The HEP-GRAPHIC package*

TikZ, pgfplots and tikz-feynman setup with externalisation and figure inclusion helpers

Jan Hajer†

?

Abstract

The HEP-GRAPHIC package collects graphics helpers for HEP documents. It loads TikZ, optional TikZ externalisation, and TIKZSCALE, provides path-aware inclusion commands for PGF plots and Feynman diagrams, and can load companion HEP-PLOT and HEP-FEYNMAN modules for PGFPLOTS defaults, colour cycles, line styles, legends, and TIKZ-FEYNMAN settings.

1 Graphic

After loading the `hep-graphic` package the TikZ [1] package is loaded.

`extern` The `extern=`⟨*folder*⟩ option activates the TikZ `external` library and stores externalised graphics in ⟨*folder*⟩.

`plot` The `plot` and `feynman` options load the HEP-PLOT and HEP-FEYNMAN companion modules.

`feynman`
`\graphicpath`
`\includepgf`
`\includefeynman`

The `\graphicpath{`⟨*folder*⟩`}` macro stores the folder from which generated graphics are included. The `\includepgf`⟨***⟩`[`⟨*options*⟩`]{`⟨*name*⟩`}` macro includes a PGF or PGFPlots graphic from that folder and passes ⟨*options*⟩ to both `\pgfplotsset` and `\includegraphics`; the starred form includes the graphic without applying plotting options. The `\includefeynman{`⟨*name*⟩`}` macro includes a generated Feynman diagram centred for use in math displays.

1.1 Plot

The HEP-PLOT package loads the PGFPLOTS package [2], sets global plot dimensions and fonts, adjusts numeric formatting, and provides a `three panels` style for compact multi-panel figures. It defines `rainbow` and `rainbow reversed` colormaps, spaced colour and line-style cycle-list generators, matching cycle lists for up to many curves, and the `\addlegendtitle{`⟨*text*⟩`}` helper for legend headers.

1.2 Feynman

The HEP-FEYNMAN package loads the TIKZ-FEYNMAN package [3] and adjusts default line, momentum-arrow, and diagram styles.

*This document corresponds to HEP-GRAPHIC ?.

†jan.hajer@tecnico.ulisboa.pt

A Implementation

A.1 Graphic

<*package>

Define a hepgraphic namespace for the options using the KVOPTIONS package [4].

```
\RequirePackage{kvoptions}
\SetupKeyvalOptions{
  family=hepgraphic,
  prefix=hepgraphic@
}
```

`plot` Define the `extern` option to activate externalisation into the declared folder.

```
\DeclareStringOption{extern}[.]
```

`plot` Define the `plot` switch for loading plot code.

```
\DeclareBoolOption[false]{plot}
```

`feynman` Define the `feynman` switch for loading feynman code.

```
\DeclareBoolOption[false]{feynman}

\ProcessKeyvalOptions*
```

Load the HEP-PLOT and HEP-FEYNMAN packages when required.

```
\ifhepgraphic@plot\RequirePackage{hep-plot}\fi
\ifhepgraphic@feynman\RequirePackage{hep-feynman}\fi
```

Load the TIKZ package with the EXTERNAL library [1].

```
\RequirePackage{tikz}
\ifx\hepgraphic@extern\@empty\else
 \usetikzlibrary{external}
 \tikzexternalize[
%   optimize=false,
  only named=true,
 ]
 \tikzsetexternalprefix{\hepgraphic@extern/}
\fi
```

`\graphicpath` Load the STANDALONE package [5] and define the `\graphicpath` pointing to the folder with `pgf` files.

```
\RequirePackage{tikzscale}
\def\hep@graphic@path{.}
\newcommand*{\graphicpath}[1]{\def\hep@graphic@path{#1}}
```

