tikz-palattice - draw particle accelerator lattices with Ti*k*Z

Jan Schmidt <schmidt@physik.uni-bonn.de>

v2.1 (February 23, 2015)

The tikz-palattice package allows for drawing a map of a particle accelerator just by giving a list of elements - similar to lattice files for simulation software. The package includes 12 common element types like dipoles, quadrupoles, cavities or screens, as well as automatic labels with element names, a legend, a rule and an environment to fade out parts of the accelerator. The coordinate of any element can be saved and used for custom tikz drawings or annotations. Thereby, lattices can be connected to draw injection/extraction or even a complete accelerator facility.

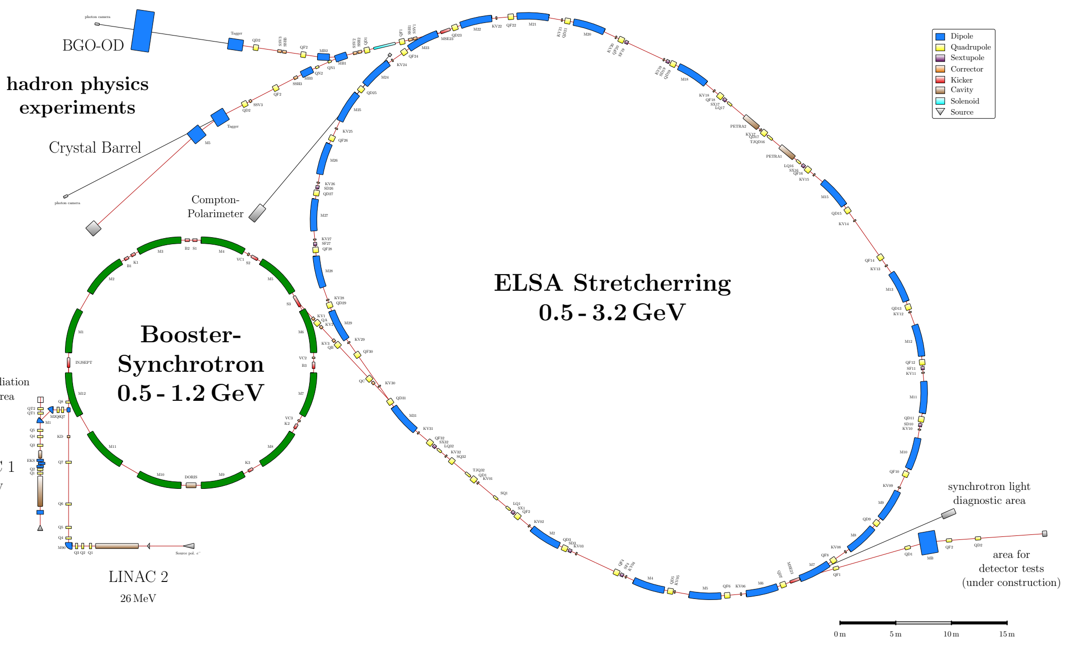

BGO-OD
hadron physics experiments
Crystal Barrel
Compton-Polarimeter
ELSA Stretcherring 0.5 - 3.2 GeV
Booster-Synchrotron 0.5 - 1.2 GeV
irradiation area
LINAC 1 20 MeV
LINAC 2 26 MeV
synchrotron light diagnostic area
area for detector tests (under construction)
Dipole
Quadrupole
Sextupole
Corrector
Kicker
Cavity
Solenoid
Source
0 m
5 m
10 m
15 m

Figure 1: The Electron Stretcher Facility ELSA at Bonn University, drawn with tikz-palattice

Contents

1 Installation

1.1 Copy tikz-palattice.sty

You just need to copy the lattice.sty file to a place where your LaTeX installation can recognize it. This can be

- the same folder as your .tex document or
- in the LaTeX system or user tree.

E.g. to add it to the system tree for texlive under ubuntu:

```
sudo mkdir -p /usr/local/share/texmf/tex/latex/lattice/
sudo cp lattice.sty /usr/local/share/texmf/tex/latex/lattice/
sudo mktexlsr
```

For this path there is also a Makefile prepared, so just enter

```
sudo make install
```

Otherwise read the documentation of your LaTeX distribution.

