A X O D R A W

J.A.M.Vermaseren

NIKHEF-H
P.O. Box 41882
1009 DB Amsterdam

Abstract

Axodraw is a set of drawing primitives for use in LaTeX. These can be used for the drawing of Feynman diagrams, flow charts and simple graphics. Because it uses postscript for its drawing commands it works only in combination with the dvips of Radical Eye Software which is presently the most popular dvips program. More will be added in the future. It allows whole articles including their pictures to be contained in a single file, thereby making it easier to exchange the article file by e-mail. The current version[1] supports color according to the scheme implemented in the file colordvi.sty which comes with most TeX distributions.

[1] An earlier version of Axodraw was published in Comp. Phys. Comm. 83 (1994) 45.

1 Using Axodraw

The file axodraw.sty is a style file for LaTeX. It should be included in the documentstyle statement at the beginning of the document. An example would be:

```
\documentstyle[a4,11pt,axodraw]{article}
```

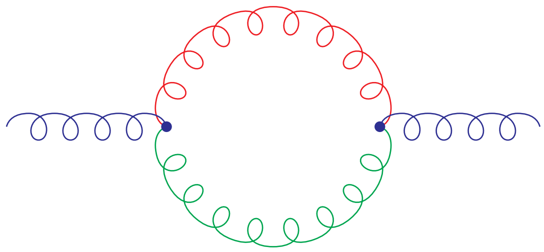
Because axodraw.sty reads also the epsf.sty file that comes with many implementations of TeX and in particular those that rely on the dvips program by Radical Eye Software for the printing, this file should be present in the system. If this file is not available one should obtain it from another system. The file colordvi.sty is also read, but if it is not present there will be no error. The user should just not use color in that case. The author feels in no way responsible for the problems that may occur when a different dvi-to-postscript program is used.

The drawing is actually done in postscript. Because the above mentioned dvi-to-postscript converter allows the inclusion of postscript code the graphics primitives have been included in the file axodraw.sty in terms of postscript. If another postscript converter is used, one may have to adapt the syntax of the inclusion of this code to the local system.

The commands of Axodraw should be executed inside either the picture or the figure environment. Inside this environment it is possible to place objects at arbitrary positions and put text between them. In principle one could try to draw objects with the facilities of LaTeX itself, but it turns out that the commands in the picture environment are not very powerful. Axodraw gives good extensions of them. An example would be

```
\begin{center} \begin{picture}(300,100)(0,0)
\SetColor{Red}
\GlueArc(150,50)(40,0,180){5}{8}
\SetColor{Green}
\GlueArc(150,50)(40,180,360){5}{8}
\SetColor{Blue}
\Gluon(50,50)(110,50){5}{4}  \Vertex(110,50){2}
\Gluon(190,50)(250,50){5}{4} \Vertex(190,50){2}
\end{picture}  \\ {\sl A gluon loop diagram} \end{center}
```

This code would result in:



A gluon loop diagram

The syntax and the meaning of these command are explained in the next section. One should note that all coordinates are presented in units of 1 point. There are 72 points in an inch. It is possible to use scale transformations if these units are not convenient.

Currently the primitives are mainly useful for the drawing of Feynman diagrams and the drawing of flowcharts. This means that the commands were designed to draw a number of these graphs. Of course many more things can be drawn with them, like scatter plots, histograms etc.

The current manual uses only those color commands that are safe on systems that do not have the required colordvi.sty file. It should however be clear from the examples how to use the other features. To allow the creation of complicated color commands that will work also in the absence of the colordvi.sty file there is a macro IfColor which is described below with the other commands.

2 The commands

The commands that are currently available in Axodraw are (in alphabetic order):

- \ArrowArc(x,y)(r,ϕ_1,ϕ_2)
  Draws an arc segment centered around (x,y). The radius is r. The arc-segment runs counterclockwise from ϕ_1 to ϕ_2. All angles are given in degrees. In the middle of the segment there will be an arrow.

- \ArrowArcn(x,y)(radius,ϕ_1,ϕ_2)
  Draws an arc segment centered around (x,y). The radius is r. The arc-segment runs clockwise from ϕ_1 to ϕ_2. All angles are given in degrees. In the middle of the segment there will be an arrow.

- \ArrowLine(x_1,y_1)(x_2,y_2)
  Draws a line from (x_1,y_1) to (x_2,y_2). There will be an arrow in the middle of the line.

- \BBox(x_1,y_1)(x_2,y_2)
  Draws a box of which the contents are blanked out. This means that anything that was present at the position of the box will be overwritten. The lower left corner of the box is at (x_1,y_1) and (x_2,y_2) is the upper right corner of the box.

- \BBoxc(x,y)(width,height)
  Draws a box of which the contents are blanked out. This means that anything that was present at the position of the box will be overwritten. The center of the box is at (x,y). Width and height refer to the full width and the full height of the box.