`\includetikz` Define a macro to include `tikz` figures using the XPARSE package [6]. The name of the externalised file applies only to the picture in the included `tikz` file and is cleared afterwards, so that it never names a later picture.

```
\RequirePackage{xparse}
\providecommand*{\tikzsetnextfilename}[1]{}
\NewDocumentCommand{\includepgf}{sO{}m}{%
 \tikzsetnextfilename{#3}%
 \IfBooleanTF{#1}{%
  \includegraphics{\hep@graphic@path/#3}%
 }{%
  \begingroup
  \ifdefined\pgfplotsset\pgfplotsset{#2}\fi
  \includegraphics[#2]{\hep@graphic@path/#3}%
  \endgroup
 }%
 \tikzsetnextfilename{}%
}
\newcommand*{\includefeynman}[1]{%
 \vcenter{\hbox{\includegraphics{\hep@graphic@path/#1}}}%
}
```

</package>

A.2 Plots

<*plot>

Load the PGF/TIKZ package [1].

```
\RequirePackage{tikz}
```

`dashdotdotdotted` `dashdotdotdotdotted` Add new line styles.

```
% \tikzset{
%  dashdotdotdotted/.style={dash pattern=on 3pt off 2pt
%   on \the\pgflinewidth off 2pt on \the\pgflinewidth off 2pt
%   on \the\pgflinewidth off 2pt
%  },
%  dashdotdotdotdotted/.style={dash pattern=on 3pt off 2pt
%   on \the\pgflinewidth off 2pt on \the\pgflinewidth off 2pt
%   on \the\pgflinewidth off 2pt on \the\pgflinewidth off 2pt
%  },
%  dotdotdashed/.style={dash pattern=on 3pt off 2pt
%   on 3pt off 2pt on \the\pgflinewidth off 2pt
%  },
%  dotdotdotdashed/.style={dash pattern=on 3pt off 2pt
%   on 3pt off 2pt on 3pt off 2pt on \the\pgflinewidth off 2pt
%  },
% }
```

Change thousand separator

```
\pgfkeys{/pgf/number format/.cd,1000 sep={\,}}%
```

Load the PGFPLOTS package [2] and set global options.

```
\RequirePackage{pgfplots}
\pgfplotsset{
 compat=newest,
 width=\linewidth,
 height=\linewidth,
 enlargelimits=false,
}
```

Fix glitch.

```
\pgfplotsset{
 every y tick scale label/.append style={
  inner sep=1pt,
  xshift=-1pt,
  yshift=-1pt,
 },
}
```

Set default font size

```
\pgfplotsset{
  legend style={font=\footnotesize},
  tick label style={font=\footnotesize},
  label style={font=\small},
  title style={font=\small},
  max space between ticks=30,
}
```

`three panels` Set font size three panel versions

```
\pgfplotsset{
  three panels/.style={
    legend style={font=\scriptsize},
    tick label style={font=\scriptsize},
    label style={font=\footnotesize},
    title style={font=\footnotesize},
    max space between ticks=25,
    /tikz/mark size=1.5pt,
    major tick length=1mm,
    minor tick length=0.66mm,
    every axis title shift=0pt,
  },
}
```

A.2.1 Cycles

`rainbow reversed` Define the `rainbow reversed` colormap. Approximate spectral colormap for visible white-light dispersion in wavelength ordering.

```
\usepgfplotslibrary{colormaps}
\pgfplotsset{
  colormap={rainbow reversed}{
    rgb255(410)=(38,0,79)
    rgb255(430)=(67,0,170)
    rgb255(450)=(58,0,200)
    rgb255(470)=(0,50,180)
    rgb255(490)=(0,100,120)
    rgb255(510)=(0,135,75)
    rgb255(530)=(0,150,35)
    rgb255(550)=(55,145,0)
    rgb255(570)=(174,155,0)
    rgb255(590)=(226,110,0)
    rgb255(610)=(241,0,0)
    rgb255(630)=(205,0,0)
    rgb255(650)=(141,0,0)
  },
```

`rainbow` Define the `rainbow` colormap in frequency ordering.

```
  colormap={rainbow}{
    colors of colormap={
      650,630,610,590,570,550,530,510,490,470,450,430,410
      of rainbow reversed,
      source range=410:650,
      target pos={
        0,83,167,250,333,417,500,583,667,750,833,917,1000
      }
    }
  },
}
```

