Applications of Lua for LaTeX Documents

Chetan Shirore*[1] and Ajit Kumar[2]

[1]Department of Mathematics, Institute of Chemical Technology, Mumbai, India

[2]Department of Mathematics, Institute of Chemical Technology, Mumbai, India

Abstract

This article discusses various applications of Lua for LaTeX documents. It mainly focuses on enhancing the graphical aspect of LaTeX using Lua and creating Android applications from LaTeX documents. This is an extension of our work of creating computational packages for LaTeX using Lua. The computational packages we developed are available for LaTeX users on the CTAN repository and bundled with standard TeX distributions. In continuation of this work, the development and deployment of the luaplot package is discussed in this article. It also describes the outline for creating Android applications from LaTeX files using Lua and other resources. The Android applications created from LaTeX files do not need the internet, are static, and do not support calculations. The one purpose of using Lua for LaTeX documents is to reduce the dependence of LaTeX users on external software for computations and graphing. The other purpose is to provide techniques and methods to make mathematical content in LaTeX documents available to Android users. Some of the packages we developed can also be deployed for pedagogical purposes.

1 Background and introduction

Lua [4] is a portable scripting language that we have used for different applications to LaTeX documents. The applications include computational and graphical aspects inside LaTeX documents and the creation of Android applications from LaTeX documents.

The article is organized in different sections. The application of Lua for performing basic mathematical computations inside LaTeX using Lua is described in the second section. The third section provides key features of the plotting tool that we developed and a few illustrations of it. The outline for the creation of Android applications from LaTeX files using Lua and other resources is given in the fourth section. The fifth section provides conclusions and the future plans and prospects of the research work are in the sixth section.

*Corresponding Author, Email: mathsbeauty@gmail.com

2 Computations inside LaTeX documents using Lua

We have developed luamaths [9],luacomplex [6], luaset [14], luagcd [7], luatruthtable [15], luanumint [12], and luamodulartables [10] packages. The `luamaths` package is for basic arithmetic on real and complex numbers and evaluation of standard real-valued functions of a real variable. The `luacomplex` package is for computations on complex numbers. The `luagcd` package can be used to find the gcd of two or more integers and to verify Bezout's identity. The truth tables of boolean variables can be generated with the `luatruthtable` package. The numerical integration of real-valued functions of a real variable can be performed using the `luanumint` package. The `luamodularmaths` package generates modular addition and multiplication tables of positive integers. The linear algebra lualinalg tool [8] is a major highlight of the packages we developed. The standard computations on vectors and matrices can be performed with this tool inside LaTeX documents. The research article "Basic Mathematical Computations inside LaTeX using Lua" [22] describes some of these packages.

3 Plotting Graphs in LaTeX using Lua

We have developed the luaplot package [13] using Lua to plot graphs of real-valued functions of a real variable in LaTeX. It makes use of the Metapost system [17] and luamplib [11] and luacode [5] packages. It provides an easy way for plotting graphs of standard mathematical functions and their finite combinations. There is no particular environment in the package for plotting graphs. It also works inside floating environments of LaTeX like tables and figures. The compilation time to plot several graphs in LaTeX using the luaplot package is significantly less with LuaLaTeX engine.

The core idea is to load mathematical functions inside Lua and determining plot points using different methods available in Lua. After determining plot points in Lua, two different approaches are used:

- parse plot points to the MetaPost system via `luampblib`.
- parse plot points to the `tikz` package.

The MetaPost system is based on the Metafont to produce precise technical illustrations. Donald Knuth designed Metafont for TeX. John Hobby designed the MetaPost system to produce scalable PostScript or scalable vector graphics. The output from MetaPost can be directly included with LaTeX. The first approach thus offers a native way of plotting graphs inside LaTeX using Lua and MetaPost.

Tikz is designed by Till Tantau for producing vector graphics from different expressions. Drawing lines, arrows, paths, geometric shapes, etcetera is possible using Tikz [18]. Tikz commands can be considered as TeX macros, but Tikz itself is a language. LaTeX users widely use Tikz to produce different graphics. The second approach combines Lua and Tikz to plot graphs inside LaTeX.

3.1 Illustrations of the luaplot tool

Listing 1 illustrates the `luaplot` command. It generates graphs shown in Figure 1. For all plotting options, the Metapost package documentation [17] and guide [3] can be referred.

