Manual for Package PgfplotsTable

Component of pgfplots, Version 1.0

http://sourceforge.net/projects/pgfplots

Christian Feuersänger*
Institut für Numerische Simulation
Universität Bonn, Germany

June 11, 2008

Abstract

This package reads tab-separated numerical tables from input and generates code for pretty-printed LaTeX-tabulars. It rounds to the desired precision and prints it in different number formatting styles.

Contents

1 Introduction

PgfplotsTable is a lightweight sub-package of pgfplots which employs its table input methods and the number formatting techniques to convert tab-separated tables into tabulars.

It's input is a text file containing space separated rows, possibly starting with column names. Its output is a LaTeX tabular which contains selected columns of the text table, rounded to the desired precision, printed in the desired number format (fixed point, integer, scientific etc).

It is used with

```
\usepackage{pgfplots}
```

and requires pgfplots and pgf installed.

*http://wissrech.ins.uni-bonn.de/people/feuersaenger

2 Text table input format

PgfplotsTable works with plain text file tables in which entries ("cells") are separated by a separation character. The initial separation character is "white space" which means "at least one space or tab". Those tables can have a header line which contains column names and most other columns typically contain numerical data.
The following listing shows `pgfplotstable.example1.dat` and is used often throughout this documentation.

```
# Convergence results
# fictional source, generated 2008
level   dof     error1  error2  info    grad(log(dof),log(error1))      quot(error1)
1       9       2.50000000e-01  7.57858283e-01  48      0       0
2       25      6.25000000e-02  5.00000000e-01  25      -1.35691545e+00 4
3       81      1.56250000e-02  2.87174589e-01  41      -1.17924958e+00 4
4       289     3.90625000e-03  1.43587294e-01  8       -1.08987331e+00 4
5       1089    9.76562500e-04  4.41941738e-02  22      -1.04500712e+00 4
6       4225    2.44140625e-04  1.69802322e-02  46      -1.02252239e+00 4
7       16641   6.10351562e-05  8.20091159e-03  40      -1.01126607e+00 4
8       66049   1.52587891e-05  3.90625000e-03  48      -1.00563427e+00 3.99999999e+00
9       263169  3.81469727e-06  1.95312500e-03  33      -1.00281745e+00 4.00000001e+00
10      1050625 9.53674316e-07  9.76562500e-04  2       -1.00140880e+00 4.00000001e+00
```

Lines starting with '`%`' or '`#`' are considered to be comment lines and are ignored.

There is future support for a second header line which must start with '`$flags `' (the space is obligatory, even if the column separator is *not* space!). Currently, such a line is ignored. It may be used to provide number formatting options like precision and number format.

`/pgfplots/table/col sep=space|comma|semicolon|colon|braces` (no default, initially `space`)

Specifies the column separation character for table reading. The initial choice, `space` means "at least one white space". White spaces are tab stops or spaces (newlines always delimit lines).

For example, the file `pgfplotstable.example1.csv` uses commas as separation characters.

```
# Convergence results
# fictional source  generated 2008
level,dof,error1,error2,info,{grad(log(dof),log(error1))},quot(error1)
1,9,2.50000000e-01,7.57858283e-01,48,0,0
2,25,6.25000000e-02,5.00000000e-01,25,-1.35691545e+00,4
3,81,1.56250000e-02,2.87174589e-01,41,-1.17924958e+00,4
4,289,3.90625000e-03,1.43587294e-01,8,-1.08987331e+00,4
5,1089,9.76562500e-04,4.41941738e-02,22,-1.04500712e+00,4
6,4225,2.44140625e-04,1.69802322e-02,46,-1.02252239e+00,4
7,16641,6.10351562e-05,8.20091159e-03,40,-1.01126607e+00,4
8,66049,1.52587891e-05,3.90625000e-03,48,-1.00563427e+00,3.99999999e+00
9,263169,3.81469727e-06,1.95312500e-03,33,-1.00281745e+00,4.00000001e+00
10,1050625,9.53674316e-07,9.76562500e-04,2,-1.00140880e+00,4.00000001e+00
```

Thus, we need to specify `col sep=comma` when we read it.

level	dof	error1	error2	info	grad(log(dof),log(error1))	quot(error1)
1	9	0.25	0.76	48	0	0
2	25	$6.25 \cdot 10^{-2}$	0.5	25	−1.36	4
3	81	$1.56 \cdot 10^{-2}$	0.29	41	−1.18	4
4	289	$3.91 \cdot 10^{-3}$	0.14	8	−1.09	4
5	1,089	$9.77 \cdot 10^{-4}$	$4.42 \cdot 10^{-2}$	22	−1.05	4
6	4,225	$2.44 \cdot 10^{-4}$	$1.7 \cdot 10^{-2}$	46	−1.02	4
7	16,641	$6.1 \cdot 10^{-5}$	$8.2 \cdot 10^{-3}$	40	−1.01	4
8	66,049	$1.53 \cdot 10^{-5}$	$3.91 \cdot 10^{-3}$	48	−1.01	4
9	$2.63 \cdot 10^{5}$	$3.81 \cdot 10^{-6}$	$1.95 \cdot 10^{-3}$	33	−1	4
10	$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$	$9.77 \cdot 10^{-4}$	2	−1	4

```
\pgfplotstabletypesetfile[col sep=comma]{pgfplotstable.example1.csv}
```

You may call `\pgfplotstableset{col sep=comma}` once in your preamble if all your tables uses commas as column separator.

Please note that if cell entries (for example column names) contain the separation character, you need to enclose the column entry in *braces*: `{grad(log(dof),log(error1))}`. If you want to use unmatched braces, you need to write a backslash before the brace. For example the name '`column{withbrace`' needs to be written as '`column\{withbrace`'.

Furthermore, if you need empty cells in case `col sep=space`, you have to provide `{}` to delimit such a cell since `col sep=space` uses *at least* one white space (consuming all following ones).