`ainbow_positions_seq` `l_rainbow_target_int` `bow_positions_full_p` True once the current generated list has reached its requested length.

```
\ExplSyntaxOn
\seq_new:N \l_rainbow_positions_seq
\int_new:N \l_rainbow_target_int
\cs_new:Npn \rainbow_positions_full_p:{
  \int_compare_p:n{
    \seq_count:N \l_rainbow_positions_seq >= \l_rainbow_target_int
  }
}
```

`rainbow_add_position` Append a colormap coordinate if the requested number of colors has not yet been reached. Values are rounded only to keep printed output readable.

```
\cs_new_protected:Npn \rainbow_add_position:n #1{
```

```
  \int_compare:nNnT {
    \seq_count:N \l_rainbow_positions_seq } < { \l_rainbow_target_int
  }{
    \seq_put_right:Nx \l_rainbow_positions_seq{\fp_eval:n{round(#1, 6)}}
  }
}
```

ainbowCyclePositions
nbow_denominator_int
cedColormapCycleList

`\CreateSpacedColormapCycleList{<cycle list>}{<colormap>}{<count>}` Builds a PGFPlots cycle list by sampling <count> colors from <colormap>. The positions are generated by repeatedly bisecting the largest remaining gaps. Within each bisection level they are emitted from the outside inward: 0,1000,500,250,750,125,875,... This preserves even coverage while keeping neighboring colors visually distinct. Side effect: `\RainbowCyclePositions` is set to the comma-separated list of generated colormap coordinates for logging, inspection, or typesetting.

```
\newcommand{\RainbowCyclePositions}{}
\int_new:N \l_rainbow_denominator_int
\int_set:Nn \l_rainbow_denominator_int { 2 }
\NewDocumentCommand{\CreateSpacedColormapCycleList}{mmm}{
  \seq_clear:N \l_rainbow_positions_seq
  \int_set:Nn \l_rainbow_denominator_int { 2 }
  \int_set:Nn \l_rainbow_target_int { \int_max:nn { 1 } { #3 } }
  \rainbow_add_position:n { 0 }
  \rainbow_add_position:n { 1000 }
```

Largest-gap bisection, ordered outside-in for high adjacent contrast.

```
  \bool_until_do:nn { \rainbow_positions_full_p: }{
    \int_step_inline:nnnn { 1 } { 2 } { \l_rainbow_denominator_int / 2 }{
      \rainbow_add_position:n{ 1000 * ##1 / \l_rainbow_denominator_int }
      \int_compare:nNnT { ##1 } < { \l_rainbow_denominator_int - ##1 }{
        \rainbow_add_position:n{
          1000*(\l_rainbow_denominator_int-##1)/\l_rainbow_denominator_int
        }
      }
    }
    \int_set:Nn \l_rainbow_denominator_int{2*\l_rainbow_denominator_int}
  }
  \cs_gset:Npx \RainbowCyclePositions{
    \seq_use:Nn \l_rainbow_positions_seq { , }
  }
  \exp_args:Nnx \pgfplotscreateplotcyclelist { #1 }{
    [ colors~of~colormap = { \RainbowCyclePositions ~ of ~ #2 } ]
  }
}
```

inbow_line_styles_tl
inbow_line_count_int
inbow_line_add_style

Append one line style to the line-style cycle list, stopping at the requested total count.

```
\tl_new:N \l_rainbow_line_styles_tl
\int_new:N \l_rainbow_line_count_int
\cs_new_protected:Npn \rainbow_line_add_style:n #1{
  \int_compare:nNnT{\l_rainbow_line_count_int}<{\l_rainbow_target_int}{
    \tl_put_right:Nn \l_rainbow_line_styles_tl { #1 \\ }
```

```
    \int_incr:N \l_rainbow_line_count_int
  }
}
```

\LineStyleDotLength
\LineStyleDashLength
\LineStyleGapLength
nbow_line_pattern_tl

