

numerica-tables

Andrew Parsloe
(ajparsloe@gmail.com)

December 11, 2021

Abstract

The `numerica-tables` package defines a command which enables the creation of (possibly multi-column) mathematical tables of function values. *Key=value* settings allow presentation in a wide variety of table styles within the ‘formal table’ framework of the `booktabs` package.

Note:

- This document applies to version 2.0.0 of `numerica-tables`.
- A version of `numerica` from or later than 2021/12/07 is required; (`numerica` requires `amsmath`, `mathtools` and the L^AT_EX3 bundles `l3kernel` and `l3packages`).
- The `booktabs` package is required.
- I refer many times in this document to *Handbook of Mathematical Functions*, edited by Milton Abramowitz and Irene A. Segun, Dover, 1965. This is abbreviated to *HMF*, often followed by a reference to a specific table like Table 1.2.
- Version 2 of `numerica-tables`
 - is the first stand-alone version (in v.1 of `numerica` the `\nmcTabulate` command was available with the `tables` package option);
 - restricts the third item in `rspec` to rows only (`rstop` is no longer accepted there);
 - restricts the third item in `cspec` to cols only (`cstop` is no longer accepted there);
 - allows an additional row, if wanted, between table title and the column header row;
 - removes the `DEL` column function;
 - amends documentation.

Contents

1	Introduction	1
1.1	Table structure	1
1.2	Shared syntax	2
1.2.1	Settings inherited from <code>numera</code>	3
2	\nmcTabulate settings	4
2.1	Row-variable settings	4
2.1.1	Row-variable column formatting	6
2.1.1.1	Rounding: <code>rround</code>	7
2.1.1.2	Alignment: <code>ralign</code>	7
2.1.1.3	Font: <code>rfont</code>	7
2.1.1.4	Row-variable header: <code>rhead</code>	7
2.1.1.5	Nudging the header: <code>rhnu</code>	9
2.1.1.6	Position in the table: <code>rpos</code>	9
2.1.1.7	<code>rvar</code> , <code>rhead</code> , <code>rhnu</code>	10
2.1.2	Adjoined multi-function tables	10
2.2	Column-variable settings	11
2.2.1	Column header formatting	12
2.2.1.1	Header style: single-column case	12
2.2.1.2	Header style: multi-column case	13
2.2.1.3	User-defined header: <code>thead</code>	15
2.2.1.4	Alignment: <code>calign</code>	15
2.2.1.5	Nudging the headers: <code>chnu</code>	16
2.2.1.6	Rounding: <code>chrou</code>	17
2.3	Multiple functions in a single table	17
2.4	Whole-of-table formatting	19
2.4.1	Title for function-value columns: <code>ctitle</code>	19
2.4.2	Inter-header/title row: <code>cmidrow</code>	21
2.4.3	Horizontal rules: <code>rules</code>	22
2.4.4	Footer row: <code>foot</code> setting	24
2.4.4.1	Footer functions	24
2.4.5	Second row-variable column: <code>rpos=4</code>	25
2.4.6	Separating blocks of rows: <code>rbloc</code>	26
2.4.6.1	Adjusting the extra space <code>rblocsep</code>	27

2.5	Formatting function values	28
2.5.1	Trailing optional argument	28
2.5.1.1	The <code>t</code> option	29
2.5.2	Padding the exponent: <code>(pad)</code>	29
2.5.3	Accommodating signs: <code>signs</code>	30
2.5.4	Differences: <code>diffs</code>	32
2.5.5	Formatting special values: <code>Q?</code> and <code>A!</code>	33
2.6	Star option: <code>\nmcTabulate*</code>	34
2.6.1	Errors	35
2.6.2	Scientific notation	35
2.6.3	Nesting	36
2.6.3.1	Nesting of <code>\nmcTabulate</code>	36
2.6.3.2	Nesting within <code>\nmcTabulate</code>	37
2.7	Table placement	38
2.7.1	Vertical alignment	38
2.8	The <code>reuse</code> setting	39
3	Reference summary	42
3.1	Commands defined in <code>numerica-tables</code>	42
3.2	Settings for <code>\nmcTabulate</code>	42

Chapter 1

Introduction

Entering

```
\usepackage{numerica-tables}
```

in the preamble of a document gives access to a command `\nmcTabulate` for creating tables of function values in a wide variety of styles. (Since `numerica-tables` requires – and loads – the `numerica` package, all commands of that package are also available.) 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. Segun, 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 aims to reproduce most of those different solutions, within `booktabs` elegance.

To create a table we need to specify a function to tabulate. The values this 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 §2.6;
2. `[settings]` comma-separated list of *key=value* settings, at the heart of creating a table of function values; see Chapter 2;
3. `{expr.}` mandatory argument specifying the mathematical expression or expressions in L^AT_EX form to be tabulated;
4. `[vv-list]` comma-separated list 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); see §2.5.1.

Unlike `\nmcEvaluate` (from `numerica`), for `\nmcTabulate`

- math delimiters are irrelevant – it makes no difference to the display of the result whether the command wraps around math delimiters, is wrapped within math delimiters, or if there are no math delimiters involved whatever;
- 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 code from `numerica` for `\nmcEvaluate`. Each argument contains items *necessary* for the construction of any table of function values.

1.2.1 Settings inherited from `numerica`

Most of the settings available to the command `\nmcEvaluate` from `numerica` are also available to `\nmcTabulate`. To save switching between documents I reproduce the table of relevant options found in `numerica.pdf` (only the punctuation `p` setting is missing), although for discussion of the options you will need to refer to that document. (Note that the setting `vvmode` of v.1 of `numerica` is still available; it is equivalent to the `vv@` setting.) The `dbg` (and `view`) keys have been disabled for `\nmcTabulate` at present (but might be enabled in the future).