The value `col sep=braces` is special since it actually uses two separation characters. Every single cell entry is delimited by an opening and a closing brace, `{`⟨*entry*⟩`}`, for this choice. Furthermore, any wite spaces (spaces and tabs) between cell entries are *skipped* in case `braces` until the next `{`⟨*entry*⟩`}` is found.

`\pgfplotstabletypesetfile`[⟨*optional arguments*⟩]`{`⟨*file name*⟩`}`

This command loads the table file `{`⟨*file name*⟩`}` and typesets it using the current configuration of number formats and table options.

level	dof	error1	error2	info	grad(log(dof),log(error1))	quot(error1)
1	9	0.25	0.76	48	0	0
2	25	$6.25 \cdot 10^{-2}$	0.5	25	−1.36	4
3	81	$1.56 \cdot 10^{-2}$	0.29	41	−1.18	4
4	289	$3.91 \cdot 10^{-3}$	0.14	8	−1.09	4
5	1,089	$9.77 \cdot 10^{-4}$	$4.42 \cdot 10^{-2}$	22	−1.05	4
6	4,225	$2.44 \cdot 10^{-4}$	$1.7 \cdot 10^{-2}$	46	−1.02	4
7	16,641	$6.1 \cdot 10^{-5}$	$8.2 \cdot 10^{-3}$	40	−1.01	4
8	66,049	$1.53 \cdot 10^{-5}$	$3.91 \cdot 10^{-3}$	48	−1.01	4
9	$2.63 \cdot 10^{5}$	$3.81 \cdot 10^{-6}$	$1.95 \cdot 10^{-3}$	33	−1	4
10	$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$	$9.77 \cdot 10^{-4}$	2	−1	4

```
\pgfplotstabletypesetfile{pgfplotstable.example1.dat}
```

The configuration can be customized with ⟨*optional arguments*⟩.

level	dof	error1	error2	info	grad(log(dof),log(error1))	quot(error1)
1	9	0.25	0.76	48	0	0
2	25	6.25_{-2}	0.5	25	−1.36	4
3	81	1.56_{-2}	0.29	41	−1.18	4
4	289	3.91_{-3}	0.14	8	−1.09	4
5	1,089	9.77_{-4}	4.42_{-2}	22	−1.05	4
6	4,225	2.44_{-4}	1.70_{-2}	46	−1.02	4
7	16,641	6.10_{-5}	8.20_{-3}	40	−1.01	4
8	66,049	1.53_{-5}	3.91_{-3}	48	−1.01	4
9	2.63_{5}	3.81_{-6}	1.95_{-3}	33	−1	4
10	1.05_{6}	9.54_{-7}	9.77_{-4}	2	−1	4

```
\pgfplotstabletypesetfile[sci subscript,sci zerofill]{pgfplotstable.example1.dat}
```

`\pgfplotstableread`{⟨*file name*⟩}{⟨*\macro*⟩}

Loads the table `{`⟨*file name*⟩`}` into a TEX-macro ⟨*\macro*⟩. This macro can than be used several times.

dof	error1	dof	error2
9	0.25	9	0.76
25	$6.25 \cdot 10^{-2}$	25	0.5
81	$1.56 \cdot 10^{-2}$	81	0.29
289	$3.91 \cdot 10^{-3}$	289	0.14
1,089	$9.77 \cdot 10^{-4}$	1,089	$4.42 \cdot 10^{-2}$
4,225	$2.44 \cdot 10^{-4}$	4,225	$1.7 \cdot 10^{-2}$
16,641	$6.1 \cdot 10^{-5}$	16,641	$8.2 \cdot 10^{-3}$
66,049	$1.53 \cdot 10^{-5}$	66,049	$3.91 \cdot 10^{-3}$
$2.63 \cdot 10^{5}$	$3.81 \cdot 10^{-6}$	$2.63 \cdot 10^{5}$	$1.95 \cdot 10^{-3}$
$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$	$1.05 \cdot 10^{6}$	$9.77 \cdot 10^{-4}$

```
\pgfplotstableread{pgfplotstable.example1.dat}\table
\pgfplotstabletypeset[columns={dof,error1}]\table
\hspace{2cm}
\pgfplotstabletypeset[columns={dof,error2}]\table
```

\pgfplotstabletypeset[⟨*optional arguments*⟩]{⟨*\macro*⟩}

Works in the same way as \pgfplotstabletypesetfile, but it accepts an already loaded table.

3 Selecting columns

/pgfplots/table/columns={⟨*comma-separated-list*⟩} (no default, initially all available ones)

Selects particular columns the table. If this option is missing, all available columns will be selected.

Inside of {⟨*comma-separated-list*⟩}, column names as they appear in the table's header are expected. If there is no header, simply use column names. If there are column names, the special syntax [index]⟨*integer*⟩ can be used to select columns by index. The first column has index 0.

dof	level	info
9	1	48
25	2	25
81	3	41
289	4	8
1,089	5	22
4,225	6	46
16,641	7	40
66,049	8	48
$2.63 \cdot 10^5$	9	33
$1.05 \cdot 10^6$	10	2

```
\pgfplotstabletypesetfile[columns={dof,level,[index]4}]{pgfplotstable.example1.dat}
```

/pgfplots/table/columns/⟨*column name*⟩/.style={⟨*key-value-list*⟩}

Sets all options in {⟨*key-value-list*⟩} exclusively for {⟨*column name*⟩}.

level	Dof	L_2	A	info	grad(log(dof),log(error1))	quot(error1)
1	9	2.500_{-1}	7.58_{-1}	48	0	0
2	25	6.250_{-2}	5.00_{-1}	25	−1.36	4
3	81	1.563_{-2}	2.87_{-1}	41	−1.18	4
4	289	3.906_{-3}	1.44_{-1}	8	−1.09	4
5	1,089	9.766_{-4}	4.42_{-2}	22	−1.05	4
6	4,225	2.441_{-4}	1.70_{-2}	46	−1.02	4
7	16,641	6.104_{-5}	8.20_{-3}	40	−1.01	4
8	66,049	1.526_{-5}	3.91_{-3}	48	−1.01	4
9	263,169	3.815_{-6}	1.95_{-3}	33	−1	4
10	1,050,625	9.537_{-7}	9.77_{-4}	2	−1	4

```
\pgfplotstabletypesetfile[
    columns/error1/.style={
        column name=$L_2$,
        sci,sci zerofill,sci subscript,
        precision=3},
    columns/error2/.style={
        column name=$A$,
        sci,sci zerofill,sci subscript,
        precision=2},
    columns/dof/.style={
        int detect,
        column name=\textsc{Dof}
    }
]
    {pgfplotstable.example1.dat}
```

If your column name contains commas ‘,’, slashes ‘/’ or equal signs ‘=’, you need to enclose the column name in braces.

