Using `context` and `tikz` terminals for gnuplot in ConTEXt

Mojca Miklavec, 2013-04-19

With special thanks to Hans Hagen, Taco Hoekwater, Aditya Mahajan, Marco Patzer and others.

1 Requirements

1. Any ConTEXt installation (ConTEXt Distribution, TEX Live 2011 or later, ...).
2. Gnuplot 4.6.0 or later with `context` and/or `tikz` terminal built in. The `gnuplot` binary (or `gnuplot.exe` under Windows) has to be in `PATH`.
3. Gnuplot module for ConTeXt.
4. For running gnuplot on the fly or when using ConTEXt mkii, you need to have `write18` enabled. Usually this can be set with `shell_escape = t` in `texmf.cnf`.

2 Installation

2.1 Gnuplot

As long as you have gnuplot ≥ 4.6.0 installed, you should have the `context` terminal built in. If not, you might want to compile gnuplot from CVS sources:

```
cd gnuplot
```

```
./prepare
./configure [--prefix=$PWD/install]
make
make install
```

The option `--prefix=$PWD/install` (but you can choose anything) – if chosen – will install gnuplot locally even if you lack root permissions. Just make sure that you add the resulting binary to `PATH`.

Once you have the gnuplot binary running, you can check the list of supported terminals by typing

```
gnuplot> set term
```

into gnuplot shell. Make sure that it lists:

```
    context  ConTeXt with MetaFun (for PDF documents)
       tikz  Lua PGF/TikZ terminal for TeX and friends
```

The module also supports some other terminals like `png`, `metapost`, `postscript` and `pdf`, but their integration with ConTEXt is limited.

2.2 `t-gnuplot` module for ConTEXt

Under ConTEXt Distribution you can install the gnuplot module and Tik*Z* with an additional switch when running `first-setup`, for example:

```
./first-setup.sh --modules=gnuplot,tikz
```

If you have installed a `complete` or `context` scheme under TEX Live, gnuplot module and Tik*Z* might already be installed. Else you can use:

```
tlmgr install context-gnuplot
tlmgr install pgf
```

Under MiKTEX the module is installed automatically when it is first used (but at the moment of writing MiKTEX doesn't support ConTEXt).

3 Simple examples

3.1 Calling gnuplot directly

Let's first create a simple file (we will call it *example.plt*, but you may choose any name) with the contents below.

For `context` terminal:

```
set term context size 5in,3in standalone
set output "fullpage-example.tex"
plot sin(x)
plot cos(atan(x))*sin(x)
```

For `tikz` terminal:

```
set term tikz context size 5in,3in standalone createstyle
set output "fullpage-example.tex"
plot sin(x)
plot cos(atan(x))*sin(x)
```

In both cases the option `standalone` is used to create a complete ConTEXt document with one plot per page, including header and `\starttext` … `\stoptext`, so that it can be compiled directly. The option `createstyle` is used to create three files with required macros in working directory[1].

Both terminals should give you almost equivalent results apart from default plot size. You are highly encouraged to specify the desired plot size explicitly. You may scale the plot later on, but you probably want to get the desired proportions from the start.

[1] An alternative is to place those three files somewhere where kpathsea can find them and omit the option `createstyle`, just make sure that the versions of `tikz` terminal and the files in your TEX tree remain compatible.

Run gnuplot with

```
gnuplot example.plt
```

and compile the result with any of the following three commands (depending on your preferred engine):

```
context fullpage-example.tex                # for LuaTeX
texexec fullpage-example.tex                # for pdfTeX
texexec --xtx fullpage-example.tex          # for XeTeX
```

They are almost equivalent except that XeTeX lacks some advanced features (some patterns). The only major difference is the choice of fonts. If you want to typeset Arabic labels or use system fonts, you will probably want to choose LuaTeX or XeTeX. If you are using many graphical elements (3D plots, images, …), you might want to go for LuaTeX.

You should get a pdf document with two full-page plots that you can include into your document with `\externalfigure[fullpage-example][page=2]` for example.