Table 1.1: Settings options inherited from `\nmcEvaluate`

key	type	meaning	default
<code>dbg</code>	int	debug ‘magic’ integer	0
<code>view</code>		equivalent to <code>dbg=1</code>	
<code>^</code>	char	exponent mark for sci. notation input	e
<code>xx</code>	int (0/1)	multi-token variable switch	1
<code>()</code>	int (0/1/2)	trig. function arg. parsing	0
<code>o</code>		degree switch for trig. functions	
<code>log</code>	num	base of logarithms for <code>\log</code>	10
<code>vv@</code>	int (0/1)	vv-list calculation mode	0
<code>vvd</code>	tokens	vv-list display-style spec.	{,}\mskip 12mu plus 6mu minus 9mu(vv)
<code>vvi</code>	tokens	vv-list text-style spec.	{,}\mskip 36mu minus 24mu(vv)
<code>*</code>		suppress equation numbering if <code>\</code> in <code>vvd</code>	
<code>S+</code>	int	extra rounding for stopping criterion for sums	2
<code>S?</code>	int ≥ 0	stopping criterion query terms for sums	0
<code>P+</code>	int	extra rounding for stopping criterion for products	2
<code>P?</code>	int ≥ 0	stopping criterion query terms for products	0

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

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 (which is specified in the *vv*-list), what value to tabulate to, `rstop`, and how fine-grained the tabulation is to be, the step size `rstep`.

The two tables in the first 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

Table 2.1: Row-variable specification

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

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 L^AT_EX expression. Both `rstep` and `rstop` can also be L^AT_EX expressions. However, 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.

The difference in appearance of the two tables below 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 – and makes an obvious improvement to the table’s appearance.

```

\begin{table}
\code{\tabulate[rvar=x,rstep=0.2,rstop=1]
{ \sin x }[x=0]\qqquad}
\code{\tabulate[rvar=x,rstep=0.2,rstop=1]
{ \cos x }[x=0] [*]}
\end{table}

```

x	$\sin x$	x	$\cos x$
0	0	0.0	1.000000
0.2	0.198669	0.2	0.980067
0.4	0.389418	0.4	0.921061
0.6	0.564642	0.6	0.825336
0.8	0.717356	0.8	0.696707
1	0.841471	1.0	0.540302

Sometimes (perhaps often) it may prove more convenient to specify the number of rows, `rows`, explicitly 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 of use.

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 of use. In the third table a L^AT_EX expression has been inserted for `rows` in the `rspec` comma list. Like `rstep` and `rstop`, `rows` can be a L^AT_EX expression, but it is evaluated *before* the vv-list and therefore, unlike `rstep` and `rstop`, cannot depend on quantities specified there like the row variable.

```

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

```

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

In version 1 of `numerica` (when `\nmcTabulate` became available with the package option `tables`), the third item in `rspec` was not restricted to `rows` but could also be `rstop`; parentheses placed around the third item assigned it to `rows` rather than `rstop`. On reflection, this now seems obscure and open to error. In v.2 of `numerica-tables` (in fact its first version as a separate package) the third item of `rspec` is restricted to `rows` only, which no longer needs to be parenthesized.

2.1.1 Row-variable column formatting

The padding option (*) of the trailing optional argument is one way of formatting the row-variable column, 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 2.2.

Table 2.2: Formatting the row-variable column

key	type	meaning	default
<code>rround</code>	int	rounding	1
<code>ralign</code>	char (r/c/1)	horizontal alignment	r
<code>rfont</code>	chars	font (<code>\math<chars></code>)	
<code>rhead</code>	tokens	header	<code>rvar</code>
<code>rhnudge</code>	int	nudge header <code><int></code> mu	0
<code>rpos</code>	int (0...4)	column position(s)	1
<code>rvar'</code>	tokens	2nd row-variable col. spec.	<code>rvar</code>
<code>rhead'</code>	tokens	header of 2nd rv col. (if it exists)	<code>rvar'</code>
<code>rhnudge'</code>	int	nudge 2nd rv col. header <code><int></code> mu	0

2.1.1.1 Rounding: `rround`

After studying 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 (the first table below). `numerica-tables` uses a default rounding value of 1 for the row variable and has rounded 0.25 down to 0.2 and 0.75 up to 0.8 accordingly. The second table corrects matters by adjusting the row-variable rounding (`rround`) to 2.

```
\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] [*]
```

	<hr/>		<hr/>	
	<i>x</i>	<i>sin x</i>	<i>x</i>	<i>sin x</i>
	<hr/>	<hr/>	<hr/>	<hr/>
	0.0	0.000000	0.00	0.000000
⇒	0.2	0.247404 (Eh???)	0.25	0.247404
	0.5	0.479426	0.50	0.479426
	0.8	0.681639	0.75	0.681639
	1.0	0.841471	1.00	0.841471
	<hr/>	<hr/>	<hr/>	<hr/>

2.1.1.2 Alignment: `ralign`

By default, the alignment of all columns is to the right, as in the 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 – minus signs can 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.

2.1.1.3 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.

2.1.1.4 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`.

```

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

```

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

```

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

```

x	$\sin x$	x	$\sin x$
-0.50	-0.479426	-0.50	-0.479426
-0.25	-0.247404	-0.25	-0.247404
0.00	0.000000	0.00	0.000000
0.25	0.247404	0.25	0.247404
0.50	0.479426	0.50	0.479426

(To my eye, aligning the 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.)

2.1.1.5 Nudging the header: `rhnu`

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 `rhnu`.