Dof	L_2	slopes L_2
9	2.500_{-1}	0.0
25	6.250_{-2}	−1.4
81	1.563_{-2}	−1.2
289	3.906_{-3}	−1.1
1,089	9.766_{-4}	−1.0
4,225	2.441_{-4}	−1.0
16,641	6.104_{-5}	−1.0
66,049	1.526_{-5}	−1.0
263,169	3.815_{-6}	−1.0
1,050,625	9.537_{-7}	−1.0

```
\pgfplotstabletypesetfile[
    columns={dof,error1,{grad(log(dof),log(error1))}},
    columns/error1/.style={
        column name=$L_2$,
        sci,sci zerofill,sci subscript,
        precision=3},
    columns/dof/.style={
        int detect,
        column name=\textsc{Dof}},
    columns/{grad(log(dof),log(error1))}/.style={
        column name=slopes $L_2$,
        fixed,fixed zerofill,
        precision=1}
]
    {pgfplotstable.example1.dat}
```

/pgfplots/table/column type={⟨*tabular column type*⟩} (no default, initially c)

Contains the column type for tabular.

If all column types are empty, the complete argument is skipped (assuming that no tabular environment is generated).

Use \pgfplotstableset{column type/.add={⟨*before*⟩}{⟨*after*⟩}} to *modify* a value instead of overwriting it. The .add key handler works for other options as well.

dof	error1	info
9	0.25	48
25	$6.25 \cdot 10^{-2}$	25
81	$1.56 \cdot 10^{-2}$	41
289	$3.91 \cdot 10^{-3}$	8
1,089	$9.77 \cdot 10^{-4}$	22
4,225	$2.44 \cdot 10^{-4}$	46
16,641	$6.1 \cdot 10^{-5}$	40
66,049	$1.53 \cdot 10^{-5}$	48
$2.63 \cdot 10^{5}$	$3.81 \cdot 10^{-6}$	33
$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$	2

```
\pgfplotstabletypesetfile[
    columns={dof,error1,info},
    column type/.add={|}{}% results in '|c'
]
    {pgfplotstable.example1.dat}
```

/pgfplots/table/assign column name/.code={⟨...⟩}

Allows to *modify* the value of column name.

Argument #1 is the current column name, that means after evaluation of column name. After assign column name, a new (possibly modified) value for column name should be set.

That means you can use column name to assign the name as such and assign column name to generate final TEX code (for example to insert \multicolumn{1}{c}{#1}).

Default is empty which means no change.

/pgfplots/table/multicolumn names={⟨*tabular column type*⟩} (style, no default, initially c)

A style which typesets each column name using a \multicolumn{1}{⟨*tabular column type*⟩}{⟨*the column name*⟩} statement.

/pgfplots/table/dec sep align={⟨*header column type*⟩} (style, no default, initially c)

A style which employs the current decimal separator as alignment character.

The first argument determines the alignment of the header column.

The style dec sep align actually introduces two new tabular columns, namely r@{}l. It introduces multicolumns for column names accordingly and handles numbers which do not have a decimal separator.

dof	error1	error2	info	grad(log(dof),log(error1))
9	0.25	7.58_{-1}	48	0
25	$6.25 \cdot 10^{-2}$	5.00_{-1}	25	−1.36
81	$1.56 \cdot 10^{-2}$	2.87_{-1}	41	−1.18
289	$3.91 \cdot 10^{-3}$	1.44_{-1}	8	−1.09
1,089	$9.77 \cdot 10^{-4}$	4.42_{-2}	22	−1.05
4,225	$2.44 \cdot 10^{-4}$	1.70_{-2}	46	−1.02
16,641	$6.1 \cdot 10^{-5}$	8.20_{-3}	40	−1.01
66,049	$1.53 \cdot 10^{-5}$	3.91_{-3}	48	−1.01
$2.63 \cdot 10^{5}$	$3.81 \cdot 10^{-6}$	1.95_{-3}	33	−1
$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$	9.77_{-4}	2	−1

```
\pgfplotstabletypesetfile[
    columns={dof,error1,error2,info,{grad(log(dof),log(error1))}},
    columns/error1/.style={dec sep align},
    columns/error2/.style={sci,sci subscript,sci zerofill,dec sep align},
    columns/info/.style={fixed,dec sep align},
    columns/{grad(log(dof),log(error1))}/.style={fixed,dec sep align}
]
    {pgfplotstable.example1.dat}
```