- \BCirc(x,y){r}
  Draws a circle of which the contents are blanked out. This means that anything that was present at the position of the circle will be overwritten. The center of the circle is at (x,y). r is its radius.

- \Boxc(x,y)(width,height)
  Draws a box. The center of the box is at (x,y). Width and height refer to the full width and the full height of the box.

- \BText(x,y){text}
  Draws a box with one line of centered postscript text in it. The box is just big enough to fit around the text. The coordinates refer to the center of the box. The box is like a BBox in that it blanks out whatever was at the position of the box.

- \B2Text(x,y){text1}{text2}
  Draws a box with two lines of centered postscript text in it. The box is just big enough to fit around the text. The coordinates refer to the center of the box. The box is like a BBox in that it blanks out whatever was at the position of the box.

- \CArc(x,y)(radius,ϕ_1,ϕ_2)
  Draws an arc segment centered around (x,y). The radius is r. The arc-segment runs counterclockwise from ϕ_1 to ϕ_2. All angles are given in degrees.

- \CBox(x_1,y_1)(x_2,y_2){color1}{color2}
  Draws a box. The lower left corner of the box is at (x_1,y_1) and (x_2,y_2) is the upper right corner of the box. The contents of the box are lost. The color of the box will be color1 and the color of the background inside the box will be color2.

- \CBoxc(x,y)(width,height){color1}{color2}
  Draws a box of which the contents are blanked out. This means that anything that was present at the position of the box will be overwritten. The center of the box is at (x,y). Width and height refer to the full width and the full height of the box. The color of the box will be color1 and the color of the background inside the box will be color2.

- \CCirc(x,y){radius}{color1}{color2}
  Draws a circle around (x,y) with radius r. The contents of the circle are lost. The color of the box will be color1 and the color of the background inside the box will be color2.

- \COval(x,y)(h,w)(ϕ){color1}{color2}
  Draws an oval with an internal color indicated by color2. The oval itself has the color color1. The center of the oval is given by (x,y). Its height is h, and the width is w. In addition the oval can be rotated counterclockwise over ϕ degrees. The oval overwrites anything that used to be in its position.

- \CText(x,y){color1}{color2}{text}
  Draws a box with one line of centered postscript text in it. The box is just big enough to fit around the text. The coordinates refer to the center of the box. The box is like a CBox in that it blanks out whatever was at

the position of the box. The color of the box and the text inside is color1 and the background inside has the color color2.

- \C2Text(x,y){color1}{color2}{text1}{text2}
  Draws a box with two lines of centered postscript text in it. The box is just big enough to fit around the text. The coordinates refer to the center of the box. The box is like a CBox in that it blanks out whatever was at the position of the box. The color of the box and the text inside is color1 and the background inside has the color color2.

- \Curve{$(x_1, y_1)(x_2, y_2)\cdots(x_n, y_n)$}
  Draws a curve through the given points. The x-values are supposed to be in ascending order. The curve is a combination of quadratic and third order segments and is continuous in its first and second derivatives.

- \DashArrowArc(x,y)(r,ϕ_1,ϕ_2){dashsize}
  Draws a dashed arc segment centered around (x,y). The radius is r. The arc-segment runs counterclockwise from ϕ_1 to ϕ_2. All angles are given in degrees. In the middle of the segment there will be an arrow. The size of the dashes is approximately equal to 'dashsize'.

- \DashArrowArcn(x,y)(radius,ϕ_1,ϕ_2){dashsize}
  Draws a dashed arc segment centered around (x,y). The radius is r. The arc-segment runs clockwise from ϕ_1 to ϕ_2. All angles are given in degrees. In the middle of the segment there will be an arrow. The size of the dashes is approximately equal to 'dashsize'.

- \DashArrowLine(x_1,y_1)(x_2,y_2){dashsize}
  Draws a line from (x_1,y_1) to (x_2,y_2) with a dashed pattern. The size of the black parts of the pattern is given by 'dashsize'. The alternating pieces have equal length. The size of the pattern is adjusted so that both the begin and the end are black. Halfway the line there is an arrow.

- \DashCArc(x,y)(radius,ϕ_1,ϕ_2){dashsize}
  Draws a dashed arc segment centered around (x,y). The radius is r. The arc-segment runs counterclockwise from ϕ_1 to ϕ_2. All angles are given in degrees. The size of the dashes is determined by 'dashsize'. This size is adjusted somewhat to make the result look nice.

- \DashCurve{$(x_1, y_1)(x_2, y_2)\cdots(x_n, y_n)$}{ dashsize}
  Draws a dashed curve through the given points. The x-values are supposed to be in ascending order. The curve is a combination of quadratic and third order segments. The size of the black parts and the white parts will be approximately 'dashsize' each. Some adjustment takes place to make the pattern come out right at the endpoints.