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 `rhnu=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 `numera-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 `rhnu` a *negative* value (`rhnu=-12` in the example) it is brought back to a centred position in the row-variable column.

```

\begin{table}
\table
[rvar=x,rstep=0.25,rstop=0.5,rround=2,
 rfont=bf,rhead=\boldsymbol{x},rhnu=9]
{ \sin x }[x=-0.5][4*]\qqquad
\table
[rvar=x_{\text{int}},rstep=1,rstop=4,
 rround=0,rfont=bf,rhnu=-12,
 rhead=\boldsymbol{x_{\text{int}}}]
{ \sin x_{\text{int}} }[x_{\text{int}}=0][4*]
\end{table}

```

x	$\sin x$	x_{int}	$\sin x_{\text{int}}$
-0.50	-0.4794	0	0.0000
-0.25	-0.2474	1	0.8415
0.00	0.0000	2	0.9093
0.25	0.2474	3	0.1411
0.50	0.4794	4	-0.7568

2.1.1.6 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 §2.4.5;
- Any other integer acts like `rpos=1`.

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

2.1.1.7 `rvar'`, `rhead'`, `rhnudge'`

These settings become relevant only when `rpos=4`; see §2.4.5.

2.1.2 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] [*]

```

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.

Table 2.3: Column-variable specification

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

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, is to create and adjoin separate tables, one per parameter value, but that is clumsy. A more systematic 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*.

```

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

```

$x \backslash k$	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`. An example where this is useful is presented in §2.4.4.1. (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}`.¹ Thus, for the preceding table I could 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

```
\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.

`cstep`, `cstop` and `cols` can all be L^AT_EX expressions. The first two are evaluated *after* the vv-list; `cols` is evaluated *before* the vv-list. Hence `cstep` and `cstop` may depend on the row and column variables, which are given their initial values in the vv-list.

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 `thead`. First I discuss the built-in styles.

2.2.1.1 Header style: single-column case

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 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

¹This is a change from v.1 of `numerica`; see the boxed comment at the end of §2.1.

Table 2.4: Formatting the column-variable header

key	type	meaning	default
<code>chstyle</code>	int (0...4)	header style	0
<code>ctitle</code>	tokens	single col. alternative header	
<code>thead</code>	tokens	user-defined header	
<code>calign</code>	char (r/c/l)	column alignment	r
<code>chnudge</code>	int	nudge header <code>int</code> μ	0
<code>chround</code>	int	column header rounding	0

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 `thead` a value. This key I discuss below in §2.2.1.3. `thead` is entirely up to the user to specify, including any math environment and positioning.

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

2.2.1.2 Header style: multi-column case

`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 $k = 3$. 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. Note that the row-variable setting `rhead` no longer needs the `\backslash k` part since the column variable is now explicitly indicated. (The first table in the example below.)

`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 (the second table in the example).

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

```

\begin{table}
\tableto{
  [rspec={x,0.2,6},
   cspec={k,2,3},chstyle=2]
  { \sin kx }[k=3,x=0] [3*]
}
\end{table}

```

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

```

\begin{table}
\tableto{
  [rspec={x,0.2,6},
   cspec={k,2,3},chstyle=3]
  { \sin kx }[k=1,x=0] [4*]
}
\end{table}

```

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 result 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:

```

\begin{table}
\tableto{
  [rspec={x,0.2,6},
   cspec={k,2,3},chstyle=4]
  { \sin kx }[k=1,x=0] [4*]
}
\end{table}

```

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: `thead`

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 `thead` key. Users can assign whatever value they like to `thead`. 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 `thead` is potentially far more complicated. Thus for $k + \sin kx$,

```

\begin{table}
\thead
  [rspec={x,0.2,6},
   cspec={k,2,3},
   thead=${1+\sin x}&&{3+\sin 3x}&&{5+\sin 5x}]
{ k+\sin kx }[k=1,x=0][4*]
\end{table}

```

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 `thead` 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

```

\begin{table}
\thead
  [rspec={x,0.2,6},ralign=c,
   cspec={k,2,3},chstyle=2,calign=c]
{ \sin kx }[k=3,x=0][*]
\end{table}

```

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 §2.5.3.

2.2.1.5 Nudging the headers: `chnudge`

In left or right alignment it is possible to nudge the 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.

```

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

```

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`.

```

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

```

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`.

```

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

```

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 §2.1.2, tabulating more than one function at a time is too common a need to have to resort to a fudge like adjoining tables. There is a systematic way of handling this task available in `numerica-tables`. In v.1 of `numerica`, it sufficed to enter the functions in the main argument separated by commas, and to *precede the first function with a comma*, which was the signal `numerica` needed to make the internal adjustments for a multi-function table. In v.2 of `numerica-tables` this option is still available, but rather than use a ‘trick’ like preceding the first function with a comma, the preferred option now is to use a new (with v.2) setting, `multifn`.

The first table below uses the old ‘trick’ of starting the main argument with a comma; the second table uses the `multifn` setting (and note in both the `o` setting indicating that the arguments of `sin` and `cos` are in degrees):

```

\begin{table}
\table[o,rround=0,
        rvar=\theta,rstep=10,rstop=90]
{ ,\sin \theta,\cos \theta }[\theta=0][*]
\quad
\begin{table}
\table[o,multifn,rpos=2,rround=0,
        rvar=\theta,rstep=10,rstop=90]
{ \sin \theta,\cos \theta }[\theta=0][*]
\end{table}
\end{table}

```

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

These 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 (§2.4.5) which enables complementary functions to be tabulated in ‘half tables’ (for examples see *HMF* Tables 4.10–4.12 for the trigonometric functions).