Or with comma as decimal separator:

dof	error1	error2	info	grad(log(dof),log(error1))
9	0,25	$7{,}58_{-1}$	48	0
25	$6{,}25 \cdot 10^{-2}$	$5{,}00_{-1}$	25	−1,36
81	$1{,}56 \cdot 10^{-2}$	$2{,}87_{-1}$	41	−1,18
289	$3{,}91 \cdot 10^{-3}$	$1{,}44_{-1}$	8	−1,09
1.089	$9{,}77 \cdot 10^{-4}$	$4{,}42_{-2}$	22	−1,05
4.225	$2{,}44 \cdot 10^{-4}$	$1{,}70_{-2}$	46	−1,02
16.641	$6{,}1 \cdot 10^{-5}$	$8{,}20_{-3}$	40	−1,01
66.049	$1{,}53 \cdot 10^{-5}$	$3{,}91_{-3}$	48	−1,01
$2{,}63 \cdot 10^{5}$	$3{,}81 \cdot 10^{-6}$	$1{,}95_{-3}$	33	−1
$1{,}05 \cdot 10^{6}$	$9{,}54 \cdot 10^{-7}$	$9{,}77_{-4}$	2	−1

```
\pgfplotstabletypesetfile[
    use comma,
    columns={dof,error1,error2,info,{grad(log(dof),log(error1))}},
    columns/error1/.style={dec sep align},
    columns/error2/.style={sci,sci subscript,sci zerofill,dec sep align},
    columns/info/.style={fixed,dec sep align},
    columns/{grad(log(dof),log(error1))}/.style={fixed,dec sep align}
]
    {pgfplotstable.example1.dat}
```

It may be advisable to use the `zerofill` variants to force at least one digit after the decimal separator.

/pgfplots/table/dcolumn={⟨*tabular column type*⟩}{⟨*type for column name*⟩} (style, no default, initially {D{.}{.}{2}}{c})

A style which can be used together with the `dcolumn` package of David Carlisle. It also enables alignment at the decimal separator. However, the decimal separator needs to be exactly one character which is incompatible with '{,}' (the default setting for `use comma`).

/pgfplots/table/every first column (style, no value)

A style which is installed for every first column only.

level	dof	error1	error2	info	grad(log(dof),log(error1))	quot(error1)
1	9	0.25	0.76	48	0	0
2	25	$6.25 \cdot 10^{-2}$	0.5	25	−1.36	4
3	81	$1.56 \cdot 10^{-2}$	0.29	41	−1.18	4
4	289	$3.91 \cdot 10^{-3}$	0.14	8	−1.09	4
5	1,089	$9.77 \cdot 10^{-4}$	$4.42 \cdot 10^{-2}$	22	−1.05	4
6	4,225	$2.44 \cdot 10^{-4}$	$1.7 \cdot 10^{-2}$	46	−1.02	4
7	16,641	$6.1 \cdot 10^{-5}$	$8.2 \cdot 10^{-3}$	40	−1.01	4
8	66,049	$1.53 \cdot 10^{-5}$	$3.91 \cdot 10^{-3}$	48	−1.01	4
9	$2.63 \cdot 10^{5}$	$3.81 \cdot 10^{-6}$	$1.95 \cdot 10^{-3}$	33	−1	4
10	$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$	$9.77 \cdot 10^{-4}$	2	−1	4

```
\pgfplotstabletypesetfile[
 every head row/.style={before row=\hline,after row=\hline\hline},
 every last row/.style={after row=\hline},
 every first column/.style={
  column type/.add={|}{}
 },
 every last column/.style={
  column type/.add={}{|}
 }]
    {pgfplotstable.example1.dat}
```

/pgfplots/table/every last column (style, no value)

A style which is installed for every last column only.

/pgfplots/table/every even column (style, no value)

A style which is installed for every column with even column index (starting with 0).

```
\pgfplotstableset{
    columns={dof,error1,{grad(log(dof),log(error1))},info},
    columns/error1/.style={
        column name=$L_2$,
        sci,sci zerofill,sci subscript,
        precision=3},
    columns/dof/.style={
        int detect,
        column name=\textsc{Dof}},
    columns/{grad(log(dof),log(error1))}/.style={
        column name=slopes $L_2$,
        fixed,fixed zerofill,
        precision=1}}
```

Dof	L_2	slopes L_2	info
9	2.500_{-1}	0.0	48
25	6.250_{-2}	−1.4	25
81	1.563_{-2}	−1.2	41
289	3.906_{-3}	−1.1	8
1,089	9.766_{-4}	−1.0	22
4,225	2.441_{-4}	−1.0	46
16,641	6.104_{-5}	−1.0	40
66,049	1.526_{-5}	−1.0	48
263,169	3.815_{-6}	−1.0	33
1,050,625	9.537_{-7}	−1.0	2

```
% \usepackage{colortbl}
\pgfplotstabletypesetfile[
 every even column/.style={
  column type/.add={>{\columncolor[gray]{.8}}}{}
}]
    {pgfplotstable.example1.dat}
```

/pgfplots/table/every odd column (style, no value)

A style which is installed for every column with odd column index (starting with 0).

\pgfplotstablecol

During the evaluation of row or column options, this command expands to the current columns' index.

\pgfplotstablerow

During the evaluation of row or column options, this command expands to the current rows' index.

4 Configuring row appearance

/pgfplots/before row={⟨*TEX code*⟩} (no default)

Contains TEX code which will be installed before the first cell in a row.

/pgfplots/after row={⟨*TEX code*⟩} (no default)

Contains TEX code which will be installed after the last cell in a row (i.e. after \\).

/pgfplots/every even row (style, no value)

A style which is installed for each row with even row index. The first row is supposed to be a "head" row and does not count. Indexing starts with 0.

```
\pgfplotstableset{
    columns={dof,error1,{grad(log(dof),log(error1))}},
    columns/error1/.style={
        column name=$L_2$,
        sci,sci zerofill,sci subscript,
        precision=3},
    columns/dof/.style={
        int detect,
        column name=\textsc{Dof}},
    columns/{grad(log(dof),log(error1))}/.style={
        column name=slopes $L_2$,
        fixed,fixed zerofill,
        precision=1}}
```