- \DashLine(x_1,y_1)(x_2,y_2){dashsize}
  Draws a line from (x_1,y_1) to (x_2,y_2) with a dashed pattern. The size of the black parts of the pattern is given by 'dashsize'. The alternating pieces

have equal length. The size of the pattern is adjusted so that both the begin and the end are black.

- \EBox(x_1,y_1)(x_2,y_2)
  Draws a box. The lower left corner of the box is at (x_1,y_1) and (x_2,y_2) is the upper right corner of the box.

- \IfColor{arg1}{arg2}
  If the file colordvi.sty is present the first argument will be executed. If this file is not present the second argument will be executed. For examples, see some of the figures. This command can also be used in the regular text of the LaTeX file.

- \GBox(x_1,y_1)(x_2,y_2){grayscale}
  Draws a box. The lower left corner of the box is at (x_1,y_1) and (x_2,y_2) is the upper right corner of the box. The contents of the box are lost. They are overwritten with a color gray that is indicated by the parameter 'grayscale'. This parameter can have values ranging from 0 (black) to 1 (white).

- \GBoxc(x,y)(width,height){grayscale}
  Draws a box. The center of the box is at (x,y). Width and height refer to the full width and the full height of the box. The contents of the box are lost. They are overwritten with a color gray that is indicated by the parameter 'grayscale'. This parameter can have values ranging from 0 (black) to 1 (white).

- \GCirc(x,y){radius}{grayscale}
  Draws a circle around (x,y) with radius r. The contents of the circle are lost. They are overwritten with a color gray that is indicated by the parameter 'grayscale'. This parameter can have values ranging from 0 (black) to 1 (white).

- \GlueArc(x,y)(r,ϕ_1,ϕ_2){amplitude}{windings}
  Draws a gluon on an arc-segment. The center of the arc is (x,y) and r is its radius. The arc segment runs counterclockwise from ϕ_1 to ϕ_2. The width of the gluon is twice 'amplitude', and the number of windings is given by the last parameter. Note that whether the curls are inside or outside can be influenced with the sign of the amplitude. When it is positive the curls are on the inside.

- \Gluon(x_1,y_1)(x_2,y_2){amplitude}{windings}
  Draws a gluon from (x_1,y_1) to (x_2,y_2). The width of the gluon will be twice the value of 'amplitude'. The number of windings is given by the last parameter. If this parameter is not an integer it will be rounded to an integer value. The side at which the windings lie is determined by the order of the two coordinates. Also a negative amplitude can change this side.

- \GOval(x,y)(h,w)(ϕ){grayscale} Draws an oval with an internal color indicated by grayscale. This parameter can have values ranging from 0 (black) to 1 (white). The center of the oval is given by (x,y). Its height is h, and the width is w. In addition the oval can be rotated counterclockwise over ϕ degrees. The oval overwrites anything that used to be in its position.

- \GText(x,y){grayscale}{text}
  Draws a gray box with one line of centered postscript text in it. The box is just big enough to fit around the text. The coordinates refer to the center of the box. The box is like a BBox in that it blanks out whatever was at the position of the box.

- \G2Text(x,y){grayscale}{text1}{text2}
  Draws a gray box with two lines of centered postscript text in it. The box is just big enough to fit around the text. The coordinates refer to the center of the box. The box is like a BBox in that it blanks out whatever was at the position of the box.

- \LinAxis(x_1,y_1)(x_2,y_2)(N_D,d,hashsize ,offset,width)
  This draws a line to be used as an axis in a graph. Along the axis are hash marks. Going from the first coordinate to the second, the hash marks are on the left side if 'hashsize', which is the size of the hash marks, is positive and on the right side if it is negative. N_D is the number of 'decades', indicated by fat hash marks, and d is the number of subdivisions inside each decade. The offset parameter tells to which subdivision the first coordinate corresponds. When it is zero, this coordinate corresponds to a fat mark of a decade. Because axes have their own width, this is indicated with the last parameter.

- \Line(x_1,y_1)(x_2,y_2)
  Draws a line from (x_1,y_1) to (x_2,y_2).

- \LogAxis(x_1,y_1)(x_2,y_2)(N_L,hashsize ,offset,width)
  This draws a line to be used as an axis in a graph. Along the axis are hash marks. Going from the first coordinate to the second, the hash marks are on the left side if 'hashsize', which is the size of the hash marks, is positive and on the right side if it is negative. N_L is the number of orders of magnitude, indicated by fat hash marks. The offset parameter tells to which integer subdivision the first coordinate corresponds. When it is zero, this coordinate corresponds to a fat mark, which is identical to when the value would have been 1. Because axes have their own width, this is indicated with the last parameter.

- \LongArrow(x_1,y_1)(x_2,y_2)
  Draws a line from (x_1,y_1) to (x_2,y_2). There will be an arrow at the end of the line.

- \LongArrowArc(x,y)(r,ϕ_1,ϕ_2)
  Draws an arc segment centered around (x,y). The radius is r. The arc-segment runs counterclockwise from ϕ_1 to ϕ_2. All angles are given in degrees. At the end of the segment there will be an arrow.