Dot, dash, and gap lengths are functions of `\pgflinewidth`, so the pattern scales when the plot line width changes. Filled by `\CreateSpacedColormapCycleList` with values such as 0,1000,500,250,750,... Append one dashed style made from `#1` dashes followed by `#2` round dots. Dots are zero-length dash segments with round line caps.

```
\newcommand{\LineStyleDotLength}{0pt}
\newcommand{\LineStyleDashLength}{4\pgflinewidth}
\newcommand{\LineStyleGapLength}{3\pgflinewidth}
\tl_new:N \l_rainbow_line_pattern_tl
\cs_new_protected:Npn \rainbow_line_add_pattern:nn #1#2{
  \tl_set:Nn \l_rainbow_line_pattern_tl { line~cap=round,dash~pattern= }
  \prg_replicate:nn { #1 }{
    \tl_put_right:Nn \l_rainbow_line_pattern_tl
    { on~\LineStyleDashLength~off~\LineStyleGapLength~ }
  }
  \prg_replicate:nn { #2 }{
    \tl_put_right:Nn \l_rainbow_line_pattern_tl
    { on~\LineStyleDotLength~off~\LineStyleGapLength~ }
  }
  \exp_args:NV \rainbow_line_add_style:n \l_rainbow_line_pattern_tl
}
```

nbow_line_repeat_int
teLineStyleCycleList

`\CreateLineStyleCycleList{<cycle list>}{<count>}` Builds a PGFPlots cycle list of <count> line style cycle. Order: solid, dash, dot, dash dot, dash dash dot, dash dot dot, dash dash dot dot, dash dash dash dot, dash dot dot dot, ... This orders styles by the smallest repetition count first, while preferring dash-leading patterns over dot-leading patterns.

```
\int_new:N \l_rainbow_line_repeat_int
\NewDocumentCommand{\CreateLineStyleCycleList}{mm}{
  \tl_clear:N \l_rainbow_line_styles_tl
  \int_zero:N \l_rainbow_line_count_int
  \int_set:Nn \l_rainbow_target_int { \int_max:nn { 1 } { #2 } }
  \rainbow_line_add_style:n { solid }
  \rainbow_line_add_pattern:nn { 1 } { 0 }
  \rainbow_line_add_pattern:nn { 0 } { 1 }
  \rainbow_line_add_pattern:nn { 1 } { 1 }
  \int_set:Nn \l_rainbow_line_repeat_int { 2 }
  \bool_until_do:nn{
    \int_compare_p:n { \l_rainbow_line_count_int >= \l_rainbow_target_int }
  }{
    \int_step_inline:nn { \l_rainbow_line_repeat_int - 1 }{
      \rainbow_line_add_pattern:nn{ \l_rainbow_line_repeat_int }{ ##1 }
      \rainbow_line_add_pattern:nn{ ##1 }{ \l_rainbow_line_repeat_int }
    }
    \rainbow_line_add_pattern:nn{
      \l_rainbow_line_repeat_int
    }{
```

```
      \l_rainbow_line_repeat_int
    }
    \int_incr:N \l_rainbow_line_repeat_int
  }
  \exp_args:NnV \pgfplotscreateplotcyclelist { #1 } \l_rainbow_line_styles_tl
}
\ExplSyntaxOff
```

LineStyleCycleLength
line style cycle
\RainbowCycleLength
rainbow cycle
line styles
colors

\RainbowCycleLength controls how many colors are generated. \LineStyleCycleLength controls how many line style cycle are generated.

```
\newcommand{\LineStyleCycleLength}{64}
\CreateLineStyleCycleList{line style cycle}{\LineStyleCycleLength}
\newcommand{\RainbowCycleLength}{64}
\CreateSpacedColormapCycleList{rainbow cycle}{rainbow}{\RainbowCycleLength}
\CreateLineStyleCycleList{line styles}{\LineStyleCycleLength}
\CreateSpacedColormapCycleList{colors}{rainbow}{\RainbowCycleLength}

\pgfplotsset{
  cycle multiindex* list={
    rainbow cycle
    \nextlist
    line style cycle
    \nextlist
  },
}
```