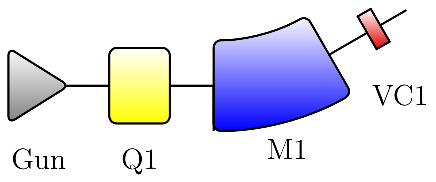
1.2 Required packages

- tikz
- siunitx
- ifthen
- xargs
- etoolbox

2 Basic Usage

Basically, an accelerator lattice is drawn by writing element commands one after the other. White spaces and newlines are ignored. Usually you will use one `drift` and one other element alternately:

```
\begin{lattice}
  \source{Gun}{0.4}
  \drift{0.267}
  \quadrupole{Q1}{0.4}
  \drift{0.29}
  \dipole{M1}{0.8}{30}
  \drift{0.29}
  \kicker{VC1}{0.1}
  \drift{0.2}
\end{lattice}
```



Descriptions of the commands are given in section 4. There are 5 examples coming with this package. Some of the drawings are shown in this document. Please look at the separate tex-files for the source code.

Some general remarks:

- it is recommended to draw lattices using `\documentclass{standalone}` especially for larger beamlines.
- lengths are set in meter, so you write {1.32} for 1.32 m.
- beamline with angle 0° goes to the right, positive angles bend counter clockwise.
- settings (colors, font, rotatelabel,...) changed within a `scope` environment are set back to the previous values outside of `scope`
- picture scale: for lattice scale=1 an element of 1m length is plotted with 2 cm length

Below, the arguments of commands are given in angle brackets. The values after an equal sign are default values of optional arguments. E.g. ⟨*scale=1*⟩ indicates that the default value of "scale" is 1.

3 The lattice Environment

To draw a lattice just add

```
\usepackage{lattice}
```

to your preambel and use the lattice environment with

```
 \begin{lattice}[⟨scale=1⟩][⟨tikz options=⟩]
   ...
 \end{lattice}
```

The lattice environment contains a tikzpicture environment in which the lattice is drawn using usual tikz commands. The lattice environment has 2 optional arguments:

1. ⟨*scale=1*⟩ scales whole picture (default is 1).
2. ⟨*tikz options=*⟩ gives any options for the tikzpicture (e.g. overlay, default is none).

4 Within lattice Environment

4.1 Elements

Here is a list of all implemented element types. The element names are self-explanatory:

```
 \drift{⟨length/m⟩}[⟨name=⟩]
 \dipole{⟨name⟩}{⟨arc length/m⟩}{⟨bending angle/deg⟩}[⟨type=s⟩][⟨thickness/m=0.6⟩]
 \quadrupole{⟨name⟩}{⟨length/m⟩}[⟨thickness/m=0.5⟩]
 \sextupole{⟨name⟩}{⟨length/m⟩}[⟨thickness/m=0.3⟩]
 \corrector{⟨name⟩}{⟨length/m⟩}[⟨thickness/m=0.25⟩]
 \kicker{⟨name⟩}{⟨length/m⟩}[⟨thickness/m=0.25⟩]
 \cavity{⟨name⟩}{⟨length/m⟩}[⟨thickness/m=0.45⟩]
 \solenoid{⟨name⟩}{⟨length/m⟩}[⟨thickness/m=0.2⟩]
 \beamdump{⟨name⟩}{⟨length/m⟩}[⟨thickness/m=0.5⟩]
 \source{⟨name⟩}{⟨length/m⟩}[⟨thickness/m=0.5⟩]
 \screen{⟨name⟩}[⟨length/m=0.2⟩]
 \valve{⟨name⟩}
 \marker{⟨name⟩}[⟨length/m=0.35⟩] % a line perpendicular to the beamline, see Fig. 2
```

The dipole option ⟨*type=s*⟩ allows to select different dipole shapes. It is shown in example 2. Possible values are:

- s for a sector magnet (entrance/exit surface 90 degree to beampipe)
- br for a bend rectangle magnet (parallel entrance/exit surfaces)
- r for a rectangle magnet

If you use any other letters, also the default (s) is used.