- \LongArrowArcn(x,y)(radius,ϕ_1,ϕ_2)
  Draws an arc segment centered around (x,y). The radius is r. The arc-segment runs clockwise from ϕ_1 to ϕ_2. All angles are given in degrees. At the end of the segment there will be an arrow.

- \Oval(x,y)(h,w)(ϕ) Draws an oval. The center of the oval is given by (x,y). Its height is h, and the width is w. In addition the oval can be rotated counterclockwise over ϕ degrees. The oval does not overwrite its contents.

- \Photon(x_1,y_1)(x_2,y_2){amplitude}{wiggles}
  Draws a photon from (x_1,y_1) to (x_2,y_2). The width of the photon will be twice the value of 'amplitude'. The number of wiggles is given by the last parameter. If twice this parameter is not an integer it will be rounded to an integer value. Whether the first wiggle starts up or down can be influenced with the sign of the amplitude.

- \PhotonArc(x,y)(r,ϕ_1,ϕ_2){amplitude}{wiggles}
  Draws a photon on an arc-segment. The center of the arc is (x,y) and r is its radius. The arc segment runs counterclockwise from ϕ_1 to ϕ_2. The width of the photon is twice 'amplitude', and the number of wiggles is given by the last parameter. Note that the sign of the amplitude influences whether the photon starts going outside (positive) or starts going inside (negative). If one likes the photon to reach both endpoints from the outside the number of wiggles should be an integer plus 0.5.

- \PText(x,y)(ϕ)[mode]{text}
  Places a postscript text. The focal point is (x,y). The text is the last parameter. The mode parameter tells how the text should be positioned with respect to the focal point. If this parameter is omitted the center of the text will correspond to the focal point. Other options are: l for having the left side correspond to the focal point, r for having the right side correspond to it, t for having the top at the focal point and b for the bottom. One may combine two letters as in [bl], as long as it makes sense. The parameter ϕ is a rotation angle. The text is written in the current postscript font. This font can be set with the SetPFont command.

- \rText(x,y)[mode][rotation]{text}
  Places a rotated text. The focal point is (x,y). The text is the last parameter. If the rotation parameter is the character l the text will be rotated left by 90 degrees, if it is an r it will be rotated to the right by 90 degrees and when it is the character u the text will be rotated by 180 degrees. When there is no character there is no rotation and the command is identical to the Text command. The mode parameter tells

how the resulting box should be positioned with respect to the focal point. If this parameter is omitted the center of the box will correspond to the focal point. Other options are: l for having the left side correspond to the focal point, r for having the right side correspond to it, t for having the top at the focal point and b for the bottom. One may combine two letters as in [bl], as long as it makes sense.

- \SetColor{NameOfColor}
  Sets the color for the next commands. This command onlt affects the current picture. In addition it does not affect the text commands that write in TEX mode. Also the commands that draw gray boxes are not affected. For influencing the color of the TEX or LATEX output one can use the commands mentioned in the colordvi.sty file.

- \SetPFont{fontname}{fontsize}
  Sets the postscript font to a given type and scale.

- \SetScale{scalevalue}
  Changes the scale of all graphics operations. Unfortunately it does not change the scale of the text operations (yet?). A 'scalevalue' of 1 is the default. It is allowed to use floating point values.

- \SetOffset(x_offset,y_offset)
  Adds the offset values to all coordinates at the TEX level. This makes it easier to move figures around.

- \SetScaledOffset(x_offset,y_offset)
  Adds the offset values to all coordinates at the postscript level. This is done after scaling has been applied. Hence one can work with the scaled coordinates. This can be very handy when drawing curves.

- \SetWidth{widthvalue}
  Changes the linewidth in all graphics operations. It does not change the linewidth of the text operations. That is a matter of font selection. A 'widthvalue' of 0.5 is the default. It is allowed to use floating point values.

- \Text(x,y)[mode]{text}
  Places a text. The focal point is (x,y). The text is the last parameter. The mode parameter tells how the text should be positioned with respect to the focal point. If this parameter is omitted the center of the text will correspond to the focal point. Other options are: l for having the left side correspond to the focal point, r for having the right side correspond to it, t for having the top at the focal point and b for the bottom. One may combine two letters as in [bl], as long as it makes sense.

- \Vertex(x,y){r}
  Draws a fat dot at (x,y). The radius of the dot is given by r.

- \ZigZag(x_1,y_1)(x_2,y_2){amplitude}{wiggles}
  Draws a zigzag line from (x_1,y_1) to (x_2,y_2). The width of the zigzagging

will be twice the value of 'amplitude'. The number of zigzags is given by the last parameter. If twice this parameter is not an integer it will be rounded to an integer value. Whether the first zigzag starts up or down can be influenced with the sign of the amplitude.

A note about color. The names of the colors can be found in the local file colordvi.sty or colordvi.tex. This file gives also the commands that allow the user to change the color of the text.

