §2.5.2.1.

2.2.1.5 Nudging header entries: `chnudge`

In left or right alignment it is possible to nudge the column headers in the opposite direction by giving a numerical value to the the key `chnudge`. The header is moved by the specified number of mu (math units; 18 to a quad). Note that the ‘mu’ does not need to be written. `numerica-tables` provides that. In the next example I have chosen `chnudge=12` to nudge the column headers to the left to give a centred effect to the header but leaving the function values with their (potentially) awkward minus signs right aligned.

```
\tabulate[rspec={x,0.2,6},ralign=c,
          cspec={k,2,3},chstyle=2,chnudge=12]
  { \sin kx }[k=3,x=0][*]
```

$\Longrightarrow$

x	$k=3$	$k=5$	$k=7$
0.0	0.000000	0.000000	0.000000
0.2	0.564642	0.841471	0.985450
0.4	0.932039	0.909297	0.334988
0.6	0.973848	0.141120	−0.871576
0.8	0.675463	−0.756802	−0.631267
1.0	0.141120	−0.958924	0.656987

The `chnudge` value does not need to be positive. Negative nudges can be useful when a column header is *longer* than the rounded function values. In the second example below, I’ve reduced the rounding value for function values to 3, and chosen an initial k value of 100 to ensure this circumstance. To centre the column headers I have used `chnudge=-9`.

```
\tabulate[rspec={x,0.2,6},ralign=c,
          cspec={k,2,3},chstyle=2,chnudge=-9]
  { \sin kx }[k=100,x=0][3*]
```

$\Longrightarrow$

x	$k=100$	$k=102$	$k=104$
0.0	0.000	0.000	0.000
0.2	0.913	1.000	0.929
0.4	0.745	0.041	−0.688
0.6	−0.305	−0.998	−0.419
0.8	−0.994	−0.081	0.999
1.0	−0.506	0.995	−0.322

2.2.1.6 Rounding: `chround`

In the examples so far, the column variable has incremented in integer steps. The default rounding value for the column variable is 0 (for the row variable it is 1), so if it increments by some non-integer amount, the result will be confusing – if k incremented by, say, 0.25, starting from $k = 3$, then the next column would also have a header $k = 3$ (since 3.25 with a rounding value 0 rounds to 3). The appropriate key to remedy this state of affairs is `chround`. For a step size of 0.25 the appropriate setting is `chround=2`.

```
\tabulate[rspec={x,0.2,6},ralign=c,
          cspec={k,0.25,3},chstyle=2,chround=2]
  { \sin kx }[k=3,x=0][*]
```

$\Longrightarrow$

x	$k = 3.00$	$k = 3.25$	$k = 3.50$
0.0	0.000000	0.000000	0.000000
0.2	0.564642	0.605186	0.644218
0.4	0.932039	0.963558	0.985450
0.6	0.973848	0.928960	0.863209
0.8	0.675463	0.515501	0.334988
1.0	0.141120	−0.108195	−0.350783

2.3 Multiple functions in a single table

As already noted in §2.1.4, tabulating more than one function at a time is too common a need to have to resort to a fudge like adjoining tables. The systematic way of handling this task is to enter the functions in the main argument of a `\tabulate` command separated by a specified mark then alert `\tabulate` that this has happened with the `ff` key in the settings option.

By default the multi-function delimiter is the comma if the decimal point is a dot (or period), or the semicolon if the decimal point is a comma (`numerica` loaded with the `comma` package option). If you are content with the default delimiter then it suffices to enter `ff` in the settings option. If not, then enter `ff=<mark>` there. For example `ff=|` would make the 'pipe' character `|` the multi-formula delimiter. If the `ff` key is overlooked then multiple formulas in the main argument of `\tabulate` will almost certainly cause a LaTeX error.

In the following example, using the default comma, note first the `ff` setting, and then the `o` setting indicating that the arguments of the trig functions are in degrees and (just to amuse myself) I have put the row variable column on both sides with the `rpos=3` setting:

```
\tabulate[ff,o,rpos=3,rround=0,
          rvar=\theta,rstep=10,rstop=90]
  { \sin \theta, \cos \theta }[\theta=0][*]
```

$\Longrightarrow$

θ	$\sin\theta$	$\cos\theta$	θ
0	0.000000	1.000000	0
10	0.173648	0.984808	10
20	0.342020	0.939693	20
30	0.500000	0.866025	30
40	0.642788	0.766044	40
50	0.766044	0.642788	50
60	0.866025	0.500000	60
70	0.939693	0.342020	70
80	0.984808	0.173648	80
90	1.000000	0.000000	90

The tables suggest a space saving possibility: since sin and cos are complementary functions ($\cos\theta = \sin(90-\theta)$), the values in the bottom half of the table duplicate values in the top half, only with the columns reversed. This is the reason for the space saving `rpos=4` setting (§2.4.6) which enables complementary functions to be tabulated in 'half tables' (*HMF* Tables 4.10–4.12 are examples for the trigonometric functions).

In the next example, the row variable column is again duplicated, left and right, with the `rpos=3` setting, and a centred alignment is used for the function values. Because sin and cos are complementary, I have stopped the table at `rstop=45` since continuing to `90` would simply give a mirror reflection of the preceding values. To accommodate the use of the comma in `\max` and `\min` I have stipulated `ff=|`. Visually, the table has an unsatisfactory, sprawling appearance – which directs attention to how tables might be titled (§2.4.1):

```
\tabulate[ff=|,o,rpos=3,rround=0,
    rvar=\theta,rstep=10,rstop=45,calign=c]
  { \max(\sin \theta,\cos \theta)|
         \min(\sin \theta,\cos \theta) }
  [\theta=0][*]
```

$\Longrightarrow$

θ	$\max(\sin\theta,\cos\theta)$	$\min(\sin\theta,\cos\theta)$	θ
0	1.000000	0.000000	0
10	0.984808	0.173648	10
20	0.939693	0.342020	20
30	0.866025	0.500000	30
40	0.766044	0.642788	40

2.4 Whole-of-table formatting

There are a number of settings pertaining to the appearance of the table as a whole, things like the position of the row variable column, division of the function values into blocks to aid readability, the presence of horizontal rules or of a collective column title or of a footer row. I discuss these here.

2.4.1 Title for function-value columns: `ctitle`

The function-value columns have individual headers, formatted in the various ways provided by the settings discussed in previous sections, but it can also be helpful to have a collective title for these columns. We saw the need in the last example. The need is met with the `ctitle` key. This can be set to whatever you like (e.g. `ctitle=\text{Fred}`) but, to more purpose, I shall use the setting to improve the look of the table in the last example. Note that the content of the `ctitle` key is enclosed in braces. This is to shield the commas there from being misinterpreted as item separators in the settings option. For an analogous reason I have specified the semicolon (`ff=;`) as the function separator in the main argument:

Table 2.6: Table formatting

key	type	meaning	initial
`ctitle`	token(s)	collective title for function-value columns	
	token(s)	subtitle row for function-value cols	
`header`	int (`0`/`1`)	suppress/show header row	`1`
`foot`	token(s)	table-wide footer row	
`rules`	char(s)	horizontal rule spec.	`ThB`
`rpos`	int (`0`...`4`)	row variable col. position(s)	`1`
`rbloc`	integer comma list	row block specification	
`valign`	char (`t`/`m`/`b`)	vertical alignment of table relative to text baseline	`m`

```
\tabulate[ff=;,o,rpos=3,rround=0,
          rvar=\theta,rstep=10,rstop=45,
          ctitle={\max,\min(\sin\theta,\cos\theta)},
          chead=max\hphantom{00} & min\hphantom{00}]
  { \max(\sin \theta,\cos \theta);
        \min(\sin \theta,\cos \theta) } [\theta=0][*]
```

$\Longrightarrow$

	$\max,\min(\sin\theta,\cos\theta)$		
θ	max	min	θ
0	1.000000	0.000000	0
10	0.984808	0.173648	10
20	0.939693	0.342020	20
30	0.866025	0.500000	30
40	0.766044	0.642788	40

Now it is clearer what is being tabulated and the wide separation of the function-value columns is reduced.