Dof	L_2	slopes L_2
9	2.500_{-1}	0.0
25	6.250_{-2}	−1.4
81	1.563_{-2}	−1.2
289	3.906_{-3}	−1.1
1,089	9.766_{-4}	−1.0
4,225	2.441_{-4}	−1.0
16,641	6.104_{-5}	−1.0
66,049	1.526_{-5}	−1.0
263,169	3.815_{-6}	−1.0
1,050,625	9.537_{-7}	−1.0

```
% \usepackage{booktabs}
\pgfplotstabletypesetfile[
    every head row/.style={
        before row=\toprule,after row=\midrule},
    every last row/.style={
        after row=\bottomrule},
]
    {pgfplotstable.example1.dat}
```

Dof	L_2	slopes L_2
9	2.500_{-1}	0.0
25	6.250_{-2}	−1.4
81	1.563_{-2}	−1.2
289	3.906_{-3}	−1.1
1,089	9.766_{-4}	−1.0
4,225	2.441_{-4}	−1.0
16,641	6.104_{-5}	−1.0
66,049	1.526_{-5}	−1.0
263,169	3.815_{-6}	−1.0
1,050,625	9.537_{-7}	−1.0

```
% \usepackage{booktabs,colortbl}
\pgfplotstabletypesetfile[
    every even row/.style={
        before row={\rowcolor[gray]{0.9}}},
    every head row/.style={
        before row=\toprule,after row=\midrule},
    every last row/.style={
        after row=\bottomrule},
]
    {pgfplotstable.example1.dat}
```

/pgfplots/every odd row (style, no value)

A style which is installed for each row with odd row index. The first row is supposed to be a "head" row and does not count. Indexing starts with 0.

/pgfplots/every head row (style, no value)

A style which is installed for each first row in the tabular. This can be used to adjust options for column names or to add extra lines/colours.

/pgfplots/every first row (style, no value)

A style which is installed for each first *data* row, i.e. after the head row.

/pgfplots/every last row (style, no value)

A style which is installed for each last *data* row.

/pgfplots/table/row predicate/.code={⟨...⟩}

A boolean predicate which allows to select particular rows of the input table. The argument `#1` contains the current row's index (starting with 0, not counting comment lines or column names).

The return value is assigned to the TEX-if `\ifpgfplotstableuserow`. If the boolean is not changed, the return value is true.

level	dof	error1	error2	info	grad(log(dof),log(error1))	quot(error1)
1	9	0.25	0.76	48	0	0
2	25	$6.25 \cdot 10^{-2}$	0.5	25	−1.36	4
3	81	$1.56 \cdot 10^{-2}$	0.29	41	−1.18	4
4	289	$3.91 \cdot 10^{-3}$	0.14	8	−1.09	4
5	1,089	$9.77 \cdot 10^{-4}$	$4.42 \cdot 10^{-2}$	22	−1.05	4
9	$2.63 \cdot 10^{5}$	$3.81 \cdot 10^{-6}$	$1.95 \cdot 10^{-3}$	33	−1	4
10	$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$	$9.77 \cdot 10^{-4}$	2	−1	4

```
% \usepackage{booktabs}
\pgfplotstabletypesetfile[
    every head row/.style={
        before row=\toprule,after row=\midrule},
    every last row/.style={
        after row=\bottomrule},
    row predicate/.code={%
        \ifnum#1>4\relax
            \ifnum#1<8\relax
                \pgfplotstableuserowfalse
            \fi
        \fi}
]
    {pgfplotstable.example1.dat}
```

Please note that `row predicate` is applied *before* any other option which affects row appearance. For example, the even/odd styles refer to those rows which are selected.

Furthermore, `row predicate` applies only to the typeset routines, not the read methods. If you want to plot only selected table entries with `\addplot table`, use the PGFPLOTS coordinate filter options.

/pgfplots/table/skip rows between index={⟨*begin*⟩}{⟨*end*⟩} (style, no default)

A style which appends an `row predicate` which discards selected rows. The selection is done by index where indexing starts with 0. Every row with index $\langle begin \rangle \leq i < \langle end \rangle$ will be skipped.

level	dof	error1	error2	info	grad(log(dof),log(error1))	quot(error1)
1	9	0.25	0.76	48	0	0
2	25	$6.25 \cdot 10^{-2}$	0.5	25	−1.36	4
5	1,089	$9.77 \cdot 10^{-4}$	$4.42 \cdot 10^{-2}$	22	−1.05	4
6	4,225	$2.44 \cdot 10^{-4}$	$1.7 \cdot 10^{-2}$	46	−1.02	4
7	16,641	$6.1 \cdot 10^{-5}$	$8.2 \cdot 10^{-3}$	40	−1.01	4
10	$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$	$9.77 \cdot 10^{-4}$	2	−1	4

```
% \usepackage{booktabs}
\pgfplotstabletypesetfile[
    every head row/.style={
        before row=\toprule,after row=\midrule},
    every last row/.style={
        after row=\bottomrule},
    skip rows between index={2}{4},
    skip rows between index={7}{9}
]
    {pgfplotstable.example1.dat}
```

5 Determining cell contents

/pgfplots/table/assign cell content/.code={⟨...⟩}

Allows to redefine the algorithm which assigns cell contents. The argument #1 is the (unformatted) contents of the input table.

The resulting output needs to be written to /pgfplots/table/@cell content.

/pgfplots/table/assign cell content as number (no value)

This here is the default implementation of assign cell contents.

/pgfplots/table/string type (style, no value)

A style which redefines assign cell contents to allow (a) text column(s).

6 Customizing and getting the tabular code

/pgfplots/table/every table (style, no value)

A style which is installed at the beginning of every \pgfplotstabletypeset command.

/pgfplots/table/font={⟨*font name*⟩} (no default, initially empty)

Assigns a font used for the complete table.

/pgfplots/table/begin table={⟨*code*⟩} (no default, initially \begin{tabular})

Contains {⟨*code*⟩} which is generated as table start.

/pgfplots/table/end table={⟨*code*⟩} (no default, initially \end{tabular})

Contains {⟨*code*⟩} which is generated as table end.