3 Examples

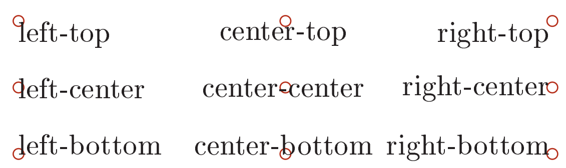
Although the previous section contains all the commands and their proper syntax a few examples may be helpful.

3.1 Text modes

The meaning of the mode characters in the text commands can best be demonstrated. The statements

```
\begin{center}  \begin{picture}(300,100)(0,0)
\SetColor{BrickRed}
\CArc(50,75)(2,0,360)  \Text(50,75)[lt]{left-top}
\CArc(50,50)(2,0,360)  \Text(50,50)[l]{left-center}
\CArc(50,25)(2,0,360)  \Text(50,25)[lb]{left-bottom}
\CArc(150,75)(2,0,360) \Text(150,75)[t]{center-top}
\CArc(150,50)(2,0,360) \Text(150,50)[]{center-center}
\CArc(150,25)(2,0,360) \Text(150,25)[b]{center-bottom}
\CArc(250,75)(2,0,360) \Text(250,75)[rt]{right-top}
\CArc(250,50)(2,0,360) \Text(250,50)[r]{right-center}
\CArc(250,25)(2,0,360) \Text(250,25)[rb]{right-bottom}
\end{picture}   \end{center}
```

produce 9 texts and for each the focal point is indicated by a little circle. It looks like



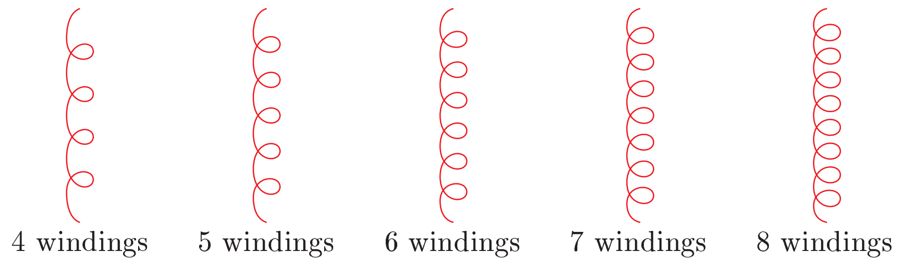
This illustrates exactly all the combinations of the mode characters and what their effects are. The commands \Text and \rText give a text according to LaTeX. This text is insensitive to the scaling commands, and the color of the text should be set with the regular color commands given in colordvi.sty. The text in the \PText command (and the various boxes with text) is a postscript text. Such text is sensitive to the scaling commands and in addition the color is set with the \SetColor command or in the command itself (in the case of the boxes). In the case of LaTeX text it can of course contain different fonts, math mode and all those little things that are usually easier in LaTeX than in postscript.

3.2 The windings of a gluon

Gluons are traditionally represented by a two dimensional projection of a helix. Actually close inspection of some pretty gluons reveals that it is usually not quite a helix. Hence the gluons in Axodraw are also not quite helices. In addition one may notice that the begin and end points deviate slightly from the regular windings. This makes it more in agreement with hand drawn gluons. When a gluon is drawn, one needs not only its begin and end points but there is an amplitude connected to this almost helix, and in addition there are windings. The number of windings is the number of curls that the gluon will have. Different people may prefer different densities of curls. This can effect the appearance considerably:

```
\begin{center}
\begin{picture}(330,100)(0,0)
\SetColor{Red}
\Gluon(25,15)(25,95){5}{4}      \Text(25,7)[]{4 windings}
\Gluon(95,15)(95,95){5}{5}      \Text(95,7)[]{5 windings}
\Gluon(165,15)(165,95){5}{6}    \Text(165,7)[]{6 windings}
\Gluon(235,15)(235,95){5}{7}    \Text(235,7)[]{7 windings}
\Gluon(305,15)(305,95){5}{8}    \Text(305,7)[]{8 windings}
\end{picture}
\end{center}
```

This code results in:

