numerica-tables

Andrew Parsloe
(ajparsloe@gmail.com)

February 15, 2021

Abstract

In this module of the `numerica` package a command is defined which enables the creation of multi-column tables of function values in a wide variety of table styles.

Note:

- This document applies to version 1.0.0 of `numerica-tables.def`.
- Reasonably recent versions of the LaTeX3 bundles `l3kernel` and `l3packages` are required.
- 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.

Contents

1 Introduction **1**
- 1.1 Shared syntax . . . 1
  - 1.1.1 Inherited settings . . . 2

2 `\nmcTabulate`-specific settings **4**
- 2.1 Row variable settings . . . 4
  - 2.1.1 Row-variable column formatting . . . 6
  - 2.1.2 Multiple function tables . . . 10
- 2.2 Column variable settings . . . 11
  - 2.2.1 Column header formatting . . . 13
- 2.3 Whole-of-table formatting . . . 17
  - 2.3.1 Title for function-value columns: `ctitle` . . . 17
  - 2.3.2 Rules: `rules` setting . . . 18
  - 2.3.3 Footer row: `foot` setting . . . 19
  - 2.3.4 Second row variable column . . . 20
  - 2.3.5 Separating blocks of rows: `rbloc` . . . 21
- 2.4 Function value formatting . . . 23
  - 2.4.1 Trailing optional argument . . . 23
  - 2.4.2 Padding the exponent: `(pad)` . . . 24
  - 2.4.3 Accommodating signs: `signs` . . . 25
  - 2.4.4 Differences: `diffs` . . . 26
  - 2.4.5 Formatting special values: `Q?` and `A!` . . . 28
- 2.5 Table placement . . . 29
  - 2.5.1 Vertical alignment . . . 29
- 2.6 Star option . . . 30
  - 2.6.1 Scientific notation . . . 31
- 2.7 The `reuse` setting . . . 31

3 Nesting **33**

4 Reference summary **35**
- 4.1 Commands defined in `numerica-tables` . . . 35
- 4.2 Settings for `\nmcTabulate` . . . 35

Chapter 1

Introduction

Calling `numerica` with the `tables` package option in the preamble,

```
\usepackage[tables]{numerica}
```

gives access to a command `\nmcTabulate` for creating tables of function values. This command is defined in the package `numerica-tables.def` which is loaded with `numerica.sty` when the `tables` option is used. Note that `\nmcTabulate` uses the `booktabs` package for the construction of its tables. The `booktabs` package is loaded automatically (provided it is available in your TEX system) when `numerica` is loaded with the `tables` option.

1.1 Shared syntax

The `\nmcTabulate` command (short-name form `\tabulate`) shares the syntax of `\nmcEvaluate` (see `numerica-basics.pdf`) and of `\nmcIterate`, `\nmcSolve` and `\nmcRecur` (see `numerica-plus.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;
2. `[settings]` comma-separated list of *key=value* settings;
3. `{expr.}` mandatory argument specifying the mathematical expression in LATEX form to be tabulated;
4. `[vv-list]` comma-separated list of *variable=value* items;
5. `[num. format]` optional format specification for presentation of the numerical result (rounding, padding with zeros, scientific notation)

Table 1.1: Settings options inherited from `\nmcEvaluate`

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

Unlike `\nmcEvaluate` and the other commands, for `\nmcTabulate`

- it makes no difference to the display of the result whether the command wraps around math delimiters, is wrapped within math delimters, 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 neither is mandatory in the LaTeX sense, each contains items necessary for the construction of any table of function values.

1.1.1 Inherited settings

To create a table we need to specify a function to tabulate – this fills the main (mandatory) argument. We also need to decide how the function values are to be displayed – to how many decimal places and whether padded with

zeros. These matters are determined by what is entered in the trailing optional argument, just as with `\eval`, `\iter`, `\solve` and `\recur`, although there are some addtional complications peculiar to tables which are more appropriately treated in the next chapter.

