latexTableClass for FORTRAN Documentation

Lukasz A. Drozd *

August 2018

*Standard GNU license applies.

Disclaimer and copyright

This document contains documentation for the latexTableClass.

1 The basics

1.1 What is latexTableClass?

latexTableClass generates latex code with tables directly from FORTRAN code. Table 1 provides an example of a table generated using the class (generated by attached test.f90 file) and a pdflatex compiler.

Table 1: My first table

	Column	Marged Columns			
		Block 1		Block 2	
Characters	Integers	Reals	Doubles	Doubles	Doubles
Block header 1					
row1	2	0.20000	0.30000	0.30000	0.30000
row2	3	0.10000	0.20000	0.20000	0.20000
row3	1	3.00000	1.00000	1.00000	1.00000
Block header 1					
row4	10	1.44300	0.40000	0.40000	0.40000
row5	200	0.54000	0.50000	0.50000	0.50000

Figure 1: Sample table generated by latexTableClass and pdflatex.

1.2 How does latexTableClass work?

The class writes a tex file with custom table generated during run time from the FORTRAN code. When setting pdflatex=.true. it issues system-level command to compile tex file and generate a pdf using pdflatex.

1.3 Is latexTableClass cross-platform compatible?

LatexTableClass works with any FORTRAN compiler/platform up to the point of issuing system level command to compile generated tex files using pdflatex. Unfortunately, issuing system commands is compiler and platform specific. This particular implementation of the class assumes Intel Fortran Compiler (Composer XE 2017 or later) and Windows operating system. pdflatex must be present and accessible from the command line (path to pdf latex must be defined). It will not function on any other platform or even older Intel Fortran compilers. However, even in such a case, the class can be used with setting pdflatex=.false. and after commenting system calls in pdflatex subroutine. If system commands are supported by the compiler but they invoked differently, the class can be modified to obtain full functionality. In such a case modify pdflatex subroutine.

1.4 What do I need to add latexTableClass to my code?

latexTableClass comprises two modules: latexTableColumn and latexTableMakerClass. They can both be found in LatexTableClassClass.f90 file. latexTableMakerClass must be added to the program prior to use with the usual use statement. latexTableColumn is added automatically. The following illustrates how to add the class to main program test:

```
program test

use latexTableMakerClass !declare the class

implicit none

!create table object
type(latexTableMaker)   :: Table
!create column data wrapper
type(latexTableColumn) :: tableColumns(6)

end program
```

The code adds both modules and defines two objects that it operates on: the table wrapper class and column data wrapper object (tableColumns). Table class defines table environment and data wrapper stores user provided data that fills the table.

To use latexTableClass, the file containing its code added to the FORTRAN project. The downloaded package contains a test program (test.f90) that demonstrates the basic functionality of the class. This documentation includes several complete examples. Output files that latexTableClass produces are stored in the current directory.

(!) IMPORTANT: For full functionality pdflatex must be available from the command line.

2 latexTableMaker table wrapper

The *latexTableMaker* wrapper class must be defined prior to writing the table in the main program. The wrapper contains formatting information passed to pdflatex.

```
type(latexTableMaker)   :: Table !create table object
```

The latexTableMaker wrapper class comprises of attributes listed below. These attributes are private and cannot be accessed from the main program other than by using an appropriate method from the list below.

```
type latexTableMaker

character(len=256), private   :: entryfmt     = 'g12.5'
character(len=1)   , private   :: separator    = 'l'
type(columnHeaderLayer), dimension(:), allocatable :: columnHeaderLayers

integer, private :: numColumnHeaderLayers=0
integer, private :: numRowHeaders=0
type(rowHeader), dimension(:), allocatable :: rowHeaders

contains

procedure :: addColumnHeaderLayer
procedure :: addRowBlockHeader
procedure :: setTableEntryFormat
procedure :: tableentryformat
procedure :: clearColumnHeaderLayers
procedure :: clearRowBlockHeaders
```

```
procedure, private   :: writeDPTableAsLatex
procedure, private   :: writeRealTableAsLatex
procedure, private   :: writeIntegerTableAsLatex
procedure, private   :: writeTableColumnArrayAsLatex
generic              :: writeTableAsLatex &
                     => writeDPTableAsLatex,  &
                         writeIntegerTableAsLatex, &
                         writeTableColumnArrayAsLatex, &
                         writeRealTableAsLatex

end type latexTableMaker
```

The following is an example that defines a new table object and prints it as pdf. The first lines of code (1-3) set the title of the table and specify axis formatting and labeling. The following lines of code (lines 5-13) add a layered column header in addition to column header by default generated from column titles. The last lines add a custom row block headers. See output in Table 1.