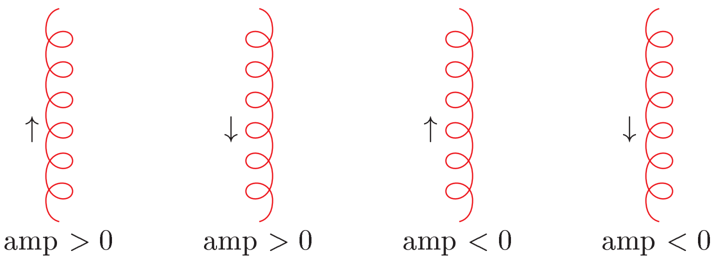


The influence of the amplitude is also rather great. The user should experiment with it. There is however an aspect to the amplitude that should be discussed. For a straight gluon the amplitude can determine on which side the curls are. So does the direction of the gluon:

```
\begin{center}
\begin{picture}(325,100)(0,0)
\SetColor{Red}
\Gluon(50,15)(50,95){5}{6}
```

```
\Text(50,7)[]{amp $> 0$}  \Text(40,50)[]{$\uparrow$}
\Gluon(125,95)(125,15){5}{6}
\Text(125,7)[]{amp $> 0$} \Text(115,50)[]{$\downarrow$}
\Gluon(200,15)(200,95){-5}{6}
\Text(200,7)[]{amp $< 0$} \Text(190,50)[]{$\uparrow$}
\Gluon(275,95)(275,15){-5}{6}
\Text(275,7)[]{amp $< 0$} \Text(265,50)[]{$\downarrow$}
\end{picture}
\end{center}
```

The picture gets the following appearance:



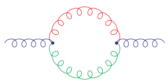
For straight gluons one does not need the option of the negative amplitude. It is however necessary for gluons on an arc segment. In that case the arc is always drawn in an anticlockwise direction. Hence the direction is fixed and only the amplitude is left as a tool for determining the side with the curls.

3.3 Scaling

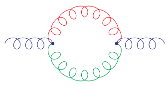
Sometimes it is much easier to design a figure on a larger scale than it is needed in the eventual printing. In that case one can use a scale factor, either during the design or in the final result. We use the figure in the first section as an example:

```
\vspace{-10pt} \hfill \\
\SetScale{0.3}
\begin{picture}(70,30)(0,13)
\SetColor{Red}
\GlueArc(120,50)(40,0,180){5}{8}
\SetColor{Green}
\GlueArc(120,50)(40,180,360){5}{8}
\SetColor{Blue}
\Gluon(20,50)(80,50){5}{4}  \Vertex(80,50){2}
\Gluon(160,50)(220,50){5}{4} \Vertex(160,50){2}
\end{picture} $+$ others
        $ = C_A(\frac{5}{3}+\frac{31}{9}\epsilon)
          + n_F(-\frac{2}{3}-\frac{10}{9}\epsilon)$
\vspace{10pt}  \hfill \\
```

We have lowered the figure by 13 points (the (0,13) in the picture statement) to make it look nice with respect to the equal sign. The result is

 $+ \text{ others} = C_A(\frac{5}{3} + \frac{31}{9}\epsilon) + n_F(-\frac{2}{3} - \frac{10}{9}\epsilon)$

This way it is rather straightforward to make whole pictorial equations. Of course some things are not scale invariant. The appreciation of a figure may be somewhat different when the scale is changed. In the above case one might consider changing the amplitude of the gluons a little bit. Changing this from 5 to 7 and at the same time reducing the number of windings from 4 to 3 for the straight gluons and from 8 to 7 for the gluons in the arcs gives

 $+ \text{ others} = C_A(\frac{5}{3} + \frac{31}{9}\epsilon) + n_F(-\frac{2}{3} - \frac{10}{9}\epsilon)$

At this scale this may please the eye more.

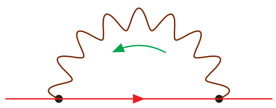
There is one problem with scaling. Currently it is only possible to have text scale with the rest of a figure when the text has been printed with the PText command. This makes the typesetting more complicated, but the scaling of the TEX pixel fonts would give rather poor results anyway.

3.4 Photons

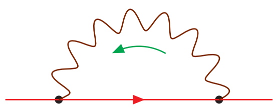
When drawing photons one should take care that the number of wiggles is selected properly. Very often this number should be an integer plus 0.5. This can be seen in the following example:

```
\begin{center}\begin{picture}(300,56)(0,0)
\Vertex(180,10){1.5} \Vertex(120,10){1.5}
\SetColor{Red}
\ArrowLine(100,10)(200,10)
\SetColor{Green}
\LongArrowArc(150,10)(20,60,120)
\SetColor{Brown}
\PhotonArc(150,10)(30,0,180){4}{8.5}   % 8.5 wiggles
\end{picture} \end{center}
```

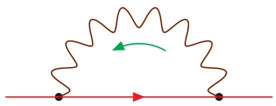
This gives the 'proper' picture as it would usually drawn by hand:



When the number of wiggles is reduced to 8 we obtain:



This is not as nice. Somehow the symmetry is violated. One should also take care that the wiggles start in the proper way. If we make the amplitude negative we see that the photons are not ‘right’ either:



Sometimes these things require some experimenting.