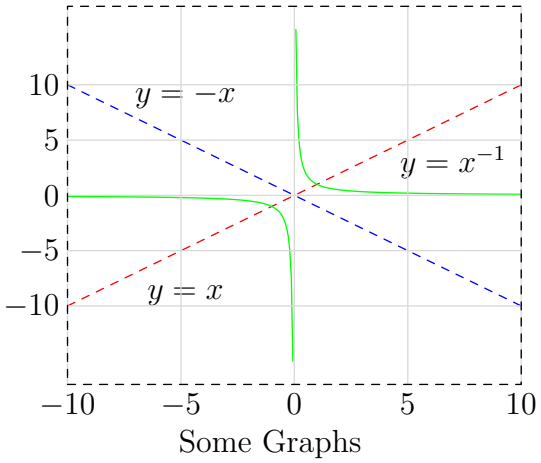

Figure 1: Plotting graphs with the luaplot package

Listing 1: Plotting with the luaplot command

```
\luaplot[
xmin=-10,
xmax=10,
plotpts=300,
hor='6cm',
ver='5cm',
clr={'red; blue; green'},
plotsty={"dashed evenly, dashed evenly"},
plotoptions={
[[
glabel(btex ${y=-x}$ etex,(-4.8,9));
glabel(btex ${y=x}$ etex,(-4.8,-9));
glabel(btex ${y=x^{-1}}$ etex,(7,3));
glabel.bot(btex Some Graphs etex, OUT);
autogrid(grid.bot,) withcolor .85white;
autogrid(grid.lft,) withcolor .85white;
frame.dashed evenly;
]]
}
]
{x,-x,1/x}
```

Listing 2 illustrates plotting graph of a function with the `luatikzpath` command. Multiple graphs can be plotted in a single picture environment.

Listing 2: Plotting with the luatikzpath command

```
\begin{tikzpicture}
\draw[thin,->] (-5,0)--(5,0)node[right]{$x$};
\draw[thin,->] (0,-3)--(0,2.5)node[above]{$y$};
\draw[red] \luatikzpath{sin(x^2)}{-4}{4}{100} node at (1.2,1.3)
```

```
   {$y=\cos(x^2)$};
\draw[blue] \luatikzpath{log(x)}{0.1}{4.9}{100} node at (3.9,1.8)
   {$y=\log(x)$};
\draw[blue] \luatikzpath{log(-x)}{-4.9}{-0.1}{100} node at (-3.5,1.8)
   {$y=\log(-x)$};
\end{tikzpicture}
```

Listing 2 generates graphs shown in Figure 2.

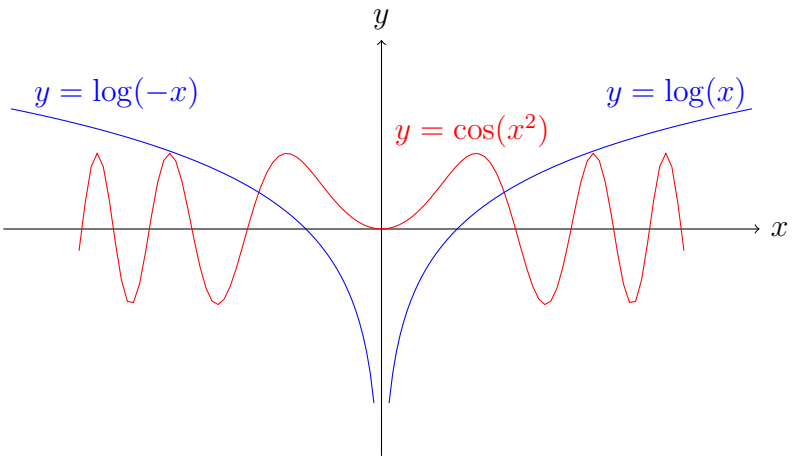

Figure 2: Plotting with the luatikzpath command

3.2 Known issues and limitations

The package does not use any external library supporting arbitrary precision arithmetic. The luaplot package can handle big and small numbers within the range of Lua that it supports. However, the MetaPost system does not support numbers in scientific notation. The coordinates of plot points produced in scientific notation are rounded off to 12 significant decimal places within the package. This may cause slight deviations from actual values. The same issue is not faced while parsing points to tikz as it supports input in scientific notation.