A comma may not always be a convenient separator – it may occur in one of the functions being tabulated (perhaps in `\max` or `\min`). By assigning a value to the setting `multifn`,

```
multifn=<char>
```

the assigned character can be used to separate the functions. In the following example, a semicolon is used. Further, the row-variable column is duplicated on the right by using the `rpos=3` setting, and the table gives another illustration of the use of `thead`:

```
\tabulate[o,multifn=;,rpos=3,rround=0,
  rvar=\theta,rstep=10,rstop=90,
  chead=max\hphantom{00} & min\hphantom{00}]
{ \max(\sin \theta,\cos \theta);
  \min(\sin \theta,\cos \theta) }
[\theta=0][*]
```

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
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 glaring omission in the table is any explicit statement of what the functions are that are being tabulated. Maximum and minimum, yes, but of what? That is (potentially) remedied with the `ctitle` setting discussed next; see §2.4.1.

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.

Table 2.5: Table formatting

key	type	meaning	default
<code>ctitle</code>	token(s)	collective title for function-value columns	
<code>cmidrow</code>	token(s)	inter-header/title row for function-value columns	
<code>rules</code>	char(s)	horizontal rules template	ThB
<code>foot</code>	token(s)	content of footer line	
<code>rpos</code>	int (0...4)	row-variable column position(s)	1
<code>rbloc</code>	comma list	division of rows into blocks	
<code>rblocsep</code>	length	extra spacing between blocks of rows	1 ex

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 use the setting to clarify the last example:

```
\tabulate[o,multifn=;,rpos=3,rround=0,
  rvar=\theta,rstep=10,rstop=90,
  ctitle=\sin\theta{,}\,\cos\theta,
  chead=max\hphantom{00} & min\hphantom{00}]
{ \max(\sin \theta,\cos \theta);
  \min(\sin \theta,\cos \theta) }
[\theta=0] [*]
```

$\sin \theta, \cos \theta$			
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
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

Now it is clearer what is being tabulated, although the reader is still being asked to interpret rather than read what the table is showing.

There are two in-built settings for `ctitle`: `ctitle=*`, which makes the formula the title, and `ctitle=**`, which makes a title of the formula and v-list. Surely, `ctitle=*` is what we want:

```
\tabulate[o,multifn=;,rpos=3,rround=0,
  rvar=\theta,rstep=10,rstop=90,
  ctitle=*,
  chead=max\hphantom{00} & min\hphantom{00}]
{ \max(\sin \theta,\cos \theta);
  \min(\sin \theta,\cos \theta) }
[\theta=0][*]
```

$\max(\sin \theta, \cos \theta); \min(\sin \theta, \cos \theta)$			
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
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

Well, that is clear but the length of the title distorts the table. This is where the `cmidrow` key can help (see below) but we can also use an `aligned` environment within `ctitle`:

```
\tabulate[o,multifn=;,rpos=3,rround=0,
  rvar=\theta,rstep=10,rstop=90,
```



```

ctitle={\begin{aligned}
        \max(\sin\theta,\cos\theta)\ll[-0.7ex]
        \min(\sin\theta,\cos\theta)
    \end{aligned}},
thead=\max\hphantom{00} & \min\hphantom{00}]
{ \max(\sin \theta,\cos \theta);
  \min(\sin \theta,\cos \theta) }
[\theta=0] [*]

```

	max(sin θ , cos θ)		
	min(sin θ , cos θ)		
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
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 table is no longer distorted in width. Note the `\ll[-0.7ex]` within the `aligned` environment. This shrinks the vertical space between the two lines of the title. Without it, the lines are too far apart.

2.4.2 Inter-header/title row: `cmidrow`

Some tables need to fit more header material or title material into their rows than can be comfortably accommodated there. 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 row and title row.

I have chosen `cmidrow` for the key name, in the sense of a row ‘mid header and title rows’. The initial ‘c’ emphasizes that it is constrained to the span of the column-variable (or function-value) columns only (like `thead` and `ctitle`). The entire content is the responsibility of the user,

```
cmidrow=<tokens>
```

including insertion of the necessary number of tab characters, `&`, and any math delimiters required.

```
\tabulate[o,multifn=;,rpos=3,rround=0,
```

```

rvar=\theta,rstep=10,rstop=90,
ctitle=\max(\sin\theta{,}\cos\theta),
cmidrow=\multicolumn{2}{c}
        {\$\min(\sin\theta,\cos\theta)$},
thead=\max\hphantom{00} & \min\hphantom{00}]
{ \max(\sin \theta,\cos \theta);
  \min(\sin \theta,\cos \theta) }
[\theta=0] [*]

```

	max(sin θ , cos θ)		
	min(sin θ , cos θ)		
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
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 display looks the same as in the previous example but was obtained perhaps more straightforwardly.

2.4.3 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).² I have used horizontal rules in the various examples in the present document because these too are tables within text. Some form of delineation seems necessary.

The `rules` key enables precisely which rules are used to be specified. The value of the key is a ‘word’ – a sequence of letters – where the characters have the significance and default thicknesses (from `booktabs`) shown in Table 2.6. The default setting is `rules=ThB`. To insert a rule beneath the title, for example, change this to `rules=TthB`. If in addition you are using a footer row and want a rule above it, then the specification is `rules=TthfB` and if you are using a row between header and title rows (a ‘midrow’) and want a rule beneath that too,

²The tables in *HMF* are often inelegantly typeset, and sometimes ugly. For all that, I have used it as a valuable source for the variety of structures that the editors found necessary, or at least useful, for presenting a multitude of different kinds of numerical data.