colors
line styles
marks
horizontal marks
vertical marks

Create cycle lists

```
\colorlet{dark green}{green!50!black}
\colorlet{darkgreen}{dark green}
% \pgfplotscreateplotcyclelist{colors}{
%  blue, red, dark green, violet, orange, yellow!75!orange,
%  brown, black
% }
% \pgfplotscreateplotcyclelist{line styles}{
%  solid, dashed, densely dotted, dashdotted,
%  dashdotdotted, dotdotdashed, dashdotdotdotted, dotdotdotdashed, dashdotdotdotdotted
% }
\pgfplotscreateplotcyclelist{marks}{
 mark=-, mark=|, mark=Mercedes star flipped, mark=Mercedes star, mark=+,
 mark=x, mark=star, mark=asterisk, mark=10-pointed star
}
\pgfplotscreateplotcyclelist{vertical marks}{
 mark=|, mark=Mercedes star flipped, mark=Mercedes star, mark=x, mark=star,
 mark=asterisk, mark=10-pointed star
}
\pgfplotscreateplotcyclelist{horizontal marks}{
 mark=-, mark=Mercedes star flipped, mark=Mercedes star, mark=x, mark=star,
 mark=asterisk, mark=10-pointed star
}
```

```
\pgfplotscreateplotcyclelist{star marks}{
 mark=Mercedes star flipped, mark=Mercedes star, mark=x, mark=star,
 mark=asterisk, mark=10-pointed star
}
\pgfplotsset{
 cycle multiindex* list={colors\nextlist line styles},
}
```

`\cyclelistshift` Define the `\cyclelistshift` macro, which skips one more step of the cycle list for the following plots of the current axis. The global counter accumulates the shifts and must be reset with `\setcounter{cyclelistshift}{0}` before the next axis.

```
\newcounter{cyclelistshift}
\newcommand*\cyclelistshift{%
  \addtocounter{cyclelistshift}{1}%
  \pgfplotsset{cycle list shift/.expanded=\the\value{cyclelistshift}}%
}
```

`legend` Set the legend style.

```
\pgfplotsset{
 legend cell align=left,
 legend style={
  at={(1,1)},
  anchor=north east,
  inner sep=1pt,
  outer sep=6pt,
  draw=none,
  fill opacity=.9,
  draw opacity=1,
  text opacity=1,
  cells={align=left},
  /tikz/every even column/.append style={column sep=.5em},
%   fill=none,
 },
}
```

A.2.2 Contour plots

`contour legend` Define basic contour legend

```
\pgfplotsset{
 contour legend/.style={
%   contour prepared={labels=false},
  colorbar sampled line,
  colorbar style={
   mark size=7pt,
   mark options={semithick},
   tickwidth=0pt,
   subtickwidth=0pt,
   },
 },
```

```
}
```

`contour legend x` Define horizontal contour legend.

```
% \usepgfplotslibrary{colormaps}
\pgfplotsset{
 contour legend x/.style={
  colorbar horizontal,
  colormap name=rainbow,
%   colormap/rainbow,
  contour legend,
  colorbar style={
   at={(0.5,1.025)},
   anchor=south,
   mark=|,
   axis x line*=top,
   axis y line=none,
   xticklabel pos=upper,
   title style={
    at={(-0.05,1)},
    anchor=east,
   },
   xlabel style={
    at={(-0.06,1)},
    anchor=south east,
   },
  },
 },
}
```

`contour legend y` Define vertical contour legend.

```
\pgfplotsset{
 contour legend y/.style={
  contour legend,
  colorbar style={
   at={(1.025,0.5)},
   anchor=west,
   mark=-,
   axis x line=none,
   title style={
    at={(1,-0.1)},
    anchor=north west,
   },
  },
 },
}
```

`contour plot x` Define vertical contour legend.
`contour plot y`

```
\pgfplotsset{
```

```
 contour plot x/.style={
  contour legend x,
  contour prepared={labels=false},
 },
 contour plot y/.style={
  contour legend y,
  contour prepared={labels=false},
 },
}
```