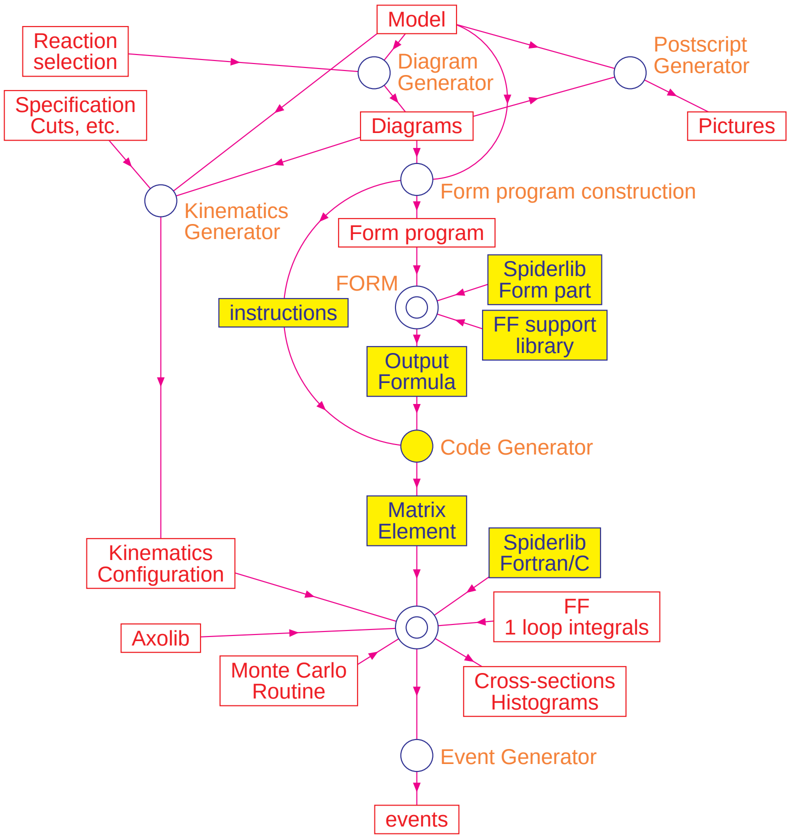
3.5 Flowcharts

There are several commands for creating boxes with text in them. This can be a box with either one line of text or with two lines of text. The rest is just a matter of drawing lines and circle segments with arrows. If the text is to scale with the picture one needs to use the postscript fonts. The result of scaling the TEX fonts is usually rather ugly, because these fonts are pixel fonts. Here we present an example. It might describe a system for the automatic computation of cross-sections:

```
\begin{center} \begin{picture}(320,320)(0,0)
\SetPFont{Helvetica}{10}
\SetScale{0.8}
\SetColor{Magenta}
\ArrowLine(200,40)(200,10)   \ArrowLine(200,100)(200,40)
\ArrowLine(200,150)(200,100) \ArrowLine(100,130)(200,100)
\ArrowLine(85,95)(200,100)   \ArrowLine(260,105)(200,100)
\ArrowLine(250,135)(200,100) \ArrowLine(160,75)(200,100)
\ArrowLine(200,100)(250,70)  \ArrowLine(200,185)(200,150)
\ArrowLine(200,220)(200,185) \ArrowLine(200,250)(200,220)
\ArrowLine(240,263)(200,250) \ArrowLine(240,237)(200,250)
\ArrowLine(200,285)(200,250) \ArrowLine(200,310)(200,285)
\ArrowLine(200,335)(200,310) \ArrowLine(180,360)(200,335)
\ArrowLine(200,385)(180,360) \ArrowLine(50,370)(180,360)
\ArrowArc(200,247.5)(62.5,90,180)
\ArrowArc(200,247.5)(62.5,180,270)
\ArrowLine(210,385)(300,360) \ArrowLine(210,335)(300,360)
\ArrowLine(80,300)(80,130)   \ArrowLine(190,335)(80,300)
\ArrowLine(190,385)(80,300)  \ArrowLine(50,335)(80,300)
\ArrowLine(300,360)(340,340) \ArrowArcn(205,347.5)(37.5,90,270)
\SetColor{Blue}
\BCirc(200,100){10}  \BCirc(200,100){5}
\BCirc(200,40){7.5}  \BCirc(200,250){10}
\BCirc(200,250){5}   \BCirc(200,310){7.5}
\BCirc(180,360){7.5} \BCirc(80,300){7.5}
\BCirc(300,360){7.5}
\IfColor{\CCirc(200,185){7.5}{Blue}{Yellow}
```

```
}{\GCirc(200,185){7.5}{0.9}}
\SetColor{Red}
\BText(200,285){Form program} \BText(200,335){Diagrams}
\BText(200,385){Model}        \BText(200,10){events}
\BText(80,95){Axolib}         \BText(350,335){Pictures}
\IfColor{\CText(137.5,247.5){Blue}{Yellow}{instructions}
}{\GText(137.5,247.5){0.9}{instructions}}
\B2Text(260,70){Cross-sections}{Histograms}
\B2Text(140,75){Monte Carlo}{Routine}
\B2Text(275,105){FF}{1 loop integrals}
\IfColor{\C2Text(260,135){Blue}{Yellow}{Spiderlib}{Fortran/C}
}{\G2Text(260,135){0.9}{Spiderlib}{Fortran/C}}
\IfColor{\C2Text(200,150){Blue}{Yellow}{Matrix}{Element}
}{\G2Text(200,150){0.9}{Matrix}{Element}}
\B2Text(80,130){Kinematics}{Configuration}
\IfColor{\C2Text(200,220){Blue}{Yellow}{Output}{Formula}
}{\G2Text(200,220){0.9}{Output}{Formula}}
\IfColor{\C2Text(260,263){Blue}{Yellow}{Spiderlib}{Form part}
}{\G2Text(260,263){0.9}{Spiderlib}{Form part}}
\IfColor{\C2Text(260,237){Blue}{Yellow}{FF support}{library}
}{\G2Text(260,237){0.9}{FF support}{library}}
\B2Text(40,370){Reaction}{selection}
\B2Text(40,340){Specification}{Cuts, etc.}
\SetColor{Orange}
\PText(211,36)(0)[lb]{Event Generator}
\PText(211,181)(0)[lb]{Code Generator}
\PText(162,258)(0)[lb]{FORM}
\PText(211,301)(0)[lb]{Form program construction}
\PText(191,362)(0)[lb]{Diagram}
\PText(191,352)(0)[lb]{Generator}
\PText(311,370)(0)[lb]{Postscript}
\PText(311,360)(0)[lb]{Generator}
\PText(91,292)(0)[lb]{Kinematics}
\PText(91,282)(0)[lb]{Generator}
\end{picture} \end{center}
```