Table 2.6: 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	<code>\heavyrulewidth=.08em</code>
t	title	below title	f-v cols	<code>\cmidrulewidth =.03em</code>
m	midrow	below midrow	f-v cols (if 1 r-v col.) < table (if 2 r-v cols)	<code>\cmidrulewidth =.03em</code>
h	header	below header	table	<code>\lightrulewidth=.05em</code>
f	footer	above footer	table	<code>\cmidrulewidth =.03em</code>
B	bottom	below table	table	<code>\heavyrulewidth=.08em</code>

then the spec. is `rules=TtmhfB`. To my eye that is too many rules; at most only one of title and midrow rules should be used.

The midrow 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 one row-variable column then, like the title rule, the midrow rule spans only the function-value columns. If there are two row-variable columns then the midrow 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 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 2.6 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`).

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] [*]
```

$a \sin kx, (a = 2/\pi)$				
x	$k = 3.00$	$k = 3.25$	$k = 3.50$	
\Rightarrow 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	

2.4.4 Footer row: foot setting

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 be 1 less than the total number of columns (including row-variable columns) in the table – or some adjustment thereof if you use `\multicolumn`. You can put into the footer what you wish:

```
foot=<tokens>
```

(*HMF* uses the footer mainly for cryptic descriptions of the accuracy and needs of interpolation methods.)

The one exception is when `foot=*`. This will fill the footer with the header, but *reversed*. 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 §2.4.5.

2.4.4.1 Footer functions

It is also possible to use the footer for displaying the values of certain column functions. `numerica-tables` provides four of these. They can be used in the footer (and only in the footer): `SUM`, `AVE` (average), `MAX` and `MIN`. These functions act on the function values of the column they are in. They *do not* combine mathematically: entering `MAX-MIN` in the footer of a given column will produce a footer entry containing two values (those of `MAX` and `MIN`) separated by a minus sign. The numerical output from each function is automatically wrapped in math delimiters (`$`) so that minus signs display correctly.

In v.1 of `numerica`, tables supported *five* footer functions. The fifth was `DEL=MAX-MIN`. It is no longer supported. First, the name was unclear and in any case, it is easy to calculate from `MAX` and `MIN`.

In the following example, I have chosen a column variable step size of zero. This is possible because in the column spec., I have also specified the exact number of columns. Zeroing the step size means the same set of figures can be used for the four footer functions to act on.

```

\begin{table}
\caption{a sin kx, (a = 2/π, k = 3.5)}
\begin{tbl_struct}
\TABSTRUCTHEAD
\TABSTRUCTBODY
\TABSTRUCTFOOT
\end{tbl_struct}
\end{table}

```

x	SUM	AVE	MAX	MIN
0.00	0.000000	0.000000	0.000000	0.000000
0.25	0.488633	0.488633	0.488633	0.488633
0.50	0.626425	0.626425	0.626425	0.626425
0.75	0.314439	0.314439	0.314439	0.314439
1.00	-0.223316	-0.223316	-0.223316	-0.223316
	1.206181	0.241236	0.626425	-0.223316

2.4.5 Second row-variable column: rpos=4

In §2.1.1.6 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. The example also gives an illustration of the use of an expression in the third element of `rspec`.

```

\begin{table}
\caption{sin and cos of theta}
\begin{tbl_struct}
\TABSTRUCTHEAD
\TABSTRUCTBODY
\TABSTRUCTFOOT
\end{tbl_struct}
\end{table}

```

θ	$\sin \theta$	$\cos \theta$	
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
	$\cos \theta$	$\sin \theta$	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 (`\theta`);
- the blank header for the `rvar'` column (since no value was set for `rhead'`);
- 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.

2.4.6 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 columns into blocks of 5 rows by means of the setting `rbloc=5`.

```
\tabulate[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
⇒ 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

2.4.6.1 Adjusting the extra space `rblocsep`

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

Table 2.7: Formatting function values

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

2.5 Formatting function values

In the examples used so far, function values have been limited to a narrow range, generally $[-1, 1]$. What happens when function values span orders of magnitude?

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*), and scientific notation is set or not. 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 ‘ $\times 10$ ’ 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 – a few function values at most.

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


⇒	<table style="border-collapse: collapse; width: 100%;"> <thead> <tr> <th style="border: none; padding: 2px 10px;">x</th> <th style="border: none; padding: 2px 10px;">e^x</th> </tr> </thead> <tbody> <tr><td style="border: none; padding: 2px 10px;">-3</td><td style="border: none; padding: 2px 10px;">4.978707×10^{-2}</td></tr> <tr><td style="border: none; padding: 2px 10px;">-2</td><td style="border: none; padding: 2px 10px;">1.353353×10^{-1}</td></tr> <tr><td style="border: none; padding: 2px 10px;">-1</td><td style="border: none; padding: 2px 10px;">3.678794×10^{-1}</td></tr> <tr><td style="border: none; padding: 2px 10px;">0</td><td style="border: none; padding: 2px 10px;">1.000000</td></tr> <tr><td style="border: none; padding: 2px 10px;">1</td><td style="border: none; padding: 2px 10px;">2.718282</td></tr> <tr><td style="border: none; padding: 2px 10px;">2</td><td style="border: none; padding: 2px 10px;">7.389056</td></tr> <tr><td style="border: none; padding: 2px 10px;">3</td><td style="border: none; padding: 2px 10px;">2.008554×10^1</td></tr> </tbody> </table>	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	<table style="border-collapse: collapse; width: 100%;"> <thead> <tr> <th style="border: none; padding: 2px 10px;">x</th> <th style="border: none; padding: 2px 10px;">e^x</th> </tr> </thead> <tbody> <tr><td style="border: none; padding: 2px 10px;">-3</td><td style="border: none; padding: 2px 10px;">4.978707×10^{-2}</td></tr> <tr><td style="border: none; padding: 2px 10px;">-2</td><td style="border: none; padding: 2px 10px;">1.353353×10^{-1}</td></tr> <tr><td style="border: none; padding: 2px 10px;">-1</td><td style="border: none; padding: 2px 10px;">3.678794×10^{-1}</td></tr> <tr><td style="border: none; padding: 2px 10px;">0</td><td style="border: none; padding: 2px 10px;">1.000000×10^0</td></tr> <tr><td style="border: none; padding: 2px 10px;">1</td><td style="border: none; padding: 2px 10px;">2.718282×10^0</td></tr> <tr><td style="border: none; padding: 2px 10px;">2</td><td style="border: none; padding: 2px 10px;">7.389056×10^0</td></tr> <tr><td style="border: none; padding: 2px 10px;">3</td><td style="border: none; padding: 2px 10px;">2.008554×10^1</td></tr> </tbody> </table>	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
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																																	