There are two built-in values for the `ctitle` key: `ctitle=*`, which forms the title from the function being tabulated, and `ctitle=**` which uses the function and vv-list for the title. Obviously these, particularly the latter, could easily become too long to be useful. An example of `ctitle=**` is presented below in §2.4.5. In the following example, inclusion of the vv-list in the title is pointless since the variables there form the row and column variables of the table; `ctitle=*` is all that is needed:

```
\tabulate[rspec={n,1,5},rround=0,
          cspec={m,1,4},chstyle=2,ctitle=*]
  { \cos(m\pi/n) }[n=3,m=2][*4]
```

			$\cos(m\pi/n)$		
	n	$m=2$	$m=3$	$m=4$	$m=5$
	3	−0.5000	−1.0000	−0.5000	0.5000
$\Longrightarrow$	4	0.0000	−0.7071	−1.0000	−0.7071
	5	0.3090	−0.3090	−0.8090	−1.0000
	6	0.5000	0.0000	−0.5000	−0.8660
	7	0.6235	0.2225	−0.2225	−0.6235

2.4.2 Between header & title: `csubttl`

Some tables need to fit more header or title material into their rows than can be comfortably accommodated in either row alone. For examples, see *HMF* Tables 7.9 (error function for complex arguments), 17.7 (Jacobian zeta function), 21.1 (eigenvalues of spheroidal wave functions), and 26.7 (probability integrals). One way of handling this problem is to resort to more complicated environments in header and title rows. Another, more direct way, is to insert a row between the header and title rows.

I have chosen `csubttl`, a contraction of 'column variable subtitle' for the key name. (In version 2 of `numerica-tables` the name `cmidrow` was used; that will still work but `csubttl` gives a clearer indication of what the key does.) The initial 'c' emphasizes that like `chead` and `ctitle` it is constrained to the span of the column variable (or function-value) columns only. The content of `csubttl` is entirely the responsibility of the user, including insertion of the necessary number of tab characters, `&`, and any math delimiters required.

```
\tabulate[ff,rspec={x,1,7},rround=0,chnudge=18,
  ctitle=\text{Hyperbolic functions},
  csubttl=\multicolumn{3}{c}{$\sinh x=\tfrac12
    (e^x-e^{-x}),\ \cosh x=\tfrac12(e^x+e^{-x})$}]
{ \tfrac12e^x, \sinh x, \cosh x }
  [{x}=0][8*]
```

			Hyperbolic functions	
			$\sinh x = \tfrac{1}{2}(e^x - e^{-x}),\ \cosh x = \tfrac{1}{2}(e^x + e^{-x})$	
	x	$\tfrac{1}{2}e^x$	$\sinh x$	$\cosh x$
	0	0.50000000	0.00000000	1.00000000
$\Longrightarrow$	1	1.35914091	1.17520119	1.54308063
	2	3.69452805	3.62686041	3.76219569
	3	10.04276846	10.01787493	10.06766200
	4	27.29907502	27.28991720	27.30823284
	5	74.20657955	74.20321058	74.20994852
	6	201.71439675	201.71315737	201.71563612

2.4.3 Suppress/show header row

Usually the header row in a table is shown. It carries essential information as to the table's contents. However, there are occasions when it should be suppressed. An example where this is appropriate is given in §2.1.2.2 where a table listing fractions of π and their values is shown. As in that example, to suppress the header enter the setting `headless=1`. (Otherwise tables default to `headless=0`.)

2.4.4 Footer row: `foot`

Some tables have a footer row and `numerica-tables` allows such a row to be inserted, but its entire content, with one exception, is the responsibility of the user, including insertion of the necessary number of tab characters `&`. This will usually be 1 less than the total number of columns (including row variable columns) in the table – or some adjustment thereof if you use `\multicolumn`. (*HMF* uses the footer mainly for cryptic descriptions of the accuracy and needs of interpolation methods.)

You can put into the footer what you wish with the setting `foot=<tokens>`.

The one exception is when `foot=*`. This will fill the footer with the header, but with the items of the header presented in *reversed* order – the last item first, and so on. This is useful for tabulating complementary functions like the sine and cosine or, more generally, $f(x)$ and $g(x)$ where $g(x) = f(k - x)$ for some constant k. Values for the complementary function are read from the bottom up and require a reversed row variable column on the right of the table; see §2.4.6.

2.4.4.1 Footer functions

In previous versions of `numerica-tables` it was possible to perform certain simple operations on columns – calculate the sum, the average and maximum and minimum values. This is no longer so in version 3. Not only does it seem tangential to the primary function of the `\tabulate` command but it was also acutely dependent on the format of the numbers being operated on. A simple change in the number-format option could cause a LaTeX error.

2.4.5 Horizontal rules: `rules`

The `booktabs` package which `numerica` uses is most emphatic that one should '1. Never, ever use vertical rules. 2. Never use double rules.' Most of the tables proper in *HMF* lack rules of any kind although closer inspection shows smaller tables within the text generally *are* delimited by horizontal rules (often also with vertical rules). In the various examples in the present document I have used horizontal rules because these too are tables within text. Some form of delineation seems necessary. (Although many of *HMF*'s tables are inelegantly typeset, I have used it as a valuable source for the variety of structures that one might need for presenting a multitude of different kinds of numerical data.)

The `rules` key allows one to specify precisely which rules are used. The content of the key is a 'word' – a sequence of letters – where the characters have the significance and default thicknesses (from `booktabs`) shown in Table 2.7. The default setting is `rules=ThB`. To insert a rule beneath the title, for example, change this to `rules=TthB`.

If you are using a subtitle row between header and title rows and want a rule beneath that too, then the spec. is `rules=TtshB`. (For legacy reasons, `m` – from 'midrow' – can also be used instead of `s`, as in version 2 of `numerica-tables`.) To my eye rules beneath both title and subtitle don't work; a rule beneath the subtitle alone gives a better result. The subtitle rule changes its behaviour depending on whether there are two row variable columns – on the left and right of the table – or not. If there is only such column then, like the title rule, the subtitle rule spans only the function-value columns. If there are two row variable columns then the subtitle rule spans the table but is trimmed by 0.5 em at each end. That degree of trim is the `booktabs` default but can be changed by giving a different value to `\cmidrulekern` in the preamble, e.g. `\cmidrulekern=1em`. Note that the changed trim will also apply to the title rule.

If you are using a footer row and want a rule above it, then add `f` to the specification, e.g. `rules=TthfB`. In version 3 of `numerica-tables` the rule is trimmed at each end. Visually, having two table-spanning rules close together, the `f` and `B` rules, doesn't work. The trimming makes a difference. (For the `T` and `h` rules, the occurrence of the table body beneath the `h` rule seems to make a difference to the visual impact of the rules.) But the question should always be: is a rule necessary at all? Usually, less is more.

If you wish to change the thickness of a rule from its default, then enter new values for any or all of `\heavyrulewidth`, `\lightrulewidth`, `\cmidrulewidth` in the preamble. The values listed in Table 2.7 are the default values in the `booktabs` package (except for the midrow and footer rules, which `booktabs` does not cover; in `numerica-tables` these rules are assigned a thickness of `\cmidrulewidth`).