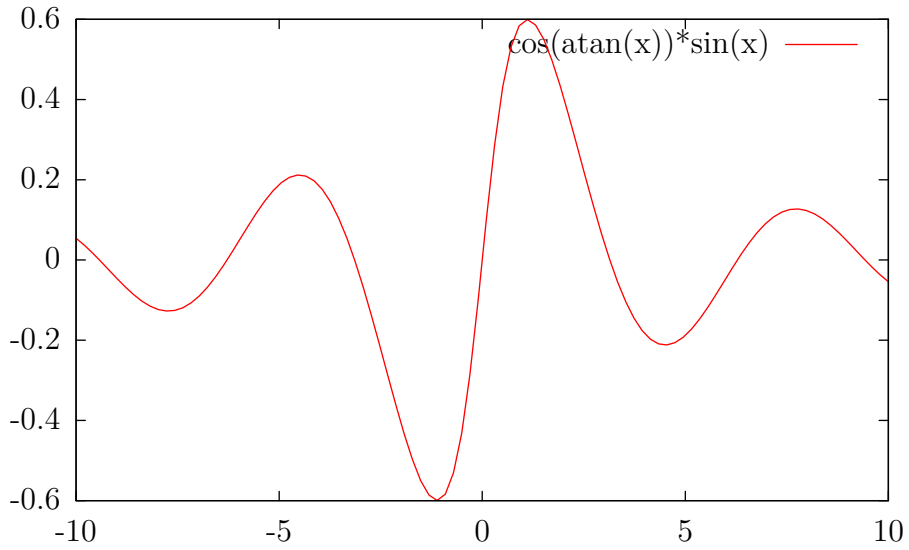

Figure 1 Second page from `fullpage-example`, included with `\externalfigure`

3.2 Calling gnuplot from TEX

As you can see you will always get Latin Modern font at 12pt unless you explicitly change it with `header "\setupbodyfont[somefontname,10pt]"` or with `font "somefontname,10pt"`. An easier way to make sure that the same font is used and to avoid having to call gnuplot manually is to simply type the gnuplot code inside your ConTEXt document:

```
\usemodule
       [gnuplot]
\setupGNUPLOTterminal
       [context]
       [width=5in,height=2.5in,fontscale=0.9]
\setupGNUPLOTterminal
       [tikz]
       [width=5in,height=2.5in,fontscale=0.9]
\starttext

\startGNUPLOTinclusions
set samples 400
set key left Left reverse
\stopGNUPLOTinclusion

\startGNUPLOTscript[myfunction]
set zeroaxis
set format y "%.1f"
plot [-4:2][0:2] 1 t '' lt 0, exp(x) t '$e^x$' lt 1 lw 3
plot cos(atan(x))*sin(x) t '$\cos(\arctan(x))\sin(x)$' lw 3 lc 3
\stopGNUPLOTscript

\placefigure{none}{\useGNUPLOTgraphic[myfunction][2]}

\setupGNUPLOT
       [terminal=tikz]

\placefigure{none}{\useGNUPLOTgraphic[myfunction][1]}

\stoptext
```

With `\setupGNUPLOT[terminal=<termname>]` you can select any supported gnuplot terminal before drawing a plot.

With `\setupGNUPLOT[<termname>][<option>=<value>]` you can set some terminal-specific options.

Anything inside `\startGNUPLOTinclusions... \stopGNUPLOTinclusion` will be applied to every plot.

The command `\startGNUPLOTscript[<name>]` creates new plots that can be included with `\useGNUPLOTgraphic[<name>][<number>][<option>=<value>]`. Both the number of plot and additional parameters (like `width=.7\textwidth` for example) are optional.