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

```

⇒	<table style="border-collapse: collapse; width: 100%;"> <thead> <tr> <th style="border: none; padding: 2px 10px;">x</th> <th style="border: none; padding: 2px 10px;">e^x</th> </tr> </thead> <tbody> <tr><td style="border: none; padding: 2px 10px;">-3</td><td style="border: none; padding: 2px 10px;">(-2) 4.978707</td></tr> <tr><td style="border: none; padding: 2px 10px;">-2</td><td style="border: none; padding: 2px 10px;">(-1) 1.353353</td></tr> <tr><td style="border: none; padding: 2px 10px;">-1</td><td style="border: none; padding: 2px 10px;">(-1) 3.678794</td></tr> <tr><td style="border: none; padding: 2px 10px;">0</td><td style="border: none; padding: 2px 10px;">1.000000</td></tr> <tr><td style="border: none; padding: 2px 10px;">1</td><td style="border: none; padding: 2px 10px;">2.718282</td></tr> <tr><td style="border: none; padding: 2px 10px;">2</td><td style="border: none; padding: 2px 10px;">7.389056</td></tr> <tr><td style="border: none; padding: 2px 10px;">3</td><td style="border: none; padding: 2px 10px;">(1) 2.008554</td></tr> </tbody> </table>	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	<table style="border-collapse: collapse; width: 100%;"> <thead> <tr> <th style="border: none; padding: 2px 10px;">x</th> <th style="border: none; padding: 2px 10px;">e^x</th> </tr> </thead> <tbody> <tr><td style="border: none; padding: 2px 10px;">-3</td><td style="border: none; padding: 2px 10px;">(-2) 4.978707</td></tr> <tr><td style="border: none; padding: 2px 10px;">-2</td><td style="border: none; padding: 2px 10px;">(-1) 1.353353</td></tr> <tr><td style="border: none; padding: 2px 10px;">-1</td><td style="border: none; padding: 2px 10px;">(-1) 3.678794</td></tr> <tr><td style="border: none; padding: 2px 10px;">0</td><td style="border: none; padding: 2px 10px;">(0) 1.000000</td></tr> <tr><td style="border: none; padding: 2px 10px;">1</td><td style="border: none; padding: 2px 10px;">(0) 2.718282</td></tr> <tr><td style="border: none; padding: 2px 10px;">2</td><td style="border: none; padding: 2px 10px;">(0) 7.389056</td></tr> <tr><td style="border: none; padding: 2px 10px;">3</td><td style="border: none; padding: 2px 10px;">(1) 2.008554</td></tr> </tbody> </table>	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
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 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] \qqquad

```

```

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

```

x	e^x	x	e^x
-3	(-2) 4.978707	-3	(-2) 4.978707
-2	(-1) 1.353353	-2	(-1) 1.353353
-1	(-1) 3.678794	-1	(-1) 3.678794
0	1.000000	0	(0) 1.000000
1	2.718282	1	(0) 2.718282
2	7.389056	2	(0) 7.389056
3	(1) 2.008554	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 `\table` 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.

2.5.3 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, the *significand* the following decimal figures. `numerica-tables` offers the `signs` key to align (or not) the exponents. The setting is

```
signs = <integer>
```

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

- `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 with `signs=-2`, `signs=-1` and `signs=2`, all give acceptable results.

```

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

```

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

```

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

```

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

But 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):

```

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

```

	x	$10 \sin 5x$		x	$10 \sin 5x$		x	$10 \sin 5x$
	-0.4	-9.0930		-0.4	-9.0930		-0.4	-9.0930
	-0.3	-9.9749		-0.3	-9.9749		-0.3	-9.9749
	-0.2	-8.4147		-0.2	-8.4147		-0.2	-8.4147
\Rightarrow	-0.1	-4.7943		-0.1	-4.7943		-0.1	-4.7943
	0.0	+0.0000		0.0	+0.0000		0.0	0.0000
	0.1	+4.7943		0.1	4.7943		0.1	4.7943
	0.2	+8.4147		0.2	8.4147		0.2	8.4147
	0.3	+9.9749		0.3	9.9749		0.3	9.9749
	0.4	+9.0930		0.4	9.0930		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 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,
rhnuage=9,chnudge=21,diffs=3]
{ \sinh x }[x=1][*4]
```

	x	$\sinh x$
	1.00	1.1752
	1.01	1.1907 ¹⁵⁵
\Rightarrow	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]\quad
\tabulate[rspec={x,0.01,1.05},rround=2,
  rhnudge=9,chnudge=30,diffs=4]
  { -\sinh x }[x=1][*4]
```

x	$\sinh x$	x	$-\sinh x$
1.00	1.1752	1.00	-1.1752
1.01	1.1907 ¹⁵⁵	1.01	-1.1907 ⁰¹⁵⁵
⇒ 1.02	1.2063 ¹⁵⁶	1.02	-1.2063 ⁰¹⁵⁶
1.03	1.2220 ¹⁵⁷	1.03	-1.2220 ⁰¹⁵⁷
1.04	1.2379 ¹⁵⁹	1.04	-1.2379 ⁰¹⁵⁹
1.05	1.2539 ¹⁶⁰	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.