Table 2.7: Rules. (In the 'span' column, 'f-v'=function-value; 'r-v'=row variable; '< table' indicates that the rule spans the table but is trimmed at each end.)

char	rule	position	span	default rule thickness
`T`	top	above table	table	`\heavyrulewidth=.08em`
`t`	title	below title	f-v cols	`\cmidrulewidth =.03em`
`s`	subtitle	below subtitle	f-v cols (if 1 r-v col.) < table (if 2 r-v cols)	`\cmidrulewidth =.03em`
`h`	header	below header	table	`\lightrulewidth=.05em`
`f`	footer	above footer	< table	`\cmidrulewidth =.03em`
`B`	bottom	below table	table	`\heavyrulewidth=.08em`

The order in which rules are placed in the specification doesn't matter. I have entered them in their 'natural' order simply because it feels natural to do so, but it is their occurrence in the spec., not their position, that matters.

In the example table below, a rule for the column title has been specified (the `t` in the setting `rules=TthB`). Also note the use of `ctitle=**`. The formula contains an extra parameter a, assigned a value in the vv-list. It now makes sense to display the vv-list in the column title (but note the braces around `k` and `x` in the vv-list so that *they* don't display).

```
\tabulate
  [rspec={x,0.25,5},rround=2,rhnudge=9,
     cspec={k,0.25,3},chstyle=2,chround=2,
       ctitle=**,rules=TthB]
  { a\sin kx }[a=2/\pi,{k}=3,{x}=0][*]
```

⟹

	$a\sin kx, \quad (a = 2/\pi)$		
x	$k = 3.00$	$k = 3.25$	$k = 3.50$
0.00	0.000000	0.000000	0.000000
0.25	0.433945	0.462191	0.488633
0.50	0.635025	0.635685	0.626425
0.75	0.495337	0.412111	0.314439
1.00	0.089840	-0.068879	-0.223316

2.4.6 Second row variable column: `rpos=4`

In §2.1.3.6 I discussed the settings `rpos=0,1,2` and in §2.3 gave an example of using `rpos=3` where repeating the row variable column on the right is helpful. There is another value available for this key, `rpos=4`. Like `rpos=3` this adds the row variable column to both left and right sides of the table, but for the right column the values are a function of those in the left column (`rpos=3` corresponds to the function being the identity). The value given to the key `rvar'` determines the function used and the value given to the key `rhead'` determines the header for the right-hand row variable column. If `rhead'` is omitted it defaults to a blank header, unless the `rvar'` setting is also omitted, when `rpos=4` behaves like `rpos=3`.

For example, the sine and cosine are complementary functions; when working in degrees, $\cos\theta = \sin(90-\theta)$. We can exploit this fact to halve the table size needed to tabulate the two functions. In the table, $\theta' = 90 - \theta$ and `rhead'=\theta'`. Simply to illustrate the use of `rhnudge'` I have nudged the header in the second (right) row variable column to sit above the tens digits of the row variable values. The example also gives an illustration of the use of an expression in the third element of `rspec`.

```
\tabulate[ff=;,o,rpos=4,
  rspec={\theta,5,1+45/5},rround=0,
    chnudge=14,rvar'=90-\theta,rhnudge'=4,
    rhead'=\theta',rules=ThfB,foot=*]
  { \sin\theta;\cos\theta }[\theta=0][*]
```

$\Longrightarrow$

θ	$\sin\theta$	$\cos\theta$	θ'
0	0.000000	1.000000	90
5	0.087156	0.996195	85
10	0.173648	0.984808	80
15	0.258819	0.965926	75
20	0.342020	0.939693	70
25	0.422618	0.906308	65
30	0.500000	0.866025	60
35	0.573576	0.819152	55
40	0.642788	0.766044	50
45	0.707107	0.707107	45
θ'	$\cos\theta$	$\sin\theta$	θ

The values of sines from 0 to 45 degrees are read downwards from the first column of function values, and from 45 to 90 degrees are read upwards from the second column of function values. For cosines it is downwards from the second column and upwards from the first column. The reversed footer line indicates the change of columns to use. In the example note

- the setting of `rvar'` to a function `90-\theta` of the row variable;
- the footer setting `foot=*` to obtain the reversed header in the footer;
- the rule *above* the footer row specified by the `f` added to the `rules` setting, `rules=ThfB`.

Note also the degree setting `o` in the settings option.

Although there is a significant space saving with tables like this (see *HMF* Tables 4.10, 4.11, 4.12), they are not 'kind to the reader'. They require a certain concentration to read and in my view should be avoided unless space is seriously constrained. *HMF* Tables 6.1 and 6.2 are tables of the gamma function and its relatives where $y = x - 1$ is used in the row variable column on the right (stemming from $y! = \Gamma(x-1)$); *HMF* Table 6.5 in effect uses $\langle 1/x \rangle$ (the nearest integer to $1/x$) for the row variable on the right.

2.4.7 Separating blocks of rows: `rbloc`

Readability of long columns of figures can be aided by breaking the columns into blocks with extra white space between blocks of rows. This is achieved with the `rbloc` key:

```
rbloc = <comma list of positive integers>
```

specifies how many rows belong to each block. For example, `rbloc={5,5,6}` breaks the table into blocks of 5 rows, 5 rows, then 6 rows. If the number of rows in the table is greater than the sum of the entries in the comma list, then division into blocks continues as specified by the last entry in the comma list. Thus `rbloc=5` (strictly `rbloc={5}` but the braces can be omitted in this case since no comma is enclosed) divides a table into blocks of 5 rows; `rbloc={1,5}` divides a table into 1 row followed by blocks of 5 rows. A division of this kind may be appropriate when, say, the row variable runs from 0 to 1 in increments of 0.1 – there are 11 rows of which the first (when the row variable is zero) may have distinctive values.

The pull of the nice round number

However, this is not how *HMF* sets out its tables. The dominant practice in *HMF* is division into blocks of (generally) 5 rows, many of which start with a zero value for the row variable. Rather than isolate this initial value, they include it in the first block of 5, then continue with blocks of 5 until a single isolated row is left at the bottom of the page or the table. There seems to be a psychological need to finish a page or table with the row variable set to a nice round number. Thus: tabulate from 0 to 10 rather than 0 to 9, from 0 to 1 rather than 0 to 0.9, and even from 0 to 30 or 0 to 2 rather than 0 to 29 or 0 to 1.9. Using blocks of 5 the consequence is that there is always an isolated line at the end – a kind of punctuation mark to signal the end of the page or the table.

In the next example I have divided the rows into blocks of 5 by means of the setting `rbloc=5`.

```
\tabulate[ff=;,o,rspec={\theta,10,1+90/10},
          rround=0,rbloc=5]
    { \sin\theta; \cos\theta }[\theta=0][*]
```

	θ	$\sin\theta$	$\cos\theta$
	0	0.000000	1.000000
	10	0.173648	0.984808
	20	0.342020	0.939693
	30	0.500000	0.866025
$\Longrightarrow$	40	0.642788	0.766044
	50	0.766044	0.642788
	60	0.866025	0.500000
	70	0.939693	0.342020
	80	0.984808	0.173648
	90	1.000000	0.000000

2.4.7.1 Adjusting the extra space `rblocsep`

By default `numerica` sets the extra space between blocks of rows at `1 ex`. This value can easily by changed with the setting `rblocsep=<length>`. The units need to be included in the specification.

2.4.8 Table placement

Tables can be nudged vertically with the LaTeX commands `\bigskip`, `\medskip\medskip`, `\smallskip`, usually about 1, $^1/_2$ and $^1/_4$ line spaces (with stretch and shrink). `booktabs` provides `\abovetopsep` and `\belowbottomsep`, both set by default to `0ex` and easily changed by writing, e.g., `\abovetopsep=1.25ex` if you want to insert `1.25ex` of space above the table (perhaps to fit captions).