`error legend` Define error legend.

```
\pgfplotsset{
 error legend/.style n args={3}{
  legend image code/.code={
   \draw[draw=none,fill=#1,#3](0mm,-1mm)rectangle(6mm,1mm);
   \draw[draw=#1,#2] (0mm,0mm)--(6mm,0mm);
  }
 },
}
```

`\addlegendtitle` Define a legend title macro.

```
\newcommand*{\addlegendtitle}[2][]{
 \addlegendimage{empty legend}
 \addlegendentry[#1]{\hspace{-7mm}#2}
}%
```

</plot>

A.3 Feynman graphs

<*feynman>

Load TIKZ-FEYNMAN package [3] to enable the drawing of Feynman diagrams. Deactivate warning

```
\RequirePackage{tikz-feynman}
\tikzfeynmanset{
 compat=1.1.0,
 warn luatex=false,
}
\makeatletter\def\tikzfeynman@luatex@required@path{}\makeatother
```

Redfine the arrow style

```
\tikzfeynmanset{
 with arrow/.style={%
  decoration={markings,mark=at position#1with\arrow{>}},
  postaction=decorate
 },
 with reversed arrow/.style={%
  decoration={markings,mark=at position#1with\arrow{<}},
```

```
  postaction=decorate
 },
 momentum/arrow style={->},
}
```

</feynman>

B Tests

<*test>

```
\documentclass{article}

\usepackage{hep-graphic}

\begin{document}

\end{document}
```

</test>

C Readme

<*readme>

```
# The ‘hep-graphic‘ package

TikZ, pgfplots and tikz-feynman setup with externalisation and figure
inclusion helpers

## Introduction

The ‘hep-graphic‘ package collects graphics helpers for HEP documents. It
loads TikZ, optional TikZ externalisation, and ‘tikzscale‘, provides
path-aware inclusion commands for generated PGF plots and Feynman
diagrams, and can load companion ‘hep-plot‘ and ‘hep-feynman‘ modules for
‘pgfplots‘ defaults, colour cycles, line styles, legends, and
‘tikz-feynman‘ settings.

It can be loaded with ‘\usepackage{hep-graphic}‘. The ‘plot‘ option loads
the plot module, the ‘feynman‘ option loads the Feynman-diagram module,
and ‘extern=<folder>‘ activates TikZ externalisation into the selected
folder.

## Author

Jan Hajer

## License

```

```
This file may be distributed and/or modified under the conditions of the
‘LaTeX‘ Project Public License, either version 1.3c 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 1.3c or later is
part of all distributions of LaTeX version 2005/12/01 or later.
```

</readme>

References

[1] T. Tantau and H. Menke. ‘The pgf package: Create PostScript and PDF graphics in TEX’ (2005). CTAN: pgf.

[2] C. Feuersänger. ‘The pgfplots package: Create normal/logarithmic plots in two and three dimensions’ (2007). CTAN: pgfplots.

[3] J. Ellis. ‘The pgf package: Feynman diagrams with TikZ’ (2016). CTAN: tikz-feynman.

[4] H. Oberdiek. ‘The kvoptions package: Key value format for package options’ (2004). CTAN: kvoptions. GitHub: ho-tex/kvoptions.

[5] M. Scharrer. ‘The standalone package: Compile TEX pictures stand-alone or as part of a document’ (2010). CTAN: standalone.

[6] *LATEX3 Project*. ‘The xparse package: A generic document command parser’ (1999). CTAN: xparse.

Change History

v1.2
General: First release on CTAN: hep-graphic, hep-plot, and hep-feynman with TikZ externalisation, PGF and Feynman diagram inclusion, pgfplots defaults, cycle lists, legend helpers, contour and error legend styles, and tikz-feynman settings. 1

v1.3
General: Refreshed the documentation. . . . 1

v1.4
General: Applied starred command definitions. 1

v1.5
General: Raised the LaTeX format requirement, expanded the documentation and README, and documented the colormap and cycle-list helpers. 1

v1.6
General: Fixed the colormap positions of repeated spaced cycle lists, \includepgf without the plot option or the external library, and the spurious spaces and global plot options of \includepgf and \cyclelistshift. 1