2.5.5 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’ or 0 for ‘false’). *This is not a L^AT_EX expression*; this is an `l3fp` expression.³ `numERICA-tables` uses `@` to denote the current function value, so queries like `Q?=@<0` (Is the current function value negative?) or `Q?=@>=pi` (Is the current function value greater than or equal to π ?) are valid questions. (Note the braces in the second question, used to hide the equality sign.) Other possible useful components of such questions are `exp(1)` for the number e , `||` for logical Or, `&&` for logical And, and `!` for logical Not, as well as the familiar arithmetic symbols, `+`, `-`, `*`, `/` and `^`, relation symbols `<`, `>`, `=` and

³Documentation about `l3fp` can be found in `interface3.pdf`, which is part of the `l3kernel` bundle.

their combinations like !=, >=, <= etc., and parentheses. In addition to these components, `numerica-tables` offers `MAX` and `MIN` which are the maximum and minimum function values tabulated, so that, e.g., `Q?={@=MIN}` (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 L^AT_EX 2_ε 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)},rround=0,rpos=2,rules=Tth,
   cspec={m,1,1+(5-2)},ctitle=*,chstyle=2,
   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
⇒	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

2.6 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*`, `\macro*`, `\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.

```

\table*
  [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]

```

⇒ 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 either less than 0 or greater than 0.5.

2.6.1 Errors

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

```

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

```

⇒ !!! No table value satisfies query Q? in: settings. !!!

And if there is no query at all when the star option is chosen, another message is shown:

```

\table*
  [rspec={n,1,1+(15-4)},cspec={m,1,1+(5-2)}]
  { \cos(m\pi/n) }[n=4,m=2] [*4]

```

⇒ !!! \nmctabulate* lacks a query Q? in: settings. !!!

2.6.2 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 it is the x option that you enter, then you will need to place the \table* command between math delimiters, otherwise the \times symbol resulting from the x option will generate a L^AT_EX error ('Missing \$ inserted'):

```

$
\table*
  [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]
$

```

⇒ 6.2349 × 10⁻¹.

2.6.3 Nesting

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

2.6.3.1 Nesting of `\nmcTabulate`

Occasionally one might want to extract a value – perhaps a maximum or minimum – 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 v.2 of `numerica` we can omit (or forget to include) the star. All we require is that the `Q?` setting is satisfied by at least one function value.

```
\eval{$
  (\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]
⇒ (0.9397) sinh t + (−1.0000) cosh t = −0.3541, (t = 2).
Evaluating the tables
```

```
\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

```
\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 an error message is shown; e.g.,

```
\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]
```


⇒ !!! No table value satisfies query Q? in: settings (1). !!!

Here, ‘settings (1)’ tells us that the message refers to the (or a) command at the first level of nesting.

2.6.3.2 Nesting within `\nmcTabulate`

Perhaps a more likely situation is to want to nest other commands within a `\tabulate` command, in particular those from the associated package `numerica-plus`. In the documentation for that package, as an illustration of the use of those commands, I describe how they can be used to numerically investigate the timing of signals between points fixed on a rotating disk. and in particular how to look for three-point circuits with the property that, despite the rotation, signals take the same time traversing the circuit in one direction as the other.

I show a complicated expression below involving a `\tabulate` command wrapped around a `\solve` command (from `numerica-plus`) wrapped around four `\iter*` commands (also from `numerica-plus`). I also show the table resulting from it all. On my High St laptop the table takes perhaps two minutes to compile – I haven’t timed it exactly but it is well over a minute. Two minutes is the blink of an eye in a lifetime but an age sitting staring at a computer screen. Hence I compiled the table separately, saved it to file using the `\reuse` command (see the discussion below, §2.8) and pasted the saved result into this document.

To explain the expression: suppose the three points on the disk have polar coordinates $(r, \pm\theta)$ and $(a, 0)$ in a co-rotating coordinate system. In the underlying inertial system, the signal speed is c and the angular velocity ω . The cosine rule for solving triangles gives for the travel times t between the points

$$t = c^{-1} \sqrt{r^2 + a^2 - 2ra \cos(\theta \pm \omega t)}$$

and

$$t = c^{-1} r \sqrt{2 - 2 \cos(2\theta \pm \omega t)}$$

where the plus sign is for a signal in the direction of rotation and the minus sign for a signal against the rotation. Thus the travel times are fixed points of the expressions on the right. Fixed points are found with the `\iter*` command, which is where those terms of the expression come from. The `\solve` command is then used to find, for a given value of θ , a value of r for which the difference of the travel times is zero for the circuit traversed in the two directions. Finally, the `\tabulate` command creates a table of r values for different values of the row-variable θ and column variable a .