/pgfplots/table/outfile={⟨*file name*⟩} (no default, initially empty)

Writes the generated tabular code into {⟨*file name*⟩}. It can then be used with \input{⟨*file name*⟩}, PgfplotsTable is no longer required since it contains a completely normal tabular.

dof	error1
9	0.25
25	$6.25 \cdot 10^{-2}$
81	$1.56 \cdot 10^{-2}$
289	$3.91 \cdot 10^{-3}$
1,089	$9.77 \cdot 10^{-4}$
4,225	$2.44 \cdot 10^{-4}$
16,641	$6.1 \cdot 10^{-5}$
66,049	$1.53 \cdot 10^{-5}$
$2.63 \cdot 10^{5}$	$3.81 \cdot 10^{-6}$
$1.05 \cdot 10^{6}$	$9.54 \cdot 10^{-7}$

```
\pgfplotstabletypesetfile[
    columns={dof,error1},
    outfile=pgfplotstable.example1.out.tex]
    {pgfplotstable.example1.dat}
```

and pgfplotstable.example1.out.tex contains

```
\begin {tabular}{cc}dof&error1\\
\pgfutilensuremath {9}&\pgfutilensuremath {0.25}\\
\pgfutilensuremath {25}&\pgfutilensuremath {6.25\cdot 10^{-2}}\\
\pgfutilensuremath {81}&\pgfutilensuremath {1.56\cdot 10^{-2}}\\
\pgfutilensuremath {289}&\pgfutilensuremath {3.91\cdot 10^{-3}}\\
\pgfutilensuremath {1{,}089}&\pgfutilensuremath {9.77\cdot 10^{-4}}\\
\pgfutilensuremath {4{,}225}&\pgfutilensuremath {2.44\cdot 10^{-4}}\\
\pgfutilensuremath {16{,}641}&\pgfutilensuremath {6.1\cdot 10^{-5}}\\
\pgfutilensuremath {66{,}049}&\pgfutilensuremath {1.53\cdot 10^{-5}}\\
\pgfutilensuremath {2.63\cdot 10^{5}}&\pgfutilensuremath {3.81\cdot 10^{-6}}\\
\pgfutilensuremath {1.05\cdot 10^{6}}&\pgfutilensuremath {9.54\cdot 10^{-7}}\\
\end {tabular}
```

The command \pgfutilensuremath checks whether math mode is active and switches to math mode if necessary[1].

/pgfplots/table/debug={⟨*boolean*⟩} (no default, initially false)

If enabled, will write every final tabular code to you log file.

7 Defining column types for tabular

Besides input of text files, it is sometimes desireable to define column types for existing tabular environments.

\newcolumntype{⟨*letter*⟩}[⟨*number of arguments*⟩]>{⟨*before column*⟩}⟨*column type*⟩<{⟨*after column*⟩}

The command \newcolumntype is part of the array package and it defines a new column type {⟨*letter*⟩} for use in LaTeX tabular environments.

```
\usepackage{array}
```

-a+	b
-c+	d

```
\newcolumntype{d}{>{-}c<{+}}
\begin{tabular}{dl}
a & b \\
c & d \\
\end{tabular}
```

Now, the environment pgfplotstablecoltype can be used in {⟨*before column*⟩} and {⟨*after column*⟩} to define numerical columns:

9	2.50_{-1}
25	6.25_{-2}
81	1.56_{-2}
289	3.91_{-3}
1,089	9.77_{-4}
4,225	2.44_{-4}
16,641	6.10_{-5}
66,049	1.53_{-5}
263,169	3.81_{-6}
1,050,625	9.54_{-7}

```
\newcolumntype{L}[1]
    {>{\begin{pgfplotstablecoltype}[#1]}r<{\end{pgfplotstablecoltype}}}

\begin{tabular}{L{int detect}L{sci,sci subscript,sci zerofill}}
9       & 2.50000000e-01\\
25      & 6.25000000e-02\\
81      & 1.56250000e-02\\
289     & 3.90625000e-03\\
1089    & 9.76562500e-04\\
4225    & 2.44140625e-04\\
16641   & 6.10351562e-05\\
66049   & 1.52587891e-05\\
263169  & 3.81469727e-06\\
1050625& 9.53674316e-07\\
\end{tabular}
```

The environment pgfplotstablecoltype accepts an optional argument which may contain any number formatting options. It is an error if numerical columns contain non-numerical data, so it may be necessary to use \multicolumn for column names.

Dof	Error
9	2.50_{-1}
25	6.25_{-2}
81	1.56_{-2}
289	3.91_{-3}
1,089	9.77_{-4}
4,225	2.44_{-4}
16,641	6.10_{-5}
66,049	1.53_{-5}
263,169	3.81_{-6}
1,050,625	9.54_{-7}

```
\newcolumntype{L}[1]
    {>{\begin{pgfplotstablecoltype}[#1]}r<{\end{pgfplotstablecoltype}}}

\begin{tabular}{L{int detect}L{sci,sci subscript,sci zerofill}}
\multicolumn{1}{r}{Dof} & \multicolumn{1}{r}{Error}\\
9       & 2.50000000e-01\\
25      & 6.25000000e-02\\
81      & 1.56250000e-02\\
289     & 3.90625000e-03\\
1089    & 9.76562500e-04\\
4225    & 2.44140625e-04\\
16641   & 6.10351562e-05\\
66049   & 1.52587891e-05\\
263169  & 3.81469727e-06\\
1050625& 9.53674316e-07\\
\end{tabular}
```

[1] Please note that \pgfutilensuremath needs to be replaced by **\ensuremath** if you want to use the output file independent of PGF. That can be done by **\let**\pgfutilensuremath=**\ensuremath** which enables the LaTeX-command **\ensuremath**.

8 Number formatting options

The following extract of [1] explains how to configure number formats.

`\pgfmathprintnumber{`⟨*x*⟩`}`

Generates pretty-printed output for the (real) number `{`⟨*x*⟩`}`. The input number `{`⟨*x*⟩`}` is parsed using `\pgfmathfloatparsenumber` which allows arbitrary precision.