4 Android Applications from LaTeX files

This section describes the technique to create Android applications from LaTeX files containing mathematical content. We have created unique technique of producing Android Apps from LaTeX files. This technique has a tremendous advantage for displaying mathematical content. No online resources are required for displaying mathematical content as it is pre-compiled. This is a significant advantage. It reduces dependability on server-side scripts such as MathJax, which needs an internet connection. The mathematical content typed in LaTeX preserves its formatting. Further the mathematical content is mobile responsive. It is not easy to create mobile responsive mathematics content using other techniques. Another advantage is that one can use javascript and/or jquery. It reduces dependability on java in android applications for scripting purposes, and it can be preferred over other techniques for creating android applications containing mathematical content.

The following outlines the procedure for creating an Android app using a LaTeX file.

I: Pre-compilation of LaTeX file using Lua and make4ht.

II: Adding jquery library and javascript file(s).

III: Adding CSS file(s).

IV: Using the WebView Class to create Android app.

4.1 Resources required

The following resources are required for implementing the technique.

- A TeX distribution (such as TeXLive or MikTeX) with Make4ht package.
- A LaTex file with the article document class.
- Node.js, npm and Mathjax node.
- A small collection of Mathjax fonts.
- A custom filter `.lua` file
- A custom build `.mk4` file
- A custom configuration `.cfg` file
- The javascript and jquery files.
- A Cascading Style Sheets (CSS) file.
- Android Studio or other IDE.
- A java code for webview class.

There is a command line application provided by mathjax-node-page. It is `mjpage`. This processes HTML files and substitutes LaTeX or MathML code with plain HTML or SVG code. The mathjax node can be installed by using the following inline command.

```
npm install -g mathjax-node-page
```

There should be a woff subdirectory in the working directory containing MathJax fonts in the woff format for using this technique. It can be downloaded from MathJax resources.

4.2 Precompilation of LaTeX file using Lua and make4ht

This subsection provides the procedure to produce pre-compiled html document using Lua and the make4ht [16] package.

With all the resources in the working directory, a latex file containing mathematical content can be pre-compiled with mathjax to html file by executing the following inline command.

```
make4ht -uc custom.cfg -e mybuild.mk4 file.tex html5
```

The source code of the custom filter `mathnode.lua` file is given in listing 3. It is used for pre-compilation of mathematical content in html files.

Listing 3: Source Code mathnode lua file

```
 -- local mathnodepath = os.getenv "mathjaxnodepath"
-- print("mathnode", mathnodepath)
local mkutils = require "mkutils"
```

```
-- other possible value is page2svg
local mathnodepath = "mjpage"
-- options for MathJax command
local options = "--output CommonHTML"
-- math fonts position
-- don't alter fonts if not set
local fontdir = nil
-- if we copy fonts
local fontdest = nil
local fontformat = "otf"

local function compile(src)
  local tmpfile = os.tmpname()
  local filename = src
  print("Compile using MathJax")
  local command = mathnodepath .. " ".. options .. " < " .. filename .. " > "
     .. tmpfile
  print(command)
  local status = os.execute(command)
  print("Result written to: ".. tmpfile)
  mkutils.cp(tmpfile, src)
  os.remove(tmpfile)
end

-- save the css code from the html page generated by MathJax
local function extract_css(file)
  local f = io.open(file, "r")
  local contents = f:read("*all")
  f:close()
  local css = ""
  local filename = "mathjax-chtml.css"
  contents = contents:gsub('<style [^>]+>(.+)</style>', function(style)
    -- replace only the style for mathjax
    if style:match "%.mjx%-math" then
      css = style
      return '<link rel="stylesheet" type="text/css" href="'..filename ..'" />'
    end
  end)
  local x = assert(io.open(file, "w"))
  x:write(contents)
  x:close()
  return filename, css
end

-- Update the paths to fonts to use the local versions
```

```
local function use_fonts(css)
  local family_pattern = "font-family:%s*(.-);.-%/([^%/]+)%.".. fontformat
  local family_build = "@font-face {font-family: %s; src: url('%s/%s.%s')
     format('%s')}"
  local fontdir = fontdir:gsub("/$","")
  css = css:gsub("(@font-face%s*{.-})", function(face)
    if not face:match("url(") then return face end
    -- print(face)
    local family, filename = face:match(family_pattern)
    print(family, filename)
    local newfile = string.format("%s/%s.%s", fontdir, filename, fontformat)
    Make:add_file(newfile)
    return family_build:format(family, fontdir, filename, fontformat,
       fontformat)
    -- return face
  end)
  return css
end

local function save_css(filename, css)
  local f = io.open(filename, "w")
  f:write(css)
  f:close()
end

return function(text, arguments)
  -- if arguments.prg then mathnodepath = arguments.prg end
  mathnodepath = arguments.prg or mathnodepath
  options    = arguments.options or options
  fontdir    = arguments.fontdir or fontdir
  fontdest   = arguments.fontdest or fontdest
  fontformat = arguments.fontformat or fontformat
  compile(text)
  filename, css = extract_css(text)
  -- use local font files if fontdir is present
  if fontdir then
    css = use_fonts(css)
  end
  save_css(filename, css)
  Make:add_file(filename)
  -- print(css)
  print(filename)
end
```