(!) Note that all attributes are automatically initiated at compile time to some default values. Some attributes are optional and initial values are assumed when their explicit specification is omitted in the code.

```
!add layered column headers
call Table % addColumnHeaderLayer(headers=&
                    (/'Column', 'Marged Columns'/), &
                      span=(/1,4/), &
                      level=2)
call Table % addColumnHeaderLayer(headers=&
                   (/'', 'Block 1', 'Block 2' /), &
                      span=(/1,2,2/), &
                      level=1)


!add row block labels:
call Table % addRowBlockHeader(header='Block header 1',row=1)
call Table % addRowBlockHeader(header='Block header 1',row=3)

!write  table to files
call Table % writeTableAsLatex(tableColumnsArray = tableColumns,&
                               title             = "MyTable", &
```

```
                                    label               = "Tab1" )
```

The following is the complete complete list of attributes assigned by the *addColumnHeaderLayer* method :

1. *headers* describes additional column headers above automatically generated base header from column titles. It is an array of strings that must match the size of the table given span specification (see below). From main program, use syntax:

```
call Table % addColumnHeaderLayer(headers=&
                  (/'', 'Block 1', 'Block 2' /), &
                  span=(/1,2,2/), &
                  level=1)
```

2. *span* is an attribute that specifies the span of each a column header. If span>1 it feeds into multicolumn command that merges several columns and adds a label above them. See example above.

3. *level* is an integer that specifies the level of each column header layer. The lower the level specification the lower the given layer appears on the final table. For example, the code above adds two headers on top of each other.

```
!add layered column headers
call Table % addColumnHeaderLayer(headers=&
                    (/'Column', 'Marged Columns'/), &
                      span=(/1,4/), &
                      level=2)
call Table % addColumnHeaderLayer(headers=&
                  (/'', 'Block 1', 'Block 2' /), &
                      span=(/1,2,2/), &
                      level=1)
```

The following is the complete complete list of attributes assigned by the *addRowBlockHeader* method:

1. *header* describes additional row block header that is automatically added to each row of the table.

2. *row* is the row number after which the header appears.

The following is the complete complete list of attributes assigned by the *writeTableAsLatex* method that generates the table:

1. *tableColumnsArray* is the data wrapper with the content of table columns. (Including row labels and base column header generated from the titles of each column in the array.)

2. *title* specifies the title of the table.

3. *label* is the label of the table and file identifier key. Avoid using !@ or any other symbols that cannot go to a file name.

3 latexTableColumn data wrapper

latexTableColumn data wrapper must be defined by the following declaration made in the main program prior to writing the table:

```
type(latexTableColumn) :: tableColumns(6)
```

The wrapper contains data for the table and headers for the rows and columns. It can be defined as an array (as above) or a scalar.

The attributes of latexTableColumn are public and can be directly accessed from the main program. To ensure a consistent declaration use the provided constructors to upload data.

```
type latexTableColumn
character(len=256)     :: title        = ''
character(len=256)     :: columnformat = '(a)'
integer                :: datatype     &
                        = column_undefined_type
real*8, allocatable    :: doublevalues(:)
character(len=256)     , allocatable :: charactervalues(:)
integer                , allocatable :: integervalues(:)
end type latexTableColumn
```

The attributes specify the title of a given column (title), data format (columnformat), and an array of values consistent with datatype specification. See below how to construct data wrapper from the main program using its dedicated set of methods.

4 Uploading data for plotting

Here is an example that uploads data using *constructLatexTableColumn*. The first line of the constructor defines the wrapper that stores data, the second line specifies the title of the column, and the third line uploads data using a FORTRAN array (e.g. vector(1:n) of type integer, single, double or character). The following example illustrates the use of the constructor to fill the previously define tableColumns(1:6) wrapper:

```
!upload data to column wrappers
call constructLatexTableColumn(tableColumn = tableColumns(1),&
                               title       = "Characters",&
                               columnData  = headers,&
                               columnFormat = "(A)")
call constructLatexTableColumn(tableColumn  = tableColumns(2),&
                               title       = "Integers",&
                               columnData  = intData,&
                               columnFormat = "(i6)")
call constructLatexTableColumn(tableColumn  = tableColumns(3),&
                               title       = "Reals",&
                               columnData  = real4Data,&
                               columnFormat = "(f12.5)")
call constructLatexTableColumn(tableColumn  = tableColumns(4),&
                               title       = "Doubles",&
                               columnData  = real8Data,&
                               columnFormat = "(f12.5)")
call constructLatexTableColumn(tableColumn  = tableColumns(5),&
                               title       = "Doubles",&
                               columnData  = real8Data,&
                               columnFormat = "(f12.5)")
call constructLatexTableColumn(tableColumn  = tableColumns(6),&
                               title       = "Doubles",&
                               columnData  = real8Data,&
                               columnFormat = "(f12.5)")
```

The following is the complete list of attributes that can be specified within the constructor:

1. *tableColumns* is the data wrapper that is being constructed by the call.