2.4.8.1 Vertical alignment

By writing `valign=<char>` where `<char>` is one of `t`, `m` or `b` the vertical alignment of the table can be set relative to the text baseline. `valign=t` aligns the top of the table with the text baseline, `valign=b` the bottom of the table with the text baseline, `valign=m` aligns the middle of the table with the text baseline. By default `valign=m` is set. Repeating an example from earlier (§2.1) I have added letters A, B, C to show where the baseline is. In the first table the top of the table aligns with the baseline; in the second table (default case) the middle of the table aligns with the baseline; in the third table, the bottom of the table aligns with the baseline.

```
A \tabulate[valign=t,rvar=x,rstep=0.2,rows=6]
  { \sin x/\cos x }[x=0][*] \quad
B \tabulate[rspec={x,0.2,1+(1/0.2)}]
  { \tan x }[x=0][*] \quad
C \tabulate[valign=b,rspec={x,0.2,(6)}]
  { \sqrt{\sec^2 x - 1} }[x=0][*]
```

$\Longrightarrow$ A

x	$\sin x/\cos x$
0.0	0.000000
0.2	0.202710
0.4	0.422793
0.6	0.684137
0.8	1.029639
1.0	1.557408

B

x	$\tan x$
0.0	0.000000
0.2	0.202710
0.4	0.422793
0.6	0.684137
0.8	1.029639
1.0	1.557408

C

x	$\sqrt{\sec^2 x - 1}$
0.0	0.000000
0.2	0.202710
0.4	0.422793
0.6	0.684137
0.8	1.029639
1.0	1.557408

As explained in §2.1.4, tables can be adjoined to give the appearance of a single larger table. If tables with different numbers of rows are adjoined in this manner, then a middle alignment fails and a top alignment is necessary (so that the header rows of the tables align).

Table 2.8: Formatting function values

key	type	meaning	initial
`(pad)`	int	t-notation phantom padding	
`signs`	int	sign handling for function-values	`0`
`diffs`	int	insert differences & pre-pad with zeros	`0`
`round`	tokens	row or col. dependent rounding value	
`Q?`	tokens	special cell conditional	
`A!`	tokens	special cell formatting	

2.5 Function value formatting

In previous tables in this document, function values have generally been limited to a fairly narrow range of values. What happens when function values span orders of magnitude? Can we accommodate scientific notation, expressly designed to cope with such orders of magnitude, in a natural way? Can we round rows or columns to different rounding values? Or, in a different direction, can we form tables of function values in fraction form?

2.5.1 Trailing optional argument

The primary tool for function-value formatting is the trailing optional argument of the `\tabulate` command where the rounding value is specified, padding with zeros is set or not (generally *set* in tables), scientific notation is set or not, and fraction-form output can be specified.

2.5.1.1 Fraction-form output

In §2.1.3.8 we saw how to display the row variable in fraction form. Function values can also be presented in that form. The problem is that such output requires far more computation than other output since finding denominators at the specified accuracy is an iterative process that needs to be performed for every function value. However, it is feasible for small tables. In the tables below, approximations to small positive and inverse powers of π are listed to 2 and 4 decimal places of accuracy. It is interesting that all the powers listed can be approximated to 4-place accuracy by 3-figure denominators (and π^2 by a 2-figure denominator).

```
\def\mydataiii{\pi,\pi^2,\pi^3,
    \pi^{\sfrac12},\pi^{\sfrac13}}
\tabulate[rdata=\mydataiii,rverb=1,rpos=1,
    rvar=k,ralign=l,chead={\small $2$ places}]
```

```
  { k }[2/s] \qquad
\tabulate[rdata=\mydataiii,rverb=1,rpos=1,
    rvar=k,ralign=l,chead={\small $4$ places}]
  { k }[4/s]
```

$\Longrightarrow$

k	2 places
π	$^{22}/_{7}$
π^2	$^{148}/_{15}$
π^3	$^{2760}/_{89}$
$\pi^{1/2}$	$^{23}/_{13}$
$\pi^{1/3}$	$^{19}/_{13}$

k	4 places
π	$^{355}/_{113}$
π^2	$^{227}/_{23}$
π^3	$^{4930}/_{159}$
$\pi^{1/2}$	$^{257}/_{145}$
$\pi^{1/3}$	$^{186}/_{127}$

A second example shows that all four built-in constants to `numerica` and their first few inverse powers can be approximated to 5 decimal places with 3-figure denominators:

```
\tabulate[rdata={\pi ,e,\phi,\gamma},rverb=1,rvar=k,
    cspec={n,1,4},chstyle=3,chnudge=9,rules=TthB,/max=1000,
    ctitle=\lvert k^{\sfrac1n}-p/q \rvert<0.5\times10^{-5}]
  { k^{\sfrac1n} }[n=1,k=1][/s5]
```

$\Longrightarrow$

	$\lvert k^{1/n}-p/q\rvert < 0.5\times 10^{-5}$			
k	$k^{1/1}$	$k^{1/2}$	$k^{1/3}$	$k^{1/4}$
π	$^{355}/_{113}$	$^{296}/_{167}$	$^{517}/_{353}$	$^{667}/_{501}$
e	$^{1264}/_{465}$	$^{582}/_{353}$	$^{1210}/_{867}$	$^{217}/_{169}$
ϕ	$^{610}/_{377}$	$^{491}/_{386}$	$^{668}/_{569}$	$^{397}/_{352}$
γ	$^{228}/_{395}$	$^{487}/_{641}$	$^{194}/_{233}$	$^{421}/_{483}$

2.5.1.2 Scientific notation

Elegant scientific notation, set with an `x` in the trailing optional argument, is generally not appropriate for use in tables; see the first table below. Repeating the `x` – `xx` – in the trailing optional argument (the second table) so that scientific notation extends to numbers in the range $[1,10)$ helps, particularly with the *left* alignment chosen for the function-value column, but the result is wasteful of space and the repetition of the '×10' is distracting and would be more so for a larger table. The `x` specification should be used in tables, if at all, only for *small* tables and special cases. The `t` option is much preferred; see §2.5.2 below.

```
\tabulate[rspec={x,1,2*3+1},rround=0]
  { e^x}[x=-5][*x]\qquad
\tabulate[rspec={x,1,2*3+1},rround=0,calign=l]
  { e^x}[x=-3][*xx]
```

⟹

x	e^x
-3	4.978707×10^{-2}
-2	1.353353×10^{-1}
-1	3.678794×10^{-1}
0	1.000000
1	2.718282
2	7.389056
3	2.008554×10^{1}

x	e^x
-3	4.978707×10^{-2}
-2	1.353353×10^{-1}
-1	3.678794×10^{-1}
0	1.000000×10^{0}
1	2.718282×10^{0}
2	7.389056×10^{0}
3	2.008554×10^{1}

2.5.2 The t option

HMF uses a special notation for coping with function values spanning orders of magnitude. This notation can be invoked by inserting `t` in the trailing optional argument. Repeating the previous two tables, and adding a `chnudge` value, gives a more compact and visually appealing result:

```
\tabulate[rspec={x,1,2*3+1},rround=0,chnudge=24]
  { e^x}[x=-3][*t]\qquad
\tabulate[rspec={x,1,2*3+1},rround=0,chnudge=24]
  { e^x}[x=-3][*tt]
```

⟹

x	e^x
-3	$(-2)\,4.978707$
-2	$(-1)\,1.353353$
-1	$(-1)\,3.678794$
0	1.000000
1	2.718282
2	7.389056
3	$(1)\,2.008554$

x	e^x
-3	$(-2)\,4.978707$
-2	$(-1)\,1.353353$
-1	$(-1)\,3.678794$
0	$(0)\,1.000000$
1	$(0)\,2.718282$
2	$(0)\,7.389056$
3	$(1)\,2.008554$