Numbers are typeset in math mode using the current set of number printing options, see below. Optional arguments can also be provided using `\pgfmathprintnumber[`⟨*options*⟩`]{`⟨*x*⟩`}`.

`\pgfmathprintnumberto{`⟨*x*⟩`}{`⟨*\macro*⟩`}`

Returns the resulting number into `{`⟨*\macro*⟩`}` instead of typesetting it directly.

`/pgf/number format/fixed` (no value)

Configures `\pgfmathprintnumber` to round the number to a fixed number of digits after the period, discarding any trailing zeros.

4.57 0 0.1 24,415.98 123,456.12

```
\pgfkeys{/pgf/number format/.cd,fixed,precision=2}
\pgfmathprintnumber{4.568}\hspace{1em}
\pgfmathprintnumber{5e-04}\hspace{1em}
\pgfmathprintnumber{0.1}\hspace{1em}
\pgfmathprintnumber{24415.98123}\hspace{1em}
\pgfmathprintnumber{123456.12345}
```

See section 8.1 for how to change the appearance.

`/pgf/number format/fixed zerofill=`{⟨*boolean*⟩} (default `true`)

Enables or disables zero filling for any number drawn in fixed point format.

4.57 0.00 0.10 24,415.98 123,456.12

```
\pgfkeys{/pgf/number format/.cd,fixed,fixed zerofill,precision=2}
\pgfmathprintnumber{4.568}\hspace{1em}
\pgfmathprintnumber{5e-04}\hspace{1em}
\pgfmathprintnumber{0.1}\hspace{1em}
\pgfmathprintnumber{24415.98123}\hspace{1em}
\pgfmathprintnumber{123456.12345}
```

This key affects numbers drawn with `fixed` or `std` styles (the latter only if no scientific format is choosen).

4.57 $5 \cdot 10^{-5}$ 1.00 $1.23 \cdot 10^{5}$

```
\pgfkeys{/pgf/number format/.cd,std,fixed zerofill,precision=2}
\pgfmathprintnumber{4.568}\hspace{1em}
\pgfmathprintnumber{5e-05}\hspace{1em}
\pgfmathprintnumber{1}\hspace{1em}
\pgfmathprintnumber{123456.12345}
```

See section 8.1 for how to change the appearance.

`/pgf/number format/sci` (no value)

Configures `\pgfmathprintnumber` to display numbers in scientific format, that means sign, mantisse and exponent (basis 10). The mantisse is rounded to the desired precision.

$4.57 \cdot 10^{0}$ $5 \cdot 10^{-4}$ $1 \cdot 10^{-1}$ $2.44 \cdot 10^{4}$ $1.23 \cdot 10^{5}$

```
\pgfkeys{/pgf/number format/.cd,sci,precision=2}
\pgfmathprintnumber{4.568}\hspace{1em}
\pgfmathprintnumber{5e-04}\hspace{1em}
\pgfmathprintnumber{0.1}\hspace{1em}
\pgfmathprintnumber{24415.98123}\hspace{1em}
\pgfmathprintnumber{123456.12345}
```

See section 8.1 for how to change the exponential display style.

/pgf/number format/sci zerofill={⟨*boolean*⟩} (default `true`)

Enables or disables zero filling for any number drawn in scientific format.

$4.57 \cdot 10^{0}$ $5.00 \cdot 10^{-4}$ $1.00 \cdot 10^{-1}$ $2.44 \cdot 10^{4}$ $1.23 \cdot 10^{5}$

```
\pgfkeys{/pgf/number format/.cd,sci,sci zerofill,precision=2}
\pgfmathprintnumber{4.568}\hspace{1em}
\pgfmathprintnumber{5e-04}\hspace{1em}
\pgfmathprintnumber{0.1}\hspace{1em}
\pgfmathprintnumber{24415.98123}\hspace{1em}
\pgfmathprintnumber{123456.12345}
```

As with `fixed zerofill`, this option does only affect numbers drawn in `sci` format (or `std` if the scientific format is chosen).

See section 8.1 for how to change the exponential display style.

/pgf/number format/zerofill={⟨*boolean*⟩} (style, default `true`)

Sets both, `fixed zerofill` and `sci zerofill` at once.

/pgf/number format/std (no value)

Configures `\pgfmathprintnumber` to a standard algorithm. It chooses either `fixed` or `sci`, depending on the order of magnitude. Let $n = s \cdot m \cdot 10^{e}$ be the input number and p the current precision. If $-p/2 \leq e \leq 4$, the number is displayed using the fixed format. Otherwise, it is displayed using the scientific format.

4.57 $5 \cdot 10^{-4}$ 0.1 24,415.98 $1.23 \cdot 10^{5}$

```
\pgfkeys{/pgf/number format/.cd,std,precision=2}
\pgfmathprintnumber{4.568}\hspace{1em}
\pgfmathprintnumber{5e-04}\hspace{1em}
\pgfmathprintnumber{0.1}\hspace{1em}
\pgfmathprintnumber{24415.98123}\hspace{1em}
\pgfmathprintnumber{123456.12345}
```

/pgf/number format/int detect (no value)

Configures `\pgfmathprintnumber` to detect integers automatically. If the input number is an integer, no period is displayed at all. If not, the scientific format is chosen.

15 20 $2.04 \cdot 10^{1}$ $1 \cdot 10^{-2}$ 0

```
\pgfkeys{/pgf/number format/.cd,int detect,precision=2}
\pgfmathprintnumber{15}\hspace{1em}
\pgfmathprintnumber{20}\hspace{1em}
\pgfmathprintnumber{20.4}\hspace{1em}
\pgfmathprintnumber{0.01}\hspace{1em}
\pgfmathprintnumber{0}
```

/pgf/number format/int trunc (no value)

Truncates every number to integers (discards any digit after the period).