The source code of `mybuild.mk4` file is given in listing 4. It is a custom build mk4 file.

Listing 4: Source Code of mybuild mk4 file

```
local mathjax_node = require "mathnode"
local format = "woff"
Make:match("html$", mathjax_node, {fontdir = format, fontformat = format})
```

The custom configuration file `custom.cfg` can have the following code.

```
\Preamble{xhtml,mathml}
\begin{document}
\Css{body{font-family: MJXc-TeX-main-Rw, MJXc-TeX-main-Iw, MJXc-TeX-main-Bw,
   sans-serif;}}
\EndPreamble
```

These sets of codes are written by Michal Hoftich, a developer of make4ht package. It is available on the tex stack exchange website [19] and it can be freely used and shared under CC-BY-SA international license as per stack exchange website guidelines.

The MathJax node project is now archived. There is still better technique. The current version of make4ht now contains an extension for mjcli. It can be requested using the following command.

```
make4ht -f html5+mjcli filename.tex
```

4.3 Adding style and script files

After pre-compilation of LaTeX file into html file, style files (CSS files) and script files (javascript files and a jquery library) files are added to create a custom web app. This depends on the structure of the app that one wants to make. A user can design his style and script files to get the desired formatting and layout in the app.

4.4 Using the WebView Class to create Android app

There are many techniques that can be used to produce android applications from customized documents or web applications [2]. One technique to create android apps from latex files is to use `WebView class`. The `WebView class` [1] is an extension of android's `View class`. The WebView can display both online and offline web applications or even combination of online and offline applications. The advantage of this technique is that if online web applications are updated, then they get reflected in android applications without needing to incorporate updates in Android app. The user interface is preserved as well. The hierarchy of android webview in the java platform of android is given in Figure 3.

4.5 Examples based on the techniques

The following android applications are created based on developed technique.

1. Series of Real Numbers [21]
2. Riemann Integration [20]

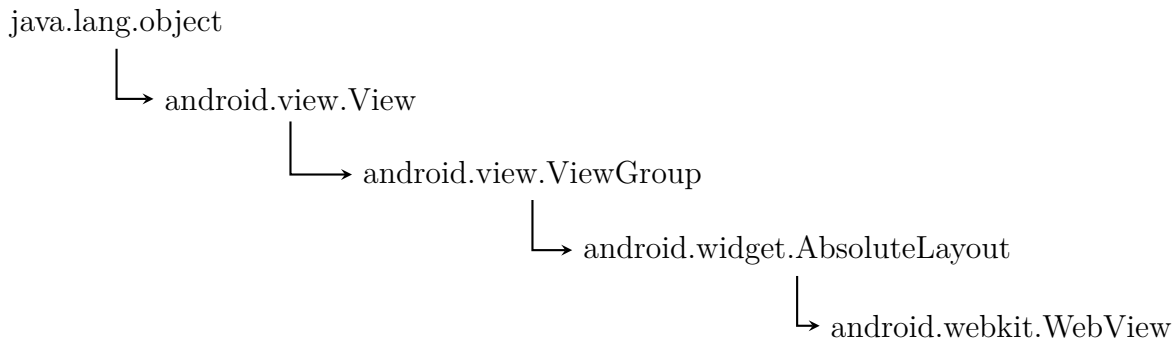

Figure 3: Android WebView Class

5 Conclusions

The following conclusions are drawn from the carried out work.