2.5.2.1 Padding the exponent: `(pad)`

In the second table of the last example some might quibble at the lack of alignment of the left parentheses. *HMF* tends to align these and `numerica-tables` offers the setting

```
(pad) = <integer>
```

to achieve the effect. (The parentheses are part of the key – a reminder of the `t`-form of scientific notation.) `<integer>` is the number of digits/characters to pad to. Repeating the last two tables with the setting `(pad)=2` produces the following results:

```
\tabulate[rspec={x,1,2*3+1},rround=0,
          chnudge=24,(pad)=2]
  { e^x}[x=-3][*t]\qquad
\tabulate[rspec={x,1,2*3+1},rround=0,
                 chnudge=24,(pad)=2]
  { e^x}[x=-3][*tt]
```

$\Longrightarrow$

x	e^x
-3	$(-2)\,4.978707$
-2	$(-1)\,1.353353$
-1	$(-1)\,3.678794$
0	1.000000
1	2.718282
2	7.389056
3	$(\ \ 1)\,2.008554$

x	e^x
-3	$(-2)\,4.978707$
-2	$(-1)\,1.353353$
-1	$(-1)\,3.678794$
0	$(\ \ 0)\,1.000000$
1	$(\ \ 0)\,2.718282$
2	$(\ \ 0)\,7.389056$
3	$(\ \ 1)\,2.008554$

Note that this setting is relevant only when the `t` option is used in the trailing number-formatting argument of the `\tabulate` command. Examples in *HMF* of the style exemplified by the first table are, among others, Tables 8.6, 9.2, 20.1, and of the style exemplified by the second table, among many, Tables 9.9, 10.5, 13.1, 14.1, 19.1.

Accommodating signs: `signs` Instead of e^x as the test function, use $e^x - 1$. Now there are positive, zero and negative function values to contend with. Recall that in the `t`-notation the *exponent* is the parenthesized integer part of a number and the *significand* the following decimal figures. `numerica-tables` offers the `signs` key to align (or not) the exponents. The setting is

```
signs = <integer>
```

Besides the do-nothing default `(signs=0)`, there are four effective values for `<integer>`:

- `signs=2` inserts a + sign between exponent and significand of every non-negative number;
- `signs=1` inserts a + sign between exponent and significand of every non-negative number that immediately precedes or follows a negative number;
- `signs=-1` inserts a + sign between exponent and significand of any non-negative number that immediately precedes or follows a negative number, and inserts a *phantom* + sign between exponent and significand of every other non-negative number;
- `signs=-2` inserts a *phantom* + sign between exponent and significand of every non-negative number;

In the following examples, `signs=-2`, `signs=-1` and `signs=2`, all give acceptable results.

```
\tabulate[rspec={x,1,2*3+1},rround=0,
          (pad)=2,signs=-2]
  { e^x-1}[x=-3][4*tt]\qquad
```

```
\tabulate[rspec={x,1,2*3+1},rround=0,
          (pad)=2,signs=-1]
  { e^x-1}[x=-3][4*tt]\qquad
\tabulate[rspec={x,1,2*3+1},rround=0,
          (pad)=2,signs=2]
  { e^x-1}[x=-3][4*tt]
```

$\Longrightarrow$

x	$e^x - 1$
-3	$(-1)\ -9.5021$
-2	$(-1)\ -8.6466$
-1	$(-1)\ -6.3212$
0	$(\ \ 0)\ \ \ 0.0000$
1	$(\ \ 0)\ \ \ 1.7183$
2	$(\ \ 0)\ \ \ 6.3891$
3	$(\ \ 1)\ \ \ 1.9086$

x	$e^x - 1$
-3	$(-1)\ -9.5021$
-2	$(-1)\ -8.6466$
-1	$(-1)\ -6.3212$
0	$(\ \ 0)\ +0.0000$
1	$(\ \ 0)\ \ \ 1.7183$
2	$(\ \ 0)\ \ \ 6.3891$
3	$(\ \ 1)\ \ \ 1.9086$

x	$e^x - 1$
-3	$(-1)\ -9.5021$
-2	$(-1)\ -8.6466$
-1	$(-1)\ -6.3212$
0	$(\ \ 0)\ +0.0000$
1	$(\ \ 0)\ +1.7183$
2	$(\ \ 0)\ +6.3891$
3	$(\ \ 1)\ +1.9086$

In *HMF* Table 23.2 illustrates `signs=-2`; Tables 10.1, 13.1, 14.1, 19.1 among many others illustrate `signs=-1`; and Tables 9.4, 10.6, 20.2, 22.11 among others illustrate `signs=2`.

`signs=1`, however, is an inappropriate setting for these function values in the `t`-notation:

```
\tabulate[rspec={x,1,2*3+1},rround=0,
          (pad)=2,signs=1]
  { e^x-1}[x=-3][4*tt] \qquad
```

$\Longrightarrow$

x	$e^x - 1$
-3	$(-1)\ -9.5021$
-2	$(-1)\ -8.6466$
-1	$(-1)\ -6.3212$
0	$(\ \ 0)\ +0.0000$
1	$(\ \ 0)\ 1.7183$
2	$(\ \ 0)\ 6.3891$
3	$(\ \ 1)\ 1.9086$

2.5.3 Indicating signs outside the t-notation

The `signs` key is not limited to the `t`-notation. In the following tables where the notation is not used, positive values for the key, including `signs=1`, give good results (I've included also the default setting – the third table):

```
\tabulate[rspec={x,0.1,9},(pad)=2,signs=2]
  { 10\sin 5x}[x=-0.4][*4]\qquad
\tabulate[rspec={x,0.1,9},(pad)=2,signs=1]
  { 10\sin 5x}[x=-0.4][*4]\qquad
\tabulate[rspec={x,0.1,9},(pad)=2]
  { 10\sin 5x}[x=-0.4][*4]
```

$\Longrightarrow$

x	$10\sin 5x$
-0.4	-9.0930
-0.3	-9.9749
-0.2	-8.4147
-0.1	-4.7943
0.0	$+0.0000$
0.1	$+4.7943$
0.2	$+8.4147$
0.3	$+9.9749$
0.4	$+9.0930$

x	$10\sin 5x$
-0.4	-9.0930
-0.3	-9.9749
-0.2	-8.4147
-0.1	-4.7943
0.0	$+0.0000$
0.1	4.7943
0.2	8.4147
0.3	9.9749
0.4	9.0930

x	$10\sin 5x$
-0.4	-9.0930
-0.3	-9.9749
-0.2	-8.4147
-0.1	-4.7943
0.0	0.0000
0.1	4.7943
0.2	8.4147
0.3	9.9749
0.4	9.0930

HMF seems to use `signs=2` when the sign of the function values changes every few entries and `signs=1` when there are runs of entries of the same sign. Over the range tabulated here for $10\sin 5x$, they would use the middle table of the three, `signs=1`.

2.5.4 Rounding to varying values

Above, in §2.5.1.1, we created two tables of fraction-form approximations to simple power functions of π, one accurate to two places of decimals, one to four places. Version 3.1 of `numerica-tables` (as distinct from version 3.0) offers the means of producing tables with rounding values depending on position in the table. This is effected through the key

```
round = f(rvar,cvar)
```

where `f(rvar,cvar)` denotes a function of row and column variables. Usually this will mean dependence either on row variable or column variable rather than both. In the present instance we form a multi-function table and with the `round` key let the rounding value equal the row variable value `r` (`round=r`) to obtain fractional approximations to simple powers of π at rounding values from 1 to 5 (and discover that all these values can be approximated to 5 decimal places with 3 figure denominators – π^2 only just).

```
\tabulate[ff,rspec={r,1,5},round=r,/max=999,chstyle=2,
      ctitle=\lvert\pi^k-\sfrac mn\rvert<0.5\times10^{-r}]
  { \pi,\pi^2,\pi^3,\pi^{1/2},\pi^{1/3}}[r=1][/s]
```

$\Longrightarrow$

		$\lvert\pi^k - {}^m\!/_n\rvert < 0.5\times 10^{-r}$			
r	π	π^2	π^3	$\pi^{1/2}$	$\pi^{1/3}$
1	$^{19}/_{6}$	$^{59}/_{6}$	$^{31}/_{1}$	$^{7}/_{4}$	$^{3}/_{2}$
2	$^{22}/_{7}$	$^{148}/_{15}$	$^{2760}/_{89}$	$^{23}/_{13}$	$^{19}/_{13}$
3	$^{267}/_{85}$	$^{227}/_{23}$	$^{4589}/_{148}$	$^{39}/_{22}$	$^{41}/_{28}$
4	$^{355}/_{113}$	$^{227}/_{23}$	$^{4930}/_{159}$	$^{257}/_{145}$	$^{186}/_{127}$
5	$^{355}/_{113}$	$^{9840}/_{997}$	$^{14821}/_{478}$	$^{296}/_{167}$	$^{517}/_{353}$

Another place where a variable rounding value can be of value is when a function being tabulated changes slowly for each step in the row variable value;

the value of the cosine for instance changes from 1.0000 to 0.9848 between 0° and 10°. *Part* of a table of the cosine might be something like the following, where values in the initial rows of the table are rounded to a higher value than in later rows. `round` is set to an expression involving the row variable in boolean elements `\theta<11` and `\theta>10` which evaluate to 0 or 1 so that `round` takes the value 6 for the initial rows of the table and the value 4 thereafter.

```
\tabulate[o,rspec={\theta,1,6},calign=l,chnudge=15,
          round=6(\theta<11)+4(\theta>10)]
  { \cos\theta }[\theta=8]
```

	θ	$\cos\theta$
	8	0.990268
	9	0.987688
$\Longrightarrow$	10	0.984808
	11	0.9816
	12	0.9781
	13	0.9744

2.5.5 Differences: `diffs`

In fine-grained tables where function values change only slowly from entry to entry it can be helpful to include a difference entry between function-value entries as an aid to interpolation (and a test of eyesight). By entering

```
diffs = <non-negative integer>
```

the `\tabulate` command will include differences in a table. The `<non-negative integer>` is the maximum number of digits in a difference.

```
\tabulate[rspec={x,0.01,1+(1.05-1)/0.01},rround=2,
  rhnudge=9,chnudge=21,diffs=3]
    { \sinh x }[x=1][*4]
```

	x	$\sinh x$	
	1.00	1.1752	
	1.01	1.1907	155
$\Longrightarrow$	1.02	1.2063	156
	1.03	1.2220	157
	1.04	1.2379	159
	1.05	1.2539	160

I have deliberately chosen the function and settings here – particularly `diffs=3` – to give a good result. With the default right alignment of the function-value columns, it is easy to get this wrong. The evidence will be either in the misalignment of the first row of function values or unnecessary padding of differences with leading zeros. It is a good idea to create your table first, see how function values change between successive rows and judge how many digits there will be in a difference. In the following examples I have deliberately put `diffs=2` and `diffs=4` to show the effect of a misjudgement. In the first table the first row of function values is misaligned by one character. (`diffs=1` would have produced a two-character misalignment.) In the second table the unnecessary fourth digit for the differences results in pre-padding with 0.

In the second table the function $-\sinh x$ is *decreasing*, showing how it is the *absolute value of the difference* between successive function values that is tabulated. A difference is always a non-negative value.

```
\tabulate[rspec={x,0.01,1+(1.05-1)/0.01},rround=2,
  rhnudge=9,chnudge=21,diffs=2]
    { \sinh x }[x=1][*4]\qquad
\tabulate[rspec={x,0.01,1.05},rround=2,
  rhnudge=9,chnudge=30,diffs=4]
    { -\sinh x }[x=1][*4]
```

$\Longrightarrow$

x	$\sinh x$
1.00	1.1752
1.01	1.1907 ^{155}
1.02	1.2063 ^{156}
1.03	1.2220 ^{157}
1.04	1.2379 ^{159}
1.05	1.2539 ^{160}

x	$-\sinh x$
1.00	−1.1752
1.01	−1.1907 ^{0155}
1.02	−1.2063 ^{0156}
1.03	−1.2220 ^{0157}
1.04	−1.2379 ^{0159}
1.05	−1.2539 ^{0160}

When the `diffs` setting is too small, function values in the first row are misaligned, the amount depending on how much too small. (A left alignment of the function value column is another way of tackling this issue.) When the `diffs` setting is too big, alignment is fine but differences are padded with unnecessary leading zeros, meaning the column header will need a bigger nudge to bring *it* into alignment.

2.5.6 Formatting special values: `Q?` and `A!`

You may wish to highlight or display in some special way a particular function value or values. `\nmcTabulate` has two related settings that enable this: `Q?=<tokens>` and `A!=<tokens>`. As the names suggest: Question? and Answer!

The question should be an expression that `l3fp` can digest and produce a boolean answer to (1 for 'true', 0 for 'false'). *This is not a LaTeX expression*; *this is an* `l3fp` *expression.*[1] For the user it should be enough to know that an expression formed from decimal numbers (but only with the dot decimal point), parentheses `( )`, the familiar arithmetic symbols, `+`, `-`, `*`, `/` and `^`, relation symbols `<`, `>`, `=` and combinations like `!=` (for $\neq$), `>=` (for $\geq$), and `<=` (for

[1] Documentation about `l3fp` can be found in `interface3.pdf`, which is part of the `l3kernel` bundle.

$\leq$) will be digested by `l3fp`. In addition there are `||` for logical Or, `&&` for logical And, and `!` for logical Not; `exp(1)` for e and `pi` (no backslash) for π. `numerica-tables` provides `MAX` and `MIN` for the maximum and minimum function values tabulated, and uses `@` to denote the current function value.

So, a query might be `Q?=@<0`, *Is the current function value negative?*, or `Q?={@>=pi}`, *Is the current function value greater than or equal to π?* (The braces hide the equality sign in the *key=value* settings option.) `Q?={@=MIN}` (again note the braces) is the question: *Is the current function value equal to the minimum function value for the whole table?*

The answer must be in the form of a LaTeX 2ε formatting statement, again using `@` to denote the current function value. Thus `A!=\mathbf{@}` is a valid answer; so is `A!=\color{red}{@}` (provided you have `\usepackage{color}` in the preamble); and so is `A!=(@)`. Another valid answer is `A!=` , meaning that function values satisfying the `Q?` question are omitted from the output.

This can be useful to suppress 'irrelevant' values in a particular context. For example, suppose we wish to focus on the values of $\cos(m\pi/n)$ lying between 0 and $\frac{1}{2}$ inclusive for certain values of m and n. Rather than cluttering the table with values outside that interval, we suppress them (the two occurrences of '`1e-14`' in the query are there to prevent rounding errors confusing the result):

```
\tabulate[rspec={n,1,1+(15-4)},rules=Tth,rround=0,
          rpos=2,cspec={m,1,1+(5-2)},chstyle=2,
          ctitle=*,Q?={@<-1e-14||@>0.5+1e-14},A!=]
  { \cos(m\pi/n) }[n=4,m=2][*4]
```

	$\cos(m\pi/n)$				
	$m=2$	$m=3$	$m=4$	$m=5$	n
	0.0000				4
	0.3090				5
	0.5000	0.0000			6
		0.2225			7
$\Longrightarrow$		0.3827	0.0000		8
		0.5000	0.1736		9
			0.3090	0.0000	10
			0.4154	0.1423	11
			0.5000	0.2588	12
				0.3546	13
				0.4339	14
				0.5000	15

2.5.6.1 Star option: \nmcTabulate*

If the `Q?` question is satisfied by at least one function value then adding a star (asterisk) to the `\tabulate` command will display the first such instance. Like other starred commands in the `numerica` suite (`\eval*`, `info*`, `\macros*`, `\constants*`, `\iter*`, `\solve*` and `\recur*`), `\tabulate*` outputs a single number. Using the star means you do not need an answering `A!` to the query `Q?` since no formatting of table values is involved.

```
\tabulate*[rspec={n,1,1+(15-4)},cspec={m,1,1+(5-2)},
           Q?={@<-1e-14||@>0.5+1e-14}]
  { \cos(m\pi/n) }[n=4,m=2][*4]
```

$\Longrightarrow$ 0.6235. Indeed, if you omit the `Q?` and `A!` settings from the previous table so that all function values are visible then this is the value that follows 0.5000 in the `m=2` column – the first function value encountered outside the interval $[0, 0.5]$.