Many of the settings available to the `\eval` command are also available to `\nmcTabulate`. To save switching between documents I reproduce the table of options found in `numerica-basics.pdf` (with only the punctuation `p` setting missing), although for discussion of the options you will need to refer to that document. But note that the `dbg` key is of limited usefulness for tables: `dbg=5` and `dbg=7` display floating point values in their 'internal' format, and for `dbg=7`, no result is displayed.

Chapter 2

\nmcTabulate-specific 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.

2.1 Row variable settings

Deciding on a function to tabulate (entered in the main or mandatory argument of `\nmcTablate`) will inevitably also mean deciding on the tabulation variable (the *row* variable `rvar`), and then what value to start tabulating from (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 the start value of the tabulation variable in the vv-list. The reason for this placement is that for more complicated functions other parameters in the function and therefore in the vv-list may depend on the

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	either `rstop` or `rows`
`rows`	int	number of rows	
`rspec`	comma list	`{start, step, stop}` or `{start, step, (rows)}`	short form spec.

row variable. Therefore it is always placed in the vv-list.

The difference in appearance of the tables results from padding with zeros in the second (the asterisk in the trailing optional argument). 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.

```
\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`) 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 second and third tables in the next example use an abbreviated form of the row variable specification (`rspec`). This is a 3-element comma list of the form `{rvar,rstep,rstop}` or `{rvar,rstep,(rows)}`. Parentheses around the third item signify that it is `rows` rather than `rstop` that is being specified. In the example below, the second table specifies `rstop` (value `1`), the third `rows` (value `6`).

It is worth noting that `rstep` and `rstop` can be LaTeX expressions – for instance `rstop=\sin \pi/2` (just as the expression for the initial value in the vv-list can be). However `rows` can only be a simple integer expression – an integer or integers linked by some or all of `+ - * / ( )`.

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

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

$\Longrightarrow$

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

2.1.1 Row-variable column formatting

Various settings are available to format the row variable column in addition to the padding option (*) of the trailing optional argument.

Table 2.2: Row-variable column formatting

key	type	meaning	default
`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 `<int>` mu	`0`
`rpos`	int (`0...4`)	column position(s)	`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`

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 and *prima facie* false result (the first table below). `numerica` 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][*]\qquad
\tabulate[rvar=x,rstep=0.25,rstop=1,rround=2]
  { \sin x }[x=0][*]
```

x	$\sin x$
0.0	0.000000
0.2	0.247404
0.5	0.479426
0.8	0.681639
1.0	0.841471

$\Longrightarrow$

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

2.1.1.2 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 – the 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 bolding (`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 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. 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.

```
\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][*]
\tabulate
  [rvar=x,rstep=0.25,rstop=0.5,rround=2,
```

```
  ralign=c,rfont=bf,rhead=\boldsymbol{x}]
{ \sin x }[x=-0.5][*]
```

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

$\Longrightarrow$

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

$\boldsymbol{x}$	$\sin x$
$\mathbf{-0.50}$	−0.479426
$\mathbf{-0.25}$	−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 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][*]
```

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

$\Longrightarrow$

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

2.1.1.5 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 using `numerica`'s nudge setting `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`.

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. Giving `rhnudge` a *negative* value (`rhnudge=-12` in the example) brings it 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

2.1.1.6 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 §2.3.4;
- Any other integer acts like `rpos=1`.

An example with `rpos=3` is given shortly below, §2.1.2.2.

2.1.1.7 `rvar', rhead', rhnudge'`

These settings become relevant only when `rpos=4`; see §2.3.4.

2.1.2 Multiple 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.

2.1.2.1 By adjoining tables

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)},rhnudge=9]
  { \sin x }[x=0][*]%
\tabulate
  [rpos=0,rspec={x,0.2,(6)},rround=2]
  { \cos x }[x=0][*]%
\tabulate
  [rpos=2,rspec={x,0.2,(6)},rhnudge=9]
  { \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

2.1.2.2 Multiple functions in a single table

However, tabulating more than one function at a time is too common a need to have to resort to adjoining tables. Instead, it suffices to enter the functions in the main argument separated by commas. The critical thing to do is to *precede the first function with a comma.* That initial comma is the signal `numerica` needs to make the internal adjustments for a multi-function table (and note the `o` setting, to indicate the arguments of sin and cos are in degrees):

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

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

$\Longrightarrow$

This table also suggests 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 (§2.3.4) which enables complementary functions to be tabulated in 'half tables'; see *HMF* Tables 4.10–4.12 for the trigonometric functions.

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, would be to create and adjoin separate tables, one per parameter value. But this 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`) of `2` and a stop value (`cstop`) of `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. Note also the setting for `rhead` which shows the

Table 2.3: Column variable specification

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`
`cspec`	comma list	`{cvar,cstep, cstop}` or `{cvar,cstep,(cols)}`	short form spec.

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)},rround=2,
     rhead=x\backslash k,
     cvar=k,cstep=2,cstop=9]
  { \sin kx }[k=3,x=0][*]
```

$\Longrightarrow$

$x\backslash k$	3	5	7	9
0.00	0.000000	0.000000	0.000000	0.000000
0.20	0.564642	0.841471	0.985450	0.973848
0.40	0.932039	0.909297	0.334988	−0.442520
0.60	0.973848	0.141120	−0.871576	−0.772764
0.80	0.675463	−0.756802	−0.631267	0.793668
1.00	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.

And again, as with the row variable, it is possible to condense the specification into a comma list (`cspec`). This is a 3-element comma list of the form `{cvar,cstep,cstop}` or `{cvar,cstep,(cols)}` where the parentheses distinguish `cols` from `cstop` in the third item. Thus, for the preceding table I could have written

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

or

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

or

```
\tabulate
  [rspec={x,0.2,(6)},rround=2,rhead=x\backslash k,
     cspec={k,2,9}]
  { \sin kx }[k=3,x=0][*]
```

and produced the same table.

Although `cstep` and `cstop` can be LaTeX expressions – for instance `cstop=\pi/2` – `cols` can only be a simple integer expression – an integer, or integers linked by some or all of `+ - * / ( )`.

Table 2.4: Column variable header formatting

key	type	meaning	default
`chstyle`	int (0/1/2/3)	header style	`0`
`ctitle`	tokens	single col. alternative header	
`chead`	tokens	user-defined header	
`calign`	char (r/c/l)	column alignment	`r`
`chnudge`	int	nudge header `int` mu	`0`
`chround`	int	column header rounding	`0`

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 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 §2.3.1 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 §2.3.1). 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 environment and positioning.

If both `ctitle` and `chead` are given, the `chead` 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 `rhead=x\backslash k` where the backslash separates row from column variable. *HMF* contains a multitude of instances; 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 apt choice when the columns are narrow, perhaps from using a small rounding value. 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.

`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. Note that the row variable setting `rhead` no longer needs the '`\backslash k`' part since the column variable is now explicitly indicated (the right-hand table).

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

x	$k=3$	5	7
0.00	0.000	0.000	0.000
0.20	0.565	0.841	0.985
0.40	0.932	0.909	0.335
0.60	0.974	0.141	−0.872
0.80	0.675	−0.757	−0.631
1.00	0.141	−0.959	0.657

$\Longrightarrow$

x	$k=3$	$k=5$	$k=7$
0.00	0.000	0.000	0.000
0.20	0.565	0.841	0.985
0.40	0.932	0.909	0.335
0.60	0.974	0.141	−0.872
0.80	0.675	−0.757	−0.631
1.00	0.141	−0.959	0.657

Finally, `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.

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

x	$\sin 3x$	$\sin 5x$	$\sin 7x$
0.00	0.0000	0.0000	0.0000
0.20	0.5646	0.8415	0.9854
0.40	0.9320	0.9093	0.3350
0.60	0.9738	0.1411	−0.8716
0.80	0.6755	−0.7568	−0.6313
1.00	0.1411	−0.9589	0.6570

$\Longrightarrow$

2.2.1.3 User-defined header: `chead`

Perhaps none of the built-in styles appeal? By assigning content to the key `chead`, users can create their own header to the function value columns. `chead` must contain the correct number of tab characters (`&`), allowing for any `\multicolumn`s used, but ignoring the row variable column or columns. It is a header only to the function value columns. The user will need to insert `$` signs as appropriate.

Non-empty content for the `chead` key overrides any `chstyle` setting and in the case of 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 the now familiar example table gives

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

$\Longrightarrow$

x	$k=3$	$k=5$	$k=7$
0.00	0.000000	0.000000	0.000000
0.20	0.564642	0.841471	0.985450
0.40	0.932039	0.909297	0.334988
0.60	0.973848	0.141120	−0.871576
0.80	0.675463	−0.756802	−0.631267
1.00	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 §2.4.3.

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` provides that. In the 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)},rround=2,rhnudge=9,
     cspec={k,2,(3)},chstyle=2,chnudge=12]
  { \sin kx }[k=3,x=0][*]
```

x	$k=3$	$k=5$	$k=7$
0.00	0.000000	0.000000	0.000000
0.20	0.564642	0.841471	0.985450
0.40	0.932039	0.909297	0.334988
0.60	0.973848	0.141120	−0.871576
0.80	0.675463	−0.756802	−0.631267
1.00	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)},rround=2,rhnudge=9,
     cspec={k,2,(3)},chstyle=2,chnudge=-9]
  { \sin kx }[k=100,x=0][3*]
```

x	$k=100$	$k=102$	$k=104$
0.00	0.000	0.000	0.000
0.20	0.913	1.000	0.929
0.40	0.745	0.041	−0.688
0.60	−0.305	−0.998	−0.419
0.80	−0.994	−0.081	0.999
1.00	−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)},rround=2,rhnudge=9,
     cspec={k,0.25,(3)},chstyle=2,chround=2]
  { \sin kx }[k=3,x=0][*]
```

$\Longrightarrow$

x	$k = 3.00$	$k = 3.25$	$k = 3.50$
0.00	0.000000	0.000000	0.000000
0.20	0.564642	0.605186	0.644218
0.40	0.932039	0.963558	0.985450
0.60	0.973848	0.928960	0.863209
0.80	0.675463	0.515501	0.334988
1.00	0.141120	−0.108195	−0.350783

2.3 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
`ctitle`	tokens	collective title for function-value columns	
`rules`	chars	horizontal rules template	`ThB`
`foot`	tokens	content of footer line	
`rpos`	int (`0...4`)	row-variable column position(s)	`1`
`rbloc`	comma list	division of rows into blocks	
`rblocsep`	length	extra spacing between blocks of rows	`1 ex`

2.3.1 Title for function-value columns: `ctitle`

The function value columns have individual headers, formatted in the various ways provided by the settings discussed above, but it can also be helpful to have a collective title for these columns. This is provided by the `ctitle` key. This can be set to whatever you like (e.g. `ctitle=\text{Fred}`), but for more relevant titles there are two in-built settings: `ctitle=*`, which makes the formula the title, and `ctitle=**`, which makes a title of the formula and vv-list. In the example below, it makes no sense to display the vv-list since it contains only the initial values of the row and column variables, already obvious in the table; hence `ctitle=*`.

```
\tabulate
  [rspec={x,0.25,(5)},rround=2,rhnudge=9,
     cspec={k,0.25,(3)},chstyle=2,chround=2,ctitle=*]
```

```
{ \sin kx }[k=3,x=0][*]
```

$\Longrightarrow$

x	$k = 3.00$	$k = 3.25$	$k = 3.50$
		$\sin kx$	
0.00	0.000000	0.000000	0.000000
0.25	0.681639	0.726009	0.767544
0.50	0.997495	0.998531	0.983986
0.75	0.778073	0.647343	0.493920
1.00	0.141120	−0.108195	−0.350783

For an example of `ctitle=**` see §2.3.2.

2.3.2 Rules: `rules` setting

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).[1] 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. Also, to my eye, the table title, $\sin kx$, in the last example seems 'naked' without a rule to emphasize its domain.

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`.

Table 2.6: Rules

char	rule	position	default rule thickness
`T`	top rule	above table	`\heavyrulewidth=.08em`
`t`	title rule	below title	`\cmidrulewidth =.03em`
`h`	header rule	below header	`\lightrulewidth=.05em`
`f`	footer rule	above footer	`\cmidrulewidth =.03em`
`B`	bottom rule	below table	`\heavyrulewidth=.08em`

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

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

`booktabs` package (except for the footer rule, which `booktabs` does not cover; in `numerica` the footer rule is 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 fornula 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` so that they don't show in the vv-list).

```
\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][*]
```

$\Longrightarrow$

	$a \sin kx, \ \ (a = 2/\pi)$		
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

2.3.3 Footer row: `foot` setting

Some tables have a footer row and `numerica` 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 `\multicol`. 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 §2.3.4.

2.3.3.1 Footer functions

It is also possible to use the footer for displaying the values of certain column functions. `numerica` provides five of these. They can be used in the footer (and

only in the footer): `SUM`, `AVE` (average), `MAX`, `MIN` and `DEL` (for Δ=`MAX-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, not the value of `DEL`. The numerical output from each function is automatically wrapped in math delimiters (`$`) so that minus signs display correctly.

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 five footer functions to act on.

```
\tabulate
  [rspec={x,0.25,(5)},rround=2,rhead=x,ralign=r,rhnudge=9,
     cspec={k,0,(5)},chstyle=2,
     chround=2,calign=r,ctitle=**,rules=TthfB,
     foot={\small Func:} & SUM & AVE & MAX & MIN & DEL ]
  { a\sin kx }[a=2/\pi,{k}=3.5,{x}=0][*]
```

$\Longrightarrow$

	$a \sin kx, \quad (a = 2/\pi)$				
x	$k = 3.50$	$k = 3.50$	$k = 3.50$	$k = 3.50$	$k = 3.50$
0.00	0.000000	0.000000	0.000000	0.000000	0.000000
0.25	0.488633	0.488633	0.488633	0.488633	0.488633
0.50	0.626425	0.626425	0.626425	0.626425	0.626425
0.75	0.314439	0.314439	0.314439	0.314439	0.314439
1.00	−0.223316	−0.223316	−0.223316	−0.223316	−0.223316
Func:	1.206181	0.241236	0.626425	−0.223316	0.849741

2.3.4 Second row variable column

In §2.1.1.6 I discussed `rpos=0,1,2,3`. 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 functions of those in the left column.

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.

```
\tabulate[o,rpos=4,rspec={\theta,5,45},rround=0,
  chnudge=14,rvar'=90-\theta,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

where $\theta' = 90 - \theta$. The first half of the tabulation is read normally, down and from the left. The second half of the tabulation is read up and from the right. Note the degree setting `o` in the settings option and

- the use of `rvar'` to specify the values tabulated on the right (`rvar'` defaults to `rvar`);
- the use of `rhead'` to specify the content of the header for the right-hand row variable column (`rhead'` defaults to `rvar'`);
- the footer setting `foot=*` to obtain the header reversed in the footer;
- a rule *above* the footer row specified by the `f` added to the `rules` setting, `rules=ThfB`.

Although there is a significant space saving with tables of this kind (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 (stemming from $y! = \Gamma(x-1)$) in the row variable column on the right; Table 6.5 in effect uses $\langle 1/x \rangle$ (the nearest integer to $1/x$) for the row variable on the right.

2.3.5 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, marking 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,10,90},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

2.3.5.1 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. The default is `1 ex`.

Table 2.7: Function value formatting

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!`	tokens	special cell formatting	

2.4 Function value formatting

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?

2.4.1 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. Adding a prime to the `x` 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 borderline distracting and would certainly be 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,1,3},rround=0]
  { e^x}[x=-5][*x]\qquad
\tabulate[rspec={x,1,3},rround=0,calign=l]
  { e^x}[x=-3][*x']\qquad
```

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}

$\Longrightarrow$

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}

2.4.1.1 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,1,3},rround=0,chnudge=24]
  { e^x}[x=-3][*t]\qquad
\tabulate[rspec={x,1,3},rround=0,chnudge=24]
  { e^x}[x=-3][*t']
```

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	$(1)\,2.008554$

$\Longrightarrow$

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	$(1)\,2.008554$

2.4.2 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` offers the setting

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

to achieve the effect. `<integer>` is the number of digits/characters to pad to. In the last example, the setting is `(pad)=2`. Here it is repeated with this setting:

```
\tabulate[rspec={x,1,3},rround=0,chnudge=24,(pad)=2]
  { e^x}[x=-3][*t]\qquad
\tabulate[rspec={x,1,3},rround=0,chnudge=24,(pad)=2]
  { e^x}[x=-3][*t']
```

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	$(\ \ 1)\,2.008554$

$\Longrightarrow$

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.