As noted earlier, it takes a long time to compile this expression on an ‘ordinary’ laptop, and so I have compiled the expression separately and pasted the result into this document:

```
\tabulate[rspec={\theta,0.2,5},chstyle=2,
          cspec={a,5,3},ctitle=r_{\Delta t=0}]
```

```

{ \solve[var=r,vvi=,+=1]
  {% circuit 1231
    2\times\iter*[var=t,+=1]{
      c^{-1}\sqrt{a^2+r^2-
        2ar\cos(\theta-\omega t)} }[4]
    + \iter*[var=t,+=1]{
      c^{-1}r\sqrt{2-2\cos(2\theta+\omega t)} }[4]
  % circuit 1321
    - 2\times\iter*[var=t,+=1]{
      c^{-1}\sqrt{a^2+r^2-
        2ar\cos(\theta+\omega t)} }[4]
    - \iter*[var=t,+=1]{
      c^{-1}r\sqrt{2-2\cos(2\theta-\omega t)} }[4]
  }[3]
}[ c=30,r=a+10,a=10,
  \theta=0.2,\omega=0.2,t=1 ] [3*]

```

	θ	$r_{\Delta t=0}$		
		$a = 10$	$a = 15$	$a = 20$
\Rightarrow	0.2	10.21	15.31	20.43
	0.4	10.87	16.33	21.82
	0.6	12.15	18.31	24.55
	0.8	14.46	21.88	29.59
	1.0	18.82	28.86	39.82

2.7 Table placement

There is only one setting in this category that is part of `numERICA-tables` as such, the tabular vertical alignment option; see §2.7.1. But L^AT_EX allows one to insert vertical space with its `\bigskip`, `\medskip`, `\smallskip`, usually about one line space, a half line space, and a quarter line space (with `stretch` and `shrink`), and `booktabs` provides `\abovetopsep` and `\belowbottomsep`, both set by default to `0ex` (or any other unit you care to use) 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).

2.7.1 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][*] \qqquad
B \tabulate[rstep={x,0.2,1+(1/0.2)}]
  { \tan x }[x=0][*] \qqquad
C \tabulate[valign=b,rstep={x,0.2,(6)}]
  { \sqrt{\sec^2 x - 1} }[x=0][*]
```

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

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

As explained in §2.1.2, 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).

2.8 The reuse setting

By entering

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

it is possible to specify what is saved when the `\tabulate` command is followed by a `\reuse` command.

- `reuse=0` saves the table as displayed (the default);
- `reuse=n` saves the n -th *column* of function values

- if `rpos` is non-zero: in a comma-separated list of braced pairs `{row-variable value,function value}`;
- if `rpos=0`: in a comma-separated list of function values.
- `reuse=-n` saves the *n*-th row of function values in a comma-separated list that includes the row-variable value in its appropriate position(s) if `rpos` is non-zero.

In the following example the third row is saved to the control sequence `\rowiii` by using the setting `reuse=-3`.

```

\begin{table}
\renewcommand{\tablewidth}{\textwidth}
\renewcommand{\tableborder}{\hrule}
\renewcommand{\tablecaption}{ $a \sin kx, (a = 2/\pi)$ }
\renewcommand{\tableborder}{\hrule}
\begin{array}{rcccc}
\text{\scriptsize } x & \text{\scriptsize } k = 3.00 & \text{\scriptsize } k = 3.25 & \text{\scriptsize } k = 3.50 & \\
\hline
0.00 & 0.000000 & 0.000000 & 0.000000 & \\
\Rightarrow 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 & \\
\hline
\end{array}
\end{table}

```

Now test the content of the control sequence

```
\rowiii ⇒ 0.50,0.635025,0.635685,0.626425
```

You can see that indeed the third row has been ‘captured for posterity’.

Alternatively, we could save the second column of function values in the control sequence `\colii`:

```

\begin{table}
\renewcommand{\tablewidth}{\textwidth}
\renewcommand{\tableborder}{\hrule}
\renewcommand{\tablecaption}{ $a \sin kx, (a = 2/\pi)$ }
\renewcommand{\tableborder}{\hrule}
\begin{array}{rcccc}
\text{\scriptsize } x & \text{\scriptsize } k = 3.00 & \text{\scriptsize } k = 3.25 & \text{\scriptsize } k = 3.50 & \\
\hline
0.00 & 0.000000 & 0.000000 & 0.000000 & \\
\Rightarrow 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 & \\
\hline
\end{array}
\end{table}

```

$a \sin kx, (a = 2/\pi)$				
x	$k = 3.00$	$k = 3.25$	$k = 3.50$	
\Rightarrow 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 to see what is saved in `\colii` I use TeX's `\meaning` command to show that it is indeed braced pairs:

```
\meaning \colii
 $\Rightarrow$  macro:->{0.00,0.000000},{0.25,0.462191},{0.50,0.635685},{0.75,0.412111},{1.00,-0.068879}
```

Finally, let's use the default `reuse` setting (`reuse=0`) by not entering an explicit `reuse` setting at all. This should save the whole table:

```
\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{wholetable}
```

$a \sin kx, (a = 2/\pi)$				
x	$k = 3.00$	$k = 3.25$	$k = 3.50$	
\Rightarrow 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 check that the whole table has been saved:

$a \sin kx, (vv)$				
x	$k = 3.00$	$k = 3.25$	$k = 3.50$	
\backslash wholetable \Rightarrow 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	

and, indeed it has.

Chapter 3

Reference summary

3.1 Commands defined in `numerica-tables`

`\nmcTabulate`, `\tabulate`

3.2 Settings for `\nmcTabulate`

Row-variable specification §2.1

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

Row-variable column formatting §2.1.1

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

Column-variable specification §2.2.

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

Column-variable header formatting §2.2.1.

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

Function-value formatting §2.5.

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

Whole-of-table formatting §2.4.

key	type	meaning	default
<code>ctitle</code>	token(s)	collective title for function-value columns	
<code>cmidrow</code>	token(s)	inter-header/title row	
<code>rules</code>	char(s)	horizontal rules template	ThB
<code>foot</code>	token(s)	content of footer line	
<code>rpos</code>	int (0...4)	row-variable column position(s)	1
<code>rbloc</code>	comma list	division of rows into blocks	
<code>rblocsep</code>	length	extra spacing between blocks of rows	1 ex

Miscellaneous settings §2.3, 2.7.1.

key	type	meaning	default
<code>multifn</code>	char	multi-function separator	,
<code>valign</code>	char (t/m/b)	vertical alignment	m