4 0 0 24,415 123,456

```
\pgfkeys{/pgf/number format/.cd,int trunc}
\pgfmathprintnumber{4.568}\hspace{1em}
\pgfmathprintnumber{5e-04}\hspace{1em}
\pgfmathprintnumber{0.1}\hspace{1em}
\pgfmathprintnumber{24415.98123}\hspace{1em}
\pgfmathprintnumber{123456.12345}
```

/pgf/number format/precision={⟨*number*⟩} (no default)

Sets the desired rounding precision for any display operation. For scientific format, this affects the mantisse.

8.1 Changing display styles

You can change the way how numbers are displayed. For example, if you use the '`fixed`' style, the input number is rounded to the desired precision and the current fixed point display style is used to typeset the number. The same is applied to any other format: first, rounding routines are used to get the correct digits, afterwards a display style generates proper TEX-code.

`/pgf/number format/set decimal separator`={⟨*text*⟩} (no default)

Assigns {⟨*text*⟩} as decimal separator for any fixed point numbers (including the mantisse in sci format).

`/pgf/number format/dec sep`={⟨*text*⟩} (no default)

Just another name for `set decimal separator`.

`/pgf/number format/set thousands separator`={⟨*text*⟩} (no default)

Assigns {⟨*text*⟩} as thousands separator for any fixed point numbers (including the mantisse in sci format).

1234.56

```
\pgfkeys{/pgf/number format/.cd,
    fixed,
    fixed zerofill,
    precision=2,
    set thousands separator={}}
\pgfmathprintnumber{1234.56}
```

1234567890.00

```
\pgfkeys{/pgf/number format/.cd,
    fixed,
    fixed zerofill,
    precision=2,
    set thousands separator={}}
\pgfmathprintnumber{1234567890}
```

1.234.567.890.00

```
\pgfkeys{/pgf/number format/.cd,
    fixed,
    fixed zerofill,
    precision=2,
    set thousands separator={.}}
\pgfmathprintnumber{1234567890}
```

1, 234, 567, 890.00

```
\pgfkeys{/pgf/number format/.cd,
    fixed,
    fixed zerofill,
    precision=2,
    set thousands separator={,}}
\pgfmathprintnumber{1234567890}
```

1,234,567,890.00

```
\pgfkeys{/pgf/number format/.cd,
    fixed,
    fixed zerofill,
    precision=2,
    set thousands separator={{{{,}}}}}
\pgfmathprintnumber{1234567890}
```

The last example employs commas and disables the default comma-spacing.

`/pgf/number format/1000 sep`={⟨*text*⟩} (no default)

Just another name for `set thousands separator`.

`/pgf/number format/use period` (no value)

A predefined style which installs periods '`.`' as decimal separators and commas '`,`' as thousands separators. This style is the default.

12.35

```
\pgfkeys{/pgf/number format/.cd,fixed,precision=2,use period}
\pgfmathprintnumber{12.3456}
```

1,234.56

```
\pgfkeys{/pgf/number format/.cd,fixed,precision=2,use period}
\pgfmathprintnumber{1234.56}
```

/pgf/number format/use comma (no value)

A predefined style which installs commas ‘,’ as decimal separators and periods ‘.’ as thousands separators.

12,35
```
\pgfkeys{/pgf/number format/.cd,fixed,precision=2,use comma}
\pgfmathprintnumber{12.3456}
```

1.234,56
```
\pgfkeys{/pgf/number format/.cd,fixed,precision=2,use comma}
\pgfmathprintnumber{1234.56}
```

/pgf/number format/skip 0.={⟨*boolean*⟩} (no default)

Configures whether numbers like 0.1 shall be typeset as .1 or not.

.56
```
\pgfkeys{/pgf/number format/.cd,
    fixed,
    fixed zerofill,precision=2,
    skip 0.}
\pgfmathprintnumber{0.56}
```

0.56
```
\pgfkeys{/pgf/number format/.cd,
    fixed,
    fixed zerofill,precision=2,
    skip 0.=false}
\pgfmathprintnumber{0.56}
```

/pgf/number format/sci 10e (no value)

Uses $m \cdot 10^e$ for any number displayed in scientific format.

$1.23 \cdot 10^{1}$
```
\pgfkeys{/pgf/number format/.cd,sci,sci 10e}
\pgfmathprintnumber{12.345}
```

/pgf/number format/sci 10^e (no value)

The same as ‘sci 10e’.

/pgf/number format/sci e (no value)

Uses the ‘1e+0’ format which is generated by common scientific tools for any number displayed in scientific format.

1.23e+1
```
\pgfkeys{/pgf/number format/.cd,sci,sci e}
\pgfmathprintnumber{12.345}
```

/pgf/number format/sci E (no value)

The same with an uppercase ‘E’.

1.23E+1
```
\pgfkeys{/pgf/number format/.cd,sci,sci E}
\pgfmathprintnumber{12.345}
```

/pgf/number format/sci subscript (no value)

Typesets the exponent as subscript for any number displayed in scientific format. This style requires very few space.

1.23_{1}
```
\pgfkeys{/pgf/number format/.cd,sci,sci subscript}
\pgfmathprintnumber{12.345}
```

/pgf/number format/assume math mode={⟨*boolean*⟩} (default true)

Set this to true if you don’t want any checks for math mode.

The initial setting installs a \pgfutilensuremath around each final number to change to math mode if necessary. Use assume math mode=true if you know that math mode is active and you don’t want \pgfutilensuremath.

9 Plain TEX and ConTEXt support

The table code generator is initialised to produce LATEX `tabular` environments. However, it only relies on '`&`' being the column separator and '`\\`' the row terminator. The `column type` feature is more or less specific to `tabular`, but you can disable it completely. Replace `begin table` and `end table` with appropriate TEX- or ConTEXt commands to change it. If you have useful default styles (or bug reports), let me know.

Index

References

[1] T. Tantau. Ti*k*Z and PGF manual. `http://sourceforge.net/projects/pgf`. $v. \geq 2.00$.