This gives the chart



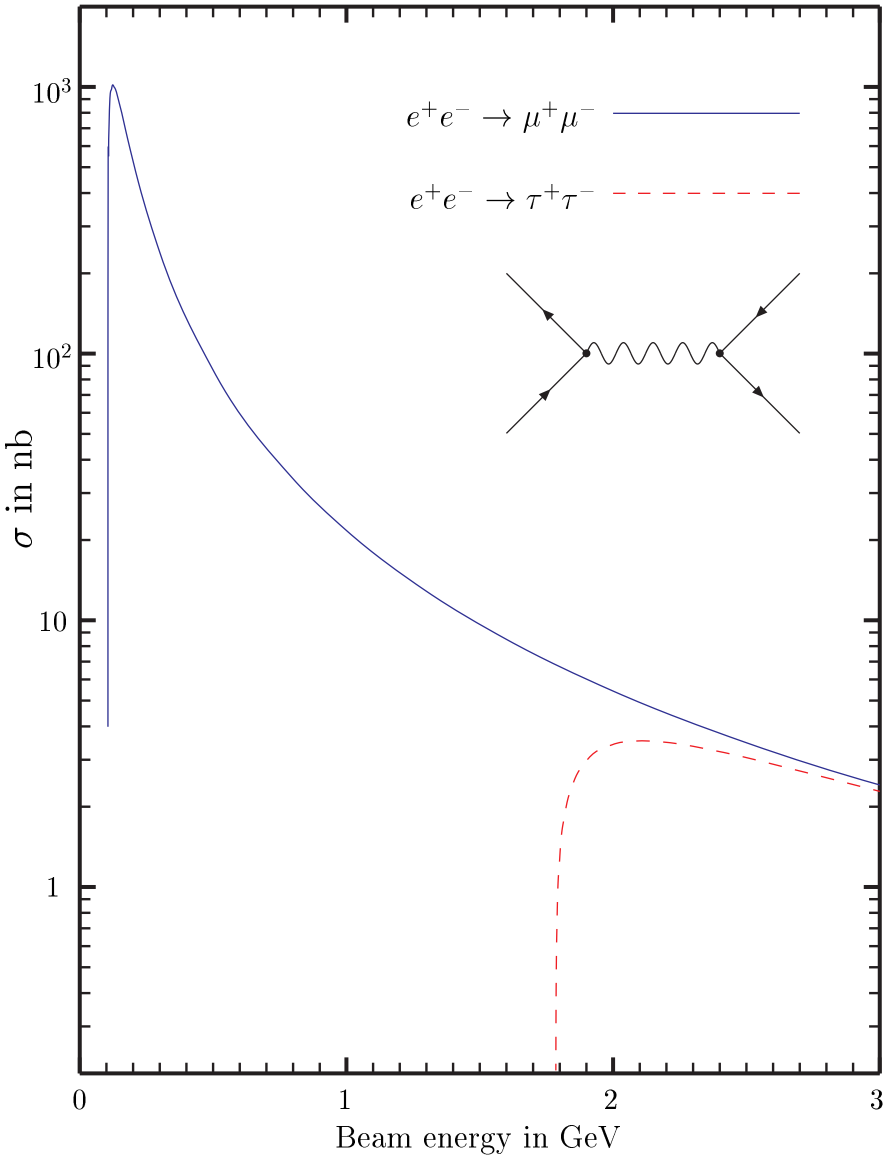
3.6 Curves and graphs

Axodraