# luaaddplot

Reinhard Kotucha

Version 1.0a
September 22, 2024

**Abstract**

The pgfplots package supports plotting data from files. If these files are generated by measuring devices they almost always have to be pre-processed.

This packge provides a macro `\luaaddplot` which extends the functionality of `\addplot` by an optional argument containing a $\lambda$- expression which is evaluated for each line of the data file.

The package requires LuaTeX or LuaLaTeX.

## Contents

**1 Introduction** **2**
- 1.1 Usage . . . 2
- 1.2 Reading the data file . . . 2
- 1.3 $\lambda$-Expressions . . . 2

**2 Examples** **3**
- 2.1 Scaling . . . 3
- 2.2 Processing . . . 3

**3 Implementation** **4**
- 3.1 luaaddplot.tex . . . 4
- 3.2 luaaddplot.sty . . . 4
- 3.3 luaaddplot.lua . . . 4

# 1 Introduction

When plotting files generated by measuring devices the data files often have to be pre-processed before they can be passed to pgfplots. In most cases a header describing the device configuration must be removed, column delimiters have to be changed, and units have to be scaled (Hz to MHz, V to μV, etc...). In some cases more complex operations are required.

## 1.1 Usage

`\luaaddplot[`⟨*options*⟩`] file "`⟨*filename*⟩`"[, `⟨*λ-expression*⟩`];`

Everything between `\luaaddplot` and the keyword `file` is passed to `\addplot`, everything between `file` and the semicolon is processed by Lua and valid Lua syntax is required here. You can consider code between `file` and the terminating semicolon as arguments of a Lua function with braces omitted.

## 1.2 Reading the data file

Data files generated by measuring devices usually have an ASCII header describing the settings. These lines are often preceded by comment characters but not always. In order to avoid pre-processing with another tool, all lines which do not begin with a number are ignored.

Different devices use different column delimiters. Therefore tabs, colons, semicolons, and commas are replaced by a space. With other words, various datafile formats are converted to a to a MATLAB compatible format.

## 1.3 λ-Expressions

A λ-expression is an anonymous function and looks like this:

```
function (a)
  return a[1], a[2]
end
```

The formal parameter `a` (its name can be chosen arbitrarily) is replaced by an ordered list of columns for each line in the data file. The return values are the $x$ and $y$ values to be plotted and are passed to `\addplot`. Always two values must be returned. Please note that the index of the first column is 1 in Lua and not 0, as in many other programming languages.

In the example above the first and second column of the data file will be plotted. This is the default if the λ-expression is omitted.

# 2 Examples

## 2.1 Scaling

The following is an excerpt of a data file generated by a spectrum analyzer.

```
x-Unit;Hz;
y-Unit;dBm;
Preamplifier;OFF;
Transducer;OFF;
Values;501;
0;43.660163879394531;
6000000;-53.616184234619141;
12000000;-60.31707763671875;
18000000;-62.548038482666016;
54000000;-66.722061157226563;
```

Frequencies are in Hz but for the plot we prefer MHz. The solution is

```
\luaaddplot[blue] file "spectrum.data", function (col)
                                          return col[1]/1e6, col[2]
                                        end;
```

## 2.2 Processing

The following is an excerpt of a data file generated by a vector network analyzer.

```
% Date: 2021-02-25 11:36:04
% Data & Calibration Information:
% Trc1:S11(Full One Port)
%freq[Hz] re:Trc1_S11 im:Trc1_S11
6.650000000000000E+008,3.684957681171731E-001,-9.205383204111828E-001,
6.652500000000000E+008,3.717039523633851E-001,-9.215835251874305E-001,
6.655000000000000E+008,3.687706454916999E-001,-9.207422307051146E-001,
6.657500000000000E+008,3.574212650433284E-001,-9.176705980253278E-001,
6.660000000000000E+008,3.453552411257967E-001,-9.254690887689956E-001,
```

The first column denotes the frequency in Hz, the second and third column denote the real and imaginary part of the impedance $s_{11}$ respectively.