If you want the *maximum* value that has been tabulated then, from version 3 of `numerica-tables`, you do not even need the query: when `\tabulate` is starred, `Q?` is initialized behind the scenes to `@=MAX`.[2] Thus, repeating the example from §2.2,

```
\tabulate[rspec={x,0.2,6},rhead=x\backslash k,
          cvar=k,cstep=2,cstop=9]
  { \sin kx }[k=3,x=0]
\tabulate*[rspec={x,0.2,6},rhead=x\backslash k,
           cvar=k,cstep=2,cstop=9]
  { \sin kx }[k=3,x=0]
```

$\Longrightarrow$

$x\backslash k$	3	5	7	9
0.0	0.000000	0.000000	0.000000	0.000000
0.2	0.564642	0.841471	0.985450	0.973848
0.4	0.932039	0.909297	0.334988	−0.442520
0.6	0.973848	0.141120	−0.871576	−0.772764
0.8	0.675463	−0.756802	−0.631267	0.793668
1.0	0.141120	−0.958924	0.656987	0.412118

0.985450

Errors If *no* function value satisfies a query then a message is generated:

```
\tabulate*[rspec={n,1,1+(15-4)},
           cspec={m,1,1+(5-2)},Q?=@>1]
  { \cos(m\pi/n) }[n=4,m=2][*4]
```

$\Longrightarrow$!!! No table value satisfies query `Q?` in: settings. !!!

Scientific notation If you want the number output in scientific notation when the star option is chosen, then enter the exponent mark in the trailing number-format option. This is straightforward for a letter like the commonly used `e`, but remember that if you enter the `x` option you will need to place the `\tabulate*` command between math delimiters, otherwise the `\times` symbol resulting from the `x` option will generate a LaTeX error ('Missing $ inserted'):

[2] In the unlikely event that someone *consistently* wanted some other query to be asked – the minimum value perhaps, or first negative value or . . . – please let the author know. It would be straightforward to use a package option to give a choice in this matter.

```
$
\tabulate*[rspec={n,1,1+(15-4)},cspec={m,1,1+(5-2)},
           Q?={@<-1e-14||@>0.5+1e-14},A!=]
  { \cos(m\pi/n) }[n=4,m=2][*4x]
$
```

$\Longrightarrow 6.2349 \times 10^{-1}$.

2.6 Other matters

Here I group items that do not fit naturally into the earlier categories.

2.6.1 Nesting

A `\tabulate` command can be nested within other commands from the `numerica` suite, and those other commands can be nested within a `\tabulate` command.

Occasionally one might want to extract a value from a table to insert in another command. This can be done by nesting a `\tabulate*` command with an appropriate `Q?` setting within the other command. In fact, from version 2 of `numerica` on, the star is unnecessary. All we require is that the `Q?` setting is satisfied by at least one tabulated function value.

```
\eval[env=$]{(\tabulate
  [rspec={n,1,15},cspec={m,1,5},Q?={@=MAX}]
    { \cos(m\pi/n) }[n=4,m=2][*4])\sinh t +
      (\tabulate[rspec={n,1,15},cspec={m,1,5},
                 Q?={@=MIN}]
         { \sin(m\pi/n) }[n=4,m=2][*4])\cosh t
            }[t=2][4]
```

$\Longrightarrow (0.9397)\sinh t + (-1.0000)\cosh t = -0.354, \quad (t = 2)$.

Forming the table

```
 \tabulate[rspec={n,1,15},rround=0,rpos=2,rules=Tth,
           cspec={m,1,5},ctitle=*,chstyle=2]
   { \cos(m\pi/n) }[n=4,m=2][*4]
```

for the cosine and the table

```
\tabulate[rspec={n,1,15},rround=0,rpos=2,rules=Tth,
          cspec={m,1,5},ctitle=*,chstyle=2]
  { \sin(m\pi/n) }[n=4,m=2][*4]
```

for the sine and checking the entries shows that indeed the maximum and minimum values are 0.9397 and −1.0000 respectively.

If the `Q?` setting is not satisfied by any function value a familiar error message is shown – with a tweak:

```
\eval{$ (\tabulate
  [rspec={n,1,15},cspec={m,1,5},Q?=@>2]
     { \cos(m\pi/n) }[n=4,m=2][*4])\sinh t
      $}[t=2][4]
```

$\Longrightarrow$!!! No table value satisfies query `Q?` in: settings (2). !!!
Here, the `(2)` tells us that the message refers to a command at the second level, a *nested* command.

Perhaps a more likely situation is to want to nest other commands within a `\tabulate` command. I give an example in the documentation to the associated package `numerica-plus` around the timing of signals between points fixed on a rotating disk.

2.6.2 Saving tables to file

In earlier versions of `numerica-tables` it was possible to save a table to file, or a row or a column or a particular value from a table, by giving a *setting* `reuse` a value. From version 3.0.0, in the interests of simplifying use (and avoiding code complications) the `reuse` *setting* has been discontinued. The `\reuse` (or `\nmcReuse`) *command* remains (as part of the `numerica` package) and can be used to save the most recent table to file.

In the following example, a table is created and then saved to file and to the macro `\mytable` by the subsequent `\reuse` command:

```
\tabulate
  [rspec={x,0.25,5},rround=2,rhead=x,
    ralign=r,rhnudge=9,
     cspec={k,0.25,3},chstyle=2,
      chround=2,calign=r,ctitle=**,
       rules=TthB]
  { a\sin kx }[a=2/\pi,{k}=3,{x}=0][*]
\reuse{mytable}
```

$\Longrightarrow$

	$a\sin kx,\quad (a=2/\pi)$		
x	$k=3.00$	$k=3.25$	$k=3.50$
0.00	0.000000	0.000000	0.000000
0.25	0.433945	0.462191	0.488633
0.50	0.635025	0.635685	0.626425
0.75	0.495337	0.412111	0.314439
1.00	0.089840	−0.068879	−0.223316

Now test the content of the control sequence

`\mytable` $\Longrightarrow$

	$a\sin kx,\quad (a=2/\pi)$		
x	$k=3.00$	$k=3.25$	$k=3.50$
0.00	0.000000	0.000000	0.000000
0.25	0.433945	0.462191	0.488633
0.50	0.635025	0.635685	0.626425
0.75	0.495337	0.412111	0.314439
1.00	0.089840	−0.068879	−0.223316

Certainly `\mytable` contains the table.

If we use the `view` setting with `\reuse` we can see that `\mytable` and its contents have also been saved to file:

```
\reuse[view]{}
```

$\Longrightarrow$

```
saved:  \mytable {\begin {tabular}[m]{rrrr}\toprule &\multicolumn {3}{c}{$a\sin
        kx,\mskip 18muminus15mu(a=2/\pi )$}\\ \cmidrule (lr){2-4}$x\mkern
        9mu$&$k=3.00$&$k=3.25$&$k=3.50$\\ \midrule
        ${0.00}$&$0.000000$&$0.000000$&$0.000000$\\
        ${0.25}$&$0.433945$&$0.462191$&$0.488633$\\
        ${0.50}$&$0.635025$&$0.635685$&$0.626425$\\
        ${0.75}$&$0.495337$&$0.412111$&$0.314439$\\
        ${1.00}$&$0.089840$&$-0.068879$&$-0.223316$\\ \bottomrule \end
        {tabular}}
```

The file that `\mytable` is saved to is the `.nmc` file of the current document, hence `numerica-tables.nmc` in the present instance. The contents of this file can be edited in a text editor, or some limited file operations can be effected with the `\reuse` command. These have been described in the associated document `numerica.pdf`.