- Lua can effectively be used for the development of mathematical packages in LaTeX.
- Lua-based mathematical packages can reduce the dependence of LaTeX users on external software.
- Lua can be used to enhance graphical aspect of LaTeX.
- Lua-based mathematical packages can have pedagogical applications. Lua-based mathematical packages can be deployed to create interactive teaching modules in LaTeX. They can also be used to create different problem sets, assignments along with solution sets.
- Lua can be implemented in LaTeX to create customized documents, web applications and android applications.

6 Future work

The following ideas can be materialized in future.

- The techniques used in the creation of Android applications from latex files involve manual work of editing html and css files and scripts or programs. This part can easily be automated with Lua scripts. This automation will reduce the efforts required in the creation of Android applications.
- Lua-based mini Computer Algebra System can be developed and integrated into LaTeX. It will consist of Lua packages developed for performing symbolic mathematical computations.

References

[1] *Android WebView Class*. URL: `https://developer.android.com/reference/android/webkit/WebView` (visited on 05/20/2021).

[2] Bogdan-Vasile Cioruta and Mirela Coman. *Create your own applied mathematics software for Android mobile device*. Nov. 2018. ISBN: 978-620-2-31631-6. URL: https://www.researchgate.net/publication/328967784 (visited on 01/18/2022).

[3] John D. Hobby. *Drawing Graphs with MetaPost*. URL: https://tug.org/docs/metapost/mpgraph.pdf.

[4] *Lua Programming Language*. URL: https://www.lua.org/manual/5.4 (visited on 03/26/2022).

[5] *luacode package page*. URL: https://ctan.org/pkg/luacode (visited on 03/10/2022).

[6] *luacomplex package page*. URL: https://ctan.org/pkg/luacomplex (visited on 12/29/2022).

[7] *luagcd package page*. URL: https://ctan.org/pkg/luatruthtable (visited on 12/30/2022).

[8] *lualinalg package page*. URL: https://ctan.org/pkg/lualinalg (visited on 02/01/2023).

[9] *luamaths package page*. URL: https://ctan.org/pkg/luamaths (visited on 12/27/2022).

[10] *luamodulartables package page*. URL: https://ctan.org/pkg/luamodulartables (visited on 12/31/2022).

[11] *luamplib package*. URL: https://mirror.kku.ac.th/CTAN/macros/luatex/generic/luamplib/luamplib.pdf (visited on 02/22/2022).

[12] *luanumint package page*. URL: https://ctan.org/pkg/luanumint (visited on 08/04/2023).

[13] *luaplot package page*. URL: https://ctan.org/pkg/luaplot (visited on 07/08/2023).

[14] *luaset package page*. URL: https://ctan.org/pkg/luaset (visited on 12/28/2022).

[15] *luatruthtable package page*. URL: https://ctan.org/pkg/luatruthtable (visited on 09/18/2022).

[16] *Make4ht package*. 2015. URL: https://ctan.org/pkg/make4ht?lang=en (visited on 02/22/2022).

[17] *MetaPost System*. URL: https://www.tug.org/docs/metapost/mpman.pdf (visited on 02/22/2022).

[18] *PGF package page*. URL: https://ctan.org/pkg/pgf (visited on 01/10/2021).

[19] *Precompilation of Mathematical Content in LaTeX to HTML using MathJax*. URL: https://tex.stackexchange.com/questions/402010 (visited on 03/02/2022).

[20] *Riemann Integration*. 2023. URL: https://play.google.com/store/apps/details?id=com.unimaths.riemint (visited on 07/02/2023).

[21] *Series of Real Numbers*. 2023. URL: https://play.google.com/store/apps/details?id=com.unimaths.series (visited on 02/22/2022).

[22] Chetan Shirore and Ajit Kumar. "Basic Mathematical Computations inside LaTeX using Lua". In: *The Electronic journal of Mathematics and Technology* 17.1 (2023). ISSN: 1933-2807. URL: https://php.radford.edu/~ejmt/deliverAbstract.php?paperID=eJMT_v17n1n1.