Now we want to plot

$$20 \log_{10} \sqrt{(\operatorname{Re} s_{11})^2 + (\operatorname{Im} s_{11})^2}$$

with frequencies in MHz.

The solution is

```
\luaaddplot[green] file "s11.data",
        function (col)
           return col[1]/1e6, 20*math.log10(math.sqrt(col[2]^2 + col[3]^2))
        end;
```

# 3 Implementation

```
⟨tex⟩%% File 'luaaddplot.tex', generated from luaaddplot.dtx'.
⟨sty⟩%% File 'luaaddplot.sty', generated from luaaddplot.dtx'.
⟨lua⟩-- File 'luaaddplot.lua', generated from luaaddplot.dtx'.
⟨lua⟩--[[
%%
%% luaaddplot.dtx
%% Copyright 2022 Reinhard Kotucha <reinhard.kotucha@gmx.net>
%%
%% This work may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either version 1.3
%% of this license or (at your option) any later version.
%% The latest version of this license is in
%%   http://www.latex-project.org/lppl.txt
%% and version 1.3 or later is part of all distributions of LaTeX
%% version 2005/12/01 or later.
%%
%% This work has the LPPL maintenance status 'maintained'.
%%
%% The Current Maintainer of this work is Reinhard Kotucha.
%%
%% This work consists of the files luaaddplot.dtx and luaaddplot.ins
%% and the derived files luaaddplot.tex, luaaddplot.sty, and luaaddplot.lua.
⟨lua⟩--]]
```

## 3.1 luaaddplot.tex

```
⟨*tex⟩
\directlua{require('luaaddplot')}
\def\luaaddplot#1file#2;{
        \directlua{luaaddplot.opts('#1')luaaddplot.readfile(#2)}}
⟨/tex⟩
```

## 3.2 luaaddplot.sty

```
⟨*sty⟩ \ProvidesPackage{luaaddplot}[2024/09/22 v1.0a]
\input luaaddplot.tex
⟨/sty⟩
```

## 3.3 luaaddplot.lua

```
⟨*lua⟩

module('luaaddplot', package.seeall)

```

`readfile()` Read and pre-process file `file` and apply $\lambda$-expression. If the second argument is `nil` the first and second column is returned.

```
function readfile (file, lambda)
```

Check whether datafile exists.

```
37  if not lfs.isfile(file) then
38    error('\nERROR: File "'..file..'" not found.')
39  end
40
41  local data = io.open(file)
42
43  for line in data:lines() do
```

Remove all spaces at the beginning of a line.

```
44    line = line:gsub('^%s+', '')
```

Ignore all lines which don't begin with a number. All comments and empty lines in data files are ignored.

```
45    if line:match('^%-?%.?[0-9]') then
```

Replace possible non-space delimiters by spaces. This allows to process various datafile formats the same way as MATLAB files without manual interaction.

```
46      line = line:gsub('[\t:;,]', ' ')
```

The result must be a table with two columns. If no $\lambda$ expression is provided, the first two colums of the data file are returned. With a $\lambda$ expression as second argument arbitrary columns can be selected from a data file and can be pre-processed in any way possible.

```
47      local a, b
48      local cols = line:explode (' +')
```

Convert strings to numbers. Because $\lambda$ expressions can access any column we convert all table entries from strings to numbers, if possible, instead of only the return values. Numbers are returned to TEX as floating point numbers.

```
49      for i, col in ipairs(cols) do
50        cols[i] = tonumber(cols[i])
51      end
52
53      if lambda then
54        a, b = lambda(cols)
55        if a and b then
56          tex.print(string.format('%g %g', a, b))
57        end
58      else
59        tex.print(string.format('%g %g', cols[1], cols[2]))
60      end
61    end
62  end
63  tex.print('};')
64  data:close()
65 end
```

`opts()` Pass optional arguments from `\luaaddplot` to `\addplot`.

```
66
67 function opts (s)
68   tex.print('\\addplot'..s..' table {')
69 end
70
71 % </lua>
```