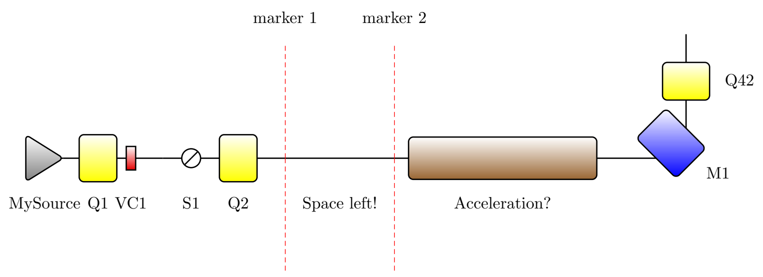


Figure 2: Example 1

4.2 Orientation of the lattice

```
\start{⟨coordinate/m⟩}
```

sets starting point of the lattice. Use it before the first element. *⟨coordinate/m⟩* is of form (x,y) or any tikz label, e.g. (mylabel.east) You can use this with \savecoordinate (section 4.5) to connect lattices, but it is recommended to do this via \goto (see below). Both are shown in example 3.

```
\rotate{⟨angle/deg⟩}
```

bends the beamline by the given angle.

```
\setangle{⟨angle/deg⟩}
```

sets the beamline angle to the given angle. The next element is drawn with beam axis in this direction.

```
\goto{⟨coordinate name⟩}
```

sets current position and angle to values saved with \savecoordinate (section 4.5). Use this to connect lattices and draw injection, extraction or even a complete accelerator facility. This is shown in example 3.

4.3 Rule and Legend

```
\drawrule{⟨position/m⟩}[⟨tick distance/m=1⟩][⟨scale=1⟩][⟨height/m=0.1⟩]
```

draws a rule to visualize the size of the lattice. Coordinate is of form (x,y) or any tikz label, e.g. (mylabel.east)

```
\legend{⟨position/m⟩}[⟨scale=1⟩]
```

draws a legend with all element types that occur in the lattice before this command. The given *⟨position/m⟩* is north west (upper left corner) of the legend box. The scale option scales the whole box including the text, which has the usual label textsize if scale=1.

```
\completelegend{⟨position/m⟩}[⟨scale=1⟩]
```

is similar to \legend, but shows all element types.

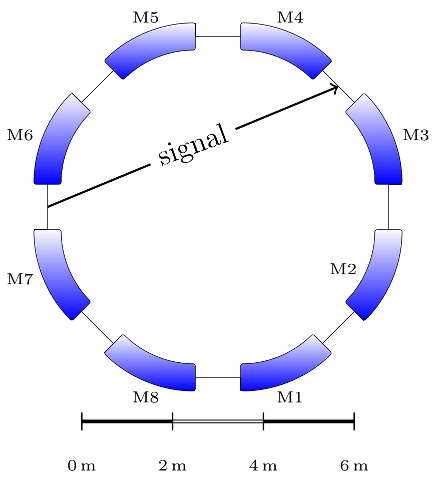


Figure 3: From example 3

4.4 Labels

Every element has a text label showing the given element name. The position and orientation of the label is set automatically according to the current angle of the beamline. Several commands to modify the labels manually are described below. If you want to disable labels, leave the element names blank or set the label color (section 4.4.1) to your background color.

```
\turnlabels
```

moves labels to other side of elements (swap with marker labels)

```
\rotatelabels{⟨angle/deg⟩}[⟨anchor=⟩]
```

allows rotation of element labels. The ⟨*anchor=*⟩ sets the center of rotation (north, center, south west, . . .). West corresponds to the labels first character. By default the anchor is set automatically depending on the current angle of the beamline.

```
\begin{labeldistance}{⟨distance/m⟩}
  ...
\end{labeldistance}
```

sets the distance of text labels to the element center for all elements within this environment. Default is 0.35.

```
\setlabeldistance{⟨distance/m⟩}
\resetlabeldistance
```

sets the distance of text labels to the element center for all following elements. the reset command sets the default value. Default is 0.35.

```
\setlabelfont{⟨fontsize⟩}
```

sets the text label font size. Default is **\normalsize**.

4.4.1 Colors

The colors can be changed at any point of the lattice. A setting is valid until the next color command. The reset commands set the according default color. Use a scope environment to change a color for a section of a lattice.

```
\setlinecolor{⟨type⟩}{⟨color⟩}
\resetlinecolor{⟨type⟩}
```

for ⟨*type*⟩ drift and marker.

```
\setelementcolor{⟨type⟩}{⟨color⟩}[⟨gradient color=white⟩]
\resetelementcolor{⟨type⟩}
```

for all element types. Set ⟨*gradient color=white*⟩ equal to ⟨*color*⟩ to "disable" the gradient.

```
\setlabelcolor{⟨color⟩}
```

for textlabels. Set to background color to hide text labels.

```
\begin{fade}[⟨opacity=0.25⟩]
  ...
\end{fade}
```

reduces the opacity of all elements within the environment and sets all colors to gray. So you can fade out regions of the lattice - e.g. for presentations. This can also be used to completely hide regions by setting ⟨*opacity=0.25*⟩ to zero.