2.6.3 Viewing the LaTeX form

In previous versions of `numerica-tables` the `dbg` and `view` settings were disabled. In version 3, they have been enabled to the extent that the LaTeX form of a table can be viewed by entering either `dbg=11` or, less nerdishly, `view` into the settings option of `\nmcTabulate`. In the example I first create the table and then use the `view` setting:

```
\tabulate[view,rvar=x,rstep=0.2,rstop=1]
  { \sin x }[x=0]
```

⟹

```
LaTeX:   \begin {tabular}[m]{rr}\toprule $x$&$\sin x$\\ \midrule ${0}$&$0$\\
         ${0.2}$&$0.198669$\\ ${0.4}$&$0.389418$\\ ${0.6}$&$0.564642$\\
         ${0.8}$&$0.717356$\\ ${1}$&$0.841471$\\ \bottomrule \end {tabular}
```

Chapter 3

Reference summary

3.1 Commands defined in `numerica-tables`

`\nmcTabulate`, `\tabulate`

3.2 Settings for `\nmcTabulate`

Row-variable specification: uniform case §2.1.1

key	type	meaning	comment
`rvar`	token(s)	row variable	
`rstep`	real num	step size	
`rstop`	real num	stop value	either `rstop` or
`rows`	int	number of rows	`rows`, not both
`rspec`	comma list	`{start`, `step`, `rows}`	short form spec.

Row-variable specification: non-uniform case §2.1.2

key	type	meaning	comment
`rdata`	comma list	list of row-var. values	may be stored in a macro
`rfile`	chars	file of row-var. values	file path/name
`rverb`	int (0/1)	display `rdata` or `rfile` values verbatim	default `0`
`rfunc`	token(s)	step function specifying row-var. values	

Row-variable column formatting §2.1.3

key	type	meaning	initial
`rround`	int	rounding	`1`
`ralign`	char (`r/c/l`)	horizontal alignment	`r`
`rfont`	chars	font (`\math<chars>`)	
`rhead`	tokens	header	`rvar`
`rhnudge`	int	nudge header `rhnudge` mu	`0`
`rpos`	int (`0`...`4`)		`1`
`rvar'`	token(s)	2nd row variable col. spec.	`rvar`
`rhead'`	token(s)	header of 2nd r-v col. (if it exists)	`rvar'`
`rhnudge'`	int	nudge 2nd r-v col. header `rhnudge'` mu	`0`
`rfrac`	int (`0`...`5`)	fraction form	`0`

Column-variable specification §2.2.

key	type	meaning	default
`cvar`	token(s)	column variable	
`cstep`	real num	step size	
`cstop`	real num	stop value	either `cstop`
`cols`	int	number of columns	or `cols`, not both
`cspec`	comma list	`{cvar,cstep,cols}`	short form spec.

Column-variable header formatting §2.2.1.

key	type	meaning	default
`chstyle`	int (`0`...`4`)	header style	`0`
`ctitle`	token(s)	single col. alternative header	
`chead`	token(s)	user-defined header	
`calign`	char (r/c/l)	column alignment	`r`
`chnudge`	int	nudge header `chnudge` mu	`0`
`chround`	int	rounding	`0`

Function-value formatting §2.5.

key	type	meaning	default
(pad)	int	t-notation phantom padding	
signs	int	sign handling for function-values	0
diffs	int	insert differences & pre-pad with zeros	0
Q?	tokens	special cell conditional	
A!	token(s)	special cell formatting	

Whole-of-table formatting §2.4.

key	type	meaning	default
ctitle	token(s)	collective title for function-value columns	
	token(s)	inter-header/title row for function-value columns	
header	int (0/1)	suppress/show header row	1
rules	char(s)	horizontal rules template	ThB
foot	token(s)	content of footer line	
rpos	int (0...4)	row variable	1
rbloc	comma list	division of rows into blocks	
valign	char (t/m/b)	vertical alignment of table relative to text baseline	m

Miscellaneous settings

view, equivalent to dbg=11: show the LaTeX expression for the table.

Index

`A!` 41–43
`\abovetopsep` 33
adjoined tables 17
`\arraystretch` 12

`\belowbottomsep` 33
`\bigskip` 33
`booktabs` 4, 28, 29, 33

`calign` 22
`chead` 20–22
`chnudge` 22–23
`chround` 23–24
`chstyle` 20–21
`cmidrow` *see* `csubttl`
`\cmidrulekern` 29
`\cmidrulewidth` 29
`cols` 18–19
column variable 5
  initial value in vv-list 18
  start, step, stop values 18–19
column variable header 19–24
  multi-column case 20–24
    `chstyle` 20–21
  single column case 19
complementary functions 25, 30–31
`cspec` 18–19
`cstep` 18–19
`cstop` 18–19
`csubttl` 27
`ctitle` 19, 25–27
  `*` setting 19
  `**` setting 19, 30
`cvar` 18–19

decimal comma 6, 24
differences 40–41

environments 6
error messages 43

`ff` 24, 25, 31
`foot` 28, 30
footer functions 28
footer row 28, 29
  complementary functions 31
formatting
  column variable header 19–24
  function values 34–44
  grouping rows 31–32
  row variable header 14–15
  row variable values 12–13
  rules 28–30
  title, subtitle, footer 25–28
fraction form
  function values 34–35, 39
  row variable values 16–17
  row/col-dependent rounding 39
function value formatting 34–44
  fraction form 34–35, 39
  highlighting special values 41–44
  padding with zeros 8
  scientific notations 35–38
  sign handling 13, 37–39

header row *see* column variable header, `headless`, row variable header
`headless` 12, 28
`\heavyrulewidth` 29

`\lightrulewidth` 29

`MAX` 42, 43
`MIN` 42
multi-function tables 17, 24–25, 39
  function delimiter 24

n-ary functions 6, 25

\nmcTabulate 5, 6
  nesting 44
  star (*) option 42

padding with phantoms 36–38
padding with zeros 8

Q? 41–43

ralign 13
rbloc 31–32
rblocsep 32
rdata 10–12
rfile 10, 12
rfont 13, 14
rfunc 9–10
rhead' 30–31
rhead 14–18
rhnudge' 30–31
rhnudge 15
round 39–40
rounding value 5, 34
  header row 23–24
  row variable 12–13
  row/col-dependent 39–40
row grouping 31–32
row variable 5
  fraction form 16–17
  initial value in vv-list 7
  start, step, stop values 7–9
  values from list, macro or file 10–12
  variably stepped 10
  verbatim values 11–12, 34–35
row variable column
  alignment 13
  position in table 15, 30
row variable header 14–15
rows 8
rpos 15, 17, 24, 25, 30
rround 12–13
rspec 8
rstep 7
rstop 7
rules 4, 28–30
  edicts 28
rules 28–30
rvar' 30–31
rvar 7
rverb 10–12

saving tables to file 45
scientific notation *see also* t-notation, 35
semicolon and decimal comma 6, 24
\sfrac 12, 16, 34, 35
signs 37–39
\smallskip 33

t-notation 36–38
  accommodating signs 37
table placement
  on the line 33
  on the page 6, 33
table structure 4
tables
  saving to file 45
  viewing LaTeX form 46
\tabulate *see* \nmcTabulate
title row 25–27
  subtitle row 27

valign 33
view 46
vv-list
  and decimal comma 6
  row/col. variable initial values 7, 10, 18, 19