2. *title* is a string label.

3. *columnData* is a FORTRAN array with a vector of data.

4. *columnFormat* is FORTRAN formatting that will apply to final output displayed in table. It can specify floating point precision, integer, or character data.

(!) IMPORTANT: Note that derived type allows series to be of different length, even though they are pooled together into a single object.

5 Writing tables to files

Once table object and data wrapper have been defined, writeTableAsLatex can be invoked to generate tex code and compile the table to a pdf file. In this case, the syntax would be as follows :

```
!write table to files
call Table % writeTableAsLatex(tableColumnsArray = tableColumns,&
                               title             = "MyTable", &
                               label             = "Tab1" )
```

tableColumnsArray feeds data wrapper. *title* specifies table caption. *label* is table's label and output file identifier.
(!) IMPORTANT: Do not use characters that are illegal for file name specification on a given system.

5.1 Example

Here is a complete code that generates a simple table. The output of this code is Table 1.

```
program test

use latexTableMakerClass !declare the class

implicit none
!create table object
type(latexTableMaker)   :: Table

!create column wrappers with data
type(latexTableColumn) :: tableColumns(6)
```

```
real*8                  :: real8Data(5) = [0.3,0.2,1.0,0.4,0.5]
real*4                  :: real4Data(5) = [0.2,0.1,3.0,1.443,0.54]
integer                 :: intData(5)   = [2,3,1,10,200]
character(LEN=5)        :: headers(5)   = &
                           ["row1","row2","row3","row4 ","row5"]
character(LEN=23)       :: title        = "My first table"
character(LEN=10)       :: label        = "myTable"

integer :: ios

!upload data to column wrappers
call constructLatexTableColumn(tableColumn = tableColumns(1),&
                                        title        = "Characters",&
                                        columnData   = headers,&
                                        columnFormat = "(A)")

call constructLatexTableColumn(tableColumn  = tableColumns(2),&
                                        title        = "Integers",&
                                        columnData   = intData,&
                                        columnFormat = "(i6)")

call constructLatexTableColumn(tableColumn  = tableColumns(3),&
                                        title        = "Reals",&
                                        columnData   = real4Data,&
                                        columnFormat = "(f12.5)")

call constructLatexTableColumn(tableColumn  = tableColumns(4),&
                                        title        = "Doubles",&
                                        columnData   = real8Data,&
                                        columnFormat = "(f12.5)")

call constructLatexTableColumn(tableColumn  = tableColumns(5),&
                                        title        = "Doubles",&
                                        columnData   = real8Data,&
                                        columnFormat = "(f12.5)")

call constructLatexTableColumn(tableColumn  = tableColumns(6),&
                                        title        = "Doubles",&
                                        columnData   = real8Data,&
```

```
                                        columnFormat = "(f12.5)")

!add layered column headers
call Table % addColumnHeaderLayer(headers=&
                        (/'Column', 'Marged Columns' /), &
                        span=(/1,4/), &
                        level=2)
call Table % addColumnHeaderLayer(headers=&
                (/'', 'Block 1', 'Block 2' /), &
                span=(/1,2,2/), &
                level=1)

!add row block labels:
call Table % addRowBlockHeader(header='Block header 1', &
                                    row=1)
call Table % addRowBlockHeader(header='Block header 1', &
                                     row=3)

!write table to files
call Table % writeTableAsLatex(tableColumnsArray = tableColumns,&
                                        title           = title , &
                                        label           = label )
end program test
```

6 Compile time settings

Additional compile time settings of the class are as follows:

```
!user defined module settings
logical :: pdflatex=.true.
!in pdflatex mode the class compiles tables to pdf files
logical :: stealth=.false.
!in stealth mode the class erases files with
! tex codes and only leaves final output
```

7 Issues and quick fixes

1. Previously generated files on same platforms do not allow pdflatex to generate new pdf files with the same name. Use call eraseGnuFile() to clean the current folder from all files that start with gnu- and have .txt, tex or pdf extension.