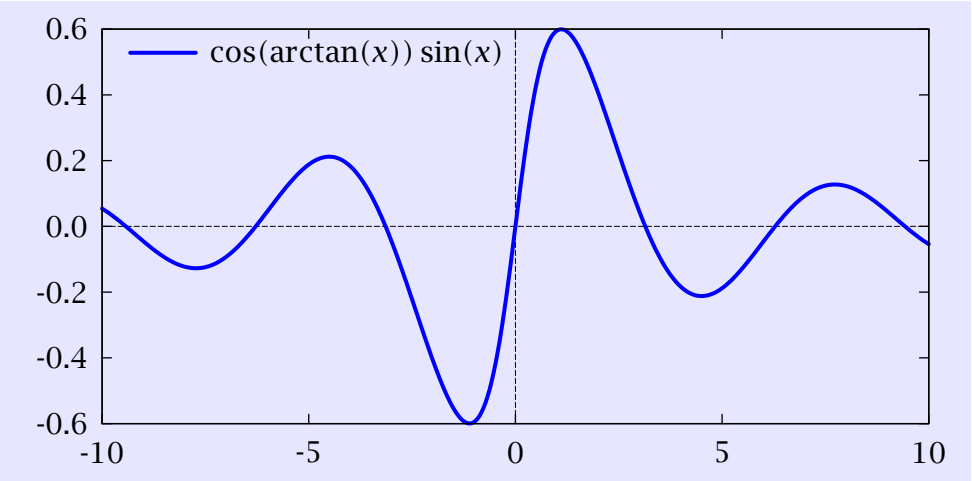

Figure 2 Framed second plot using `context` terminal

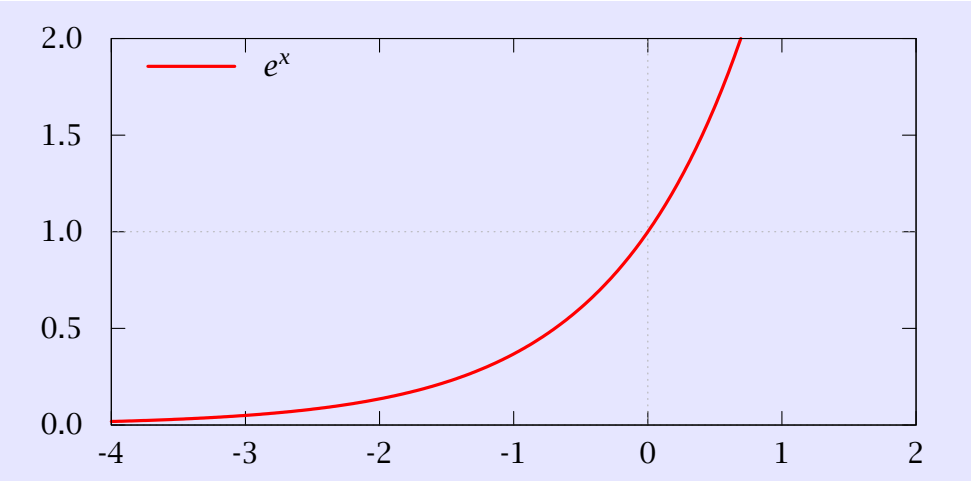

Figure 3 Framed first plot using `tikz` terminal

3.3 Including pre-generated plots

Instead of defining `\startGNUPLOTscript` and letting ConTEXt call gnuplot on the fly, you can also run gnuplot in advance and only include the resulting `filename.tex`. This is something that you might want to do when running calculation-intensive gnuplot scripts which take a long time.

You can follow the same steps as in section 3.1, except that you should not specify the `standalone` flag (and you should not compile the plot, only the main document).

The resulting file can be included[2] with

```
\processGNUPLOTfile[<name>][<filename.tex>]
```

and you can get the graphic with the same command as usual:

```
\useGNUPLOTgraphic[<name>]
```

plus any optional parameters.