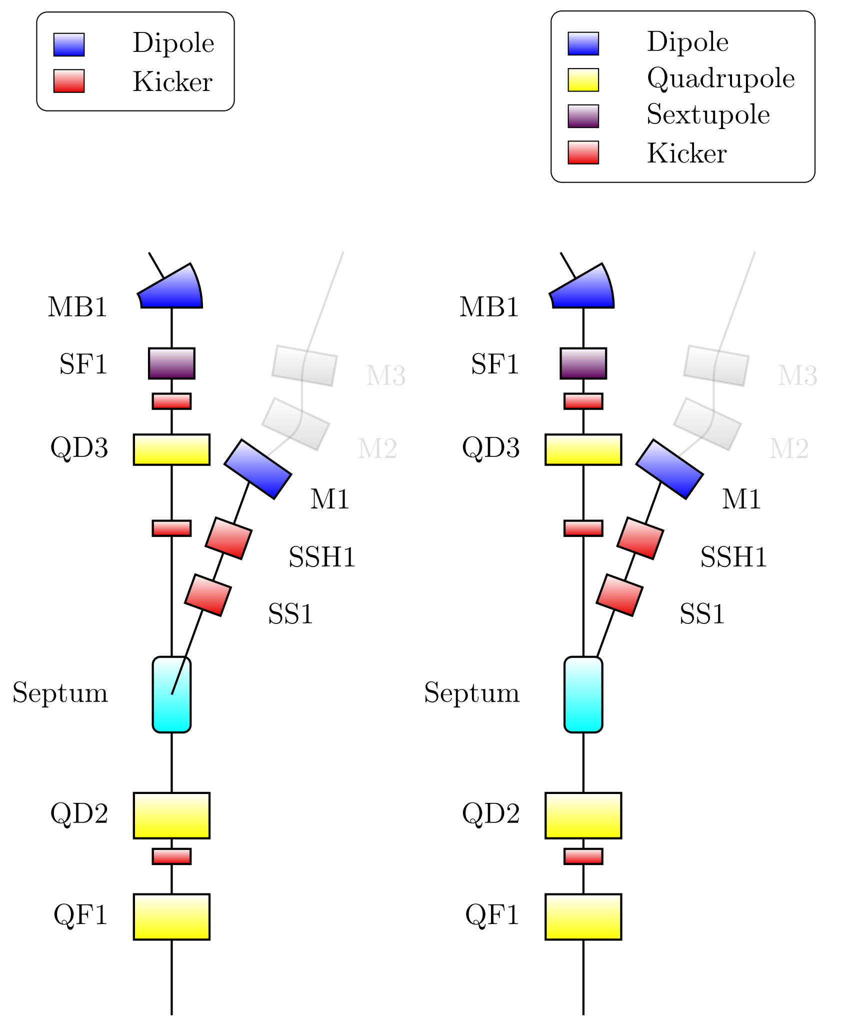
4.5 Access lattice Coordinates

You can use element coordinates to draw anything you want using pgf/tikz.

```
\savecoordinate{⟨name⟩}[⟨position=east⟩]
```

saves the coordinate of the previous element to access it later. Position specifies the exact place of the element. East and center are available. East is always downstream.

- you can use all tikz/pgf commands within the lattice environment to draw anything.
- You can use this to connect multiple beamlines **within a lattice environment** with \goto{name} (recommended, Figure 4 (b)).
- You can use this to connect **multiple lattice environments** with \start{name}. Use the tikz overlay option. (Figure 4 (a))



(a) two lattice environments and start (b) one lattice environment and goto

Figure 4: From example 3: Two ways to connect lattices

5 TODO

What is missing?

- manually adding and editing legend entries
- The look of the elements can be improved
- More element types can be added
- ...

I am not an TEX programmer. I basically used the Ti*k*Z commands and wrote a bunch of macros. So there is:

- no error handling implemented
- no dedicated scoping of internal macros (use of lattice with documentclass standalone recommended)

Known issues:

- Legend entry alignment is buggy for some legend scales that differ significantly from 1
- Please report bugs!