1 Installation

Follow the steps in the `INSTALL` file.
In the directory of the `CMakeLists.txt` do:

Under Linux/Mac OS:

```
$ mkdir build
$ cd build
$ cmake ..
$ sudo make install
```

Under Windows:
- open a terminal with administrator rights
- do the same as above but without the `sudo`

This installation requires CMake (`https://cmake.org/download/`).

2 Commands

2.1 grid

Definition:

```
void grid( const bool on = true,
           const std::string gridType = "-",
           const std::string gridCol = "h" )
```

Restricitons: None.

2.2 xlabel

Definition:

```
void xlabel( const char* label,
             const double pos = 0 )
```

Restricitons: If `setlog` is called `xlabel` should be called beforehand.
Examples:

```
Figure fig;
fig.plot(x, y, "+g"); // '+g' equals matlab/python '+-g'
fig.xlabel("Linear x axis");
fig.save("plot.eps");

Figure fig;
fig.setlog(true, true);
fig.plot(x, y, "+g");
fig.xlabel("Logarithmic x axis"); // Not good - should be called before 'setlog'
fig.save("plot.eps");

Figure fig;
fig.xlabel("Logarithmix x axis"); // Good
fig.setlog(true, true);
```

```
fig.plot(x, y, "+g");
fig.save("plot.eps");
```

2.3 ylabel

Definition:

```
void ylabel( const char* label,
             const double pos = 0 )
```

Restricitons: If `setlog` is called `ylabel` should be called beforehand.
Examples: See `xlabel`.

2.4 legend

Definition:

```
void legend( const double xPos = 1,
             const double yPos = 1 )
```

Restricitons: None.

2.5 setlog

Definition:

```
void setlog( const bool logx = true,
             const bool logy = true )
```

Restricitons: All plots will use the latest `setlog` options or default if none have been set.
Examples:

```
Figure fig;
fig.setlog(true, false); // -> semilogx
fig.plot(x0, y0, "b");
fig.setlog(false, true); // -> semilogy
fig.plot(x1, y1, "r");
fig.setlog(true, true); // -> loglog
fig.plot(x2, y2, "g");
fig.save("plot.eps"); // ATTENTION: all plots will have been plotted in loglog-scale

Figure fig;
fig.plot(x, y, "b");
fig.save("plot.eps"); // -> default (= linear) scaling
```

2.6 plot

Definition:

```
void plot( const std::vector<double> x,
           const std::vector<double> y,
           const char* style,
           const char* legend = 0 )

void plot( const Eigen::VectorXd x,
```

```
            const Eigen::VectorXd y,
            const char* style,
            const char* legend = 0 )
```

Restricitons: `x` and `y` must have same length. Also note that the `style`-argument is required!
Examples:

```
Figure fig;
fig.plot(x, y, "b");
fig.save("data.eps");

Figure fig;
fig.plot(x, y); // Not OK - style missing
fig.save("data.eps");

Figure fig;
fig.plot(x, y, " *r", "Data w/ red dots"); // ' *r' equals matlab/python 'r*'
fig.save("data.eps");
```

2.7 ranges

Definition:

```
void ranges( const double xMin,
             const double xMax,
             const double yMin,
             const double yMax )
```

Restricitons: xMin $<$ xMax, yMin $<$ yMax and ranges must be > 0 for axis in logarithmic scale.
Examples:

```
Figure fig;
fig.ranges(-1,1,-1,1);
fig.plot(x, y, "b");

Figure fig;
fig.plot(x, y, "b");
fig.ranges(0, 2.3, 4, 5); // ranges can be called before or after 'plot'
```

2.8 save

Definition:

```
void save( const char* file )
```

Restrictions: The filename *must* end on `.eps`!
Examples:

```
Figure fig;
fig.save("plot.eps"); // OK

Figure fig;
fig.save("plot.png"); // Not OK - Only eps-format supported!
```

2.9 title

Definition:

```
void title( const char* text )
```

3 Line characteristics

Linecolors[a]:

blue	b
green	g
red	r
cyan	c
magenta	m
yellow	y
gray	h
green-blue	l
sky-blue	n
orange	q
green-yellow	e
blue-violet	u
purple	p

[a] Upper-case letters will give a darker version of the lower-case version.

Linestyles:

solid	-
dashed	;
small dashed	=
long dashed	|
dotted	:
dash-dotted	j
small dash-dotted	i

Linemarkers:

+	+
o	o
◇	d
·	.
△	^
▽	v
◁	<
▷	>
⊙	#.
⊞	#+
⊠	#x