[2] `\include filename.tex` won't work

4 Terminal options

4.1 context

```
set term context { default }
                 { defaultsize | size <xsize> {in|cm}, <ysize> {in|cm}
}
                 { input | standalone }
                 { timestamp | notimestamp }
                 { noheader | header "<header>" }
                 { color | colour | monochrome }
                 { rounded | mitered | beveled }
                 { round | butt | squared }
                 { dashed | solid }
                 { dashlength | dl <DL> }
                 { linewidth | lw <LW> }
                 { fontscale <fontscale> }
                 { mppoints | texpoints }
                 { inlineimages | externalimages }
                 { defaultfont | font {<fontsize>} |
                   font "<fontname>{,<fontsize>}" {fontsize} }
```

4.2 tikz

```
set term tikz { latex | tex | context }
              { size <x>{unit},<y>{unit} }
              { scale <x>,<y> }
              { nofulldoc | nostandalone | fulldoc | standalone }
              { color | monochrome }
              { dashed | solid }
              { nooriginreset | originreset }
              { nogparrows | gparrows }
              { nogppoints | gppoints }
              { picenvironment | nopicenvironment }
              { noclip | clip }
              { notightboundingbox | tightboundingbox }
              { background "<colorpec>" }
              { plotsize <x>{unit},<y>{unit} }
              { charsize <x>{unit},<y>{unit} }
              { font "<fontdesc>" }
              { fontscale <fontscale> }
              { {preamble | header} "<preamble_string>" }
              { tikzplot <ltn>,... }
              { notikzarrows | tikzarrows }
              { rgbimages | cmykimages }
              { noexternalimages | externalimages }
              { bitmap | nobitmap }
              { providevars <var name>,... }
              { createstyle }
              { help }
```

5 Comparison of supported terminals

The gnuplot module for ConTEXt supports the following terminals:

- **bitmap terminals**
  - `png, pngcairo`
- **vector terminals**
  - **`context, tikz`**
  - `metapost, postscript, pdf, pdfcairo`

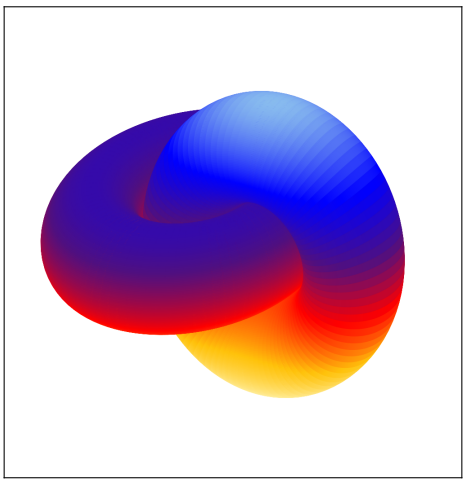

Figure 4 An example of graphic generated with png terminal

6 Known bugs

6.1 Buggy implementation in ConTEXt module

- Point sizes of TEX symbols for points have to be fine-tuned for proper size.
- Points don't scale properly. Line widths should not be scaled when bigger points are requested. Also, when thicker lines are used, points don't inherit that thickness. The reason is buggy implementation that stores all points as pictures in the beginning instead of drawing each point separately when that is requested.
- Patterns fills are a semi-hack. They are composed out of little tiles and drawn next to each other. This doesn't look properly when rendered. This also means that line widths don't scale properly.
- MetaPost could be highly optimized. In particular the transparency should be handled more efficiently.

6.2 Support in ConTEXt core

- Switching to a different font for font labels doesn't work in mkiv and uses an ugly hack in mkii.
- External images don't work in mkiv at the moment. Use `images=inline` (`inlineimages` in gnuplot). This is because the only acceptable mkii syntax is `externalfigure "name.png"`, while mkiv requires `draw externalfigure "name.png"`. This has to be fixed in ConTEXt core.
- Transparent inline images are not yet supported.
- There might be still some memory leaks in MetaPost. The major ones were fixed.

6.3 Limitations

- Plots with many graphical elements don't work in mkii since TEX runs out of memory.
- Inline bitmap images are not (and might never be) supported in mkii. If you want to use external bitmap images, use the option `externalimages` in `context` terminal (`images=external` in ConTEXt).