2.4.3 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` 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 immediately preceding or following a negative number;
- `signs=-1` inserts a + sign between exponent and significand of any non-negative number immediately preceding or following a negative number, and 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;

With the 't-notation', the negative values and `signs=2` seem the appropriate ones to use :

```
\tabulate[rspec={x,1,3},rround=0,chnudge=9,
    (pad)=2,signs=-2]
  { e^x-1}[x=-3][4*t']\qquad
\tabulate[rspec={x,1,3},rround=0,chnudge=9,
```

```
    (pad)=2,signs=-1]
  { e^x-1}[x=-3][4*t']\qquad
\tabulate[rspec={x,1,3},rround=0,chnudge=9,
    (pad)=2,signs=2]
  { e^x-1}[x=-3][4*t']
```

$\Longrightarrow$

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`.

However 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 give appropriate results (I've included also the default setting):

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

$\Longrightarrow$

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
-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 function values changes every few entries and `signs=1` when there are runs of entries of the same sign. They would use the middle table of the three here.

2.4.4 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,0.01,1.05},rround=2,
  rhnudge=9,chnudge=21,diffs=3]
    { \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}

I have deliberately chosen the function and settings here – particularly `diffs=3` – to give a good result. It is easy to get this wrong. The evidence will be 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 second table the function is *decreasing*, $-\sinh x$, to show 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,0.01,1.05},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=21,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. 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.

2.4.5 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'). `numerica` 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 their combinations like `!= >= <=` etc., and parentheses. For everything `l3fp` makes available see the relevant chapter in `Interface3.pdf`, part of the documentation that comes with the LaTeX3 package `l3kernel`. In addition to these components, `numerica` 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 LaTeX 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.

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,1,15},rround=0,rpos=2,rules=Tth,
     cspec={m,1,5},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

2.5 Table placement

There is only one setting in this category that is part of `numerica` as such, the tabular vertical alignment option; see §2.5.1. But LaTeX 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).

2.5.1 Vertical alignment

By writing

```
valign = <char>
```

where `<char>` is one of `t` 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. By default (no `valign` setting or `valign` equated to some character other than `t` or `b`) the middle of the table is aligned with the text baseline. 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][*] \qquad
B \tabulate[rspec={x,0.2,1}]
```

```
  { \tan x }[x=0][*] \qquad
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 §2.1.2.1, 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 either a top or bottom alignment is necessary.

2.6 Star option

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. I.e., like the other starred commands, `\eval*`, `\iter*`, `\solve*` and `\recur*`, `\tabulate*` outputs a single number:

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

$\Longrightarrow$ 0.6235. Ineed, if you omit the `Q?` and `A!` settings from the previous table then this is the value that follows 0.5000 in the `m=2` column – the first function value encountered satisfying the query `Q?`.

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

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

$\Longrightarrow$!!! 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:

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

$\Longrightarrow$!!! No * option query `Q?` in: settings. !!!

2.6.1 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 `\tabulate*` command between math delimiters, otherwise the `\times` symbol resulting from the `x` option will generate a LaTeX error ('Missing $ inserted'):

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

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

2.7 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`.

```
\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,reuse=-3]
  { a\sin kx }[a=2/\pi,{k}=3,{x}=0][*] \reuse[rowiii]
```

$\Longrightarrow$

	$a \sin kx, \ \ (a = 2/\pi)$		
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

`\rowiii` $\Longrightarrow$ 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 in the control sequence `\colii`:

```
[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,reuse=2]
  { a\sin kx }[a=2/\pi,{k}=3,{x}=0][*] \reuse[colii]
```

$\Longrightarrow$

	$a \sin kx, \ \ (a = 2/\pi)$		
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

To see what is saved in `\colii` I use TEX's `\meaning` command to show that it is indeed braced pairs:

```
\meaning \colii
```

$\Longrightarrow$ macro:->{0.00,0.000000},{0.25,0.462191},{0.50,0.635685},{0.75,0.412111},{1.00,-0.068879}

Chapter 3

Nesting

Provided the starred form of any one of `\eval`, `\iter`, `\solve`, `recur` actually does produce a numerical result and not an error message then it can be nested within the main argument of any one of the other commands, including itself. The associated document `numerica-plus.pdf` has a number of illustrations of this. The starred form can also be used in the vv-list of any one of the commands, including itself. `numerica-plus.pdf` has an example of `\solve*` being used in the vv-list of an `\eval` command, and the document `numerica-basics.pdf` shows examples of `\eval*` being used in the vv-list of an `\eval` command.

These nesting properties are also true of `\tabulate`. It's starred form can be used in other commands. Other commands' starred forms can be used in `\tabulate`.

For the first, `\tabulate*` inside `\eval`, I have also used the example to show how formatting settings (shown for the first `\tabulate*` in the example) can in fact be dispensed with (as shown for the second `\tabulate*`). Only *functional* settings are required. Hence there is no `A!` setting in either. It is the *question*, `Q?`, not the answer, that isolates the single-value that is the result of `\tabulate*`.

```
\eval{$
  (\tabulate*
    [rspec={n,1,15},rround=0,rpos=2,rules=Tth,
       cspec={m,1,5},ctitle=*,chstyle=2,
         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]
```

$$\implies (0.9135)\sinh t + (-0.7071)\cosh t = 0.6529, \quad (t = 2).$$

For using the starred forms of other commands inside `\tabulate`, it makes more sense to use one or more of `\iter*`, `\solve*` or `\recur*` inside `\tabulate` than it does `\eval*`, but that assumes the reader is familiar with those com mands and that `numerica-plus.def` is loaded for this document, which it is not. So, here is a rather pointless illustration of nesting `\eval*` in `\tabulate`. (At least it is reassuring that the values of $\Delta(N)$ are identical in the two columns.)

```
\tabulate[rspec={N,100,500},rround=0,ctitle=\Delta(N),
          chead=eval*&direct]
   { ,\eval*{ \sum_{n=1}^N 1/n-\ln N-\gamma },
              \sum_{n=1}^N 1/n-\ln N-\gamma }[N=100][4*]
```

$\Longrightarrow$

	$\Delta(N)$	
N	eval*	direct
100	0.0050	0.0050
200	0.0025	0.0025
300	0.0017	0.0017
400	0.0012	0.0012
500	0.0010	0.0010

Chapter 4

Reference summary

4.1 Commands defined in `numerica-tables`

`\nmcTabulate, \tabulate`

4.2 Settings for `\nmcTabulate`

Row variable specification §2.1

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, stop}` or `{start, step, (rows)}`	short form spec.

Row variable column formatting §2.1.1

key	type	meaning	default
`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 `<int>` mu	`0`
`rpos`	int (`0...4`)	column position(s)	`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`

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, cstop}` or `{cvar,cstep,(cols)}`	short form spec.

Column variable header formatting §2.2.1.

key	type	meaning	default
`chstyle`	int (0/1/2/3)	header style	`0`
`ctitle`	tokens	single col. alternative header	
`chead`	tokens	user-defined header	
`calign`	char (r/c/l)	column alignment	`r`
`chnudge`	int	nudge header `int` mu	`0`
`chround`	int	column header rounding	`0`

Function value formatting §2.4.

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!`	tokens	special cell formatting	

Whole-of-table formatting §2.3.

key	type	meaning	default
`ctitle`	tokens	collective title for function-value columns	
`rules`	chars	horizontal rules template	`ThB`
`foot`	tokens	content of footer line	
`rpos`	int (`0...4`)	row-variable column position(s)	`1`
`rbloc`	comma list	division of rows into blocks	
`rblocsep`	length	extra spacing between blocks of rows	`1 ex`

Table placement §2.5.1.

key	type	meaning	default
`valign`	char (t/b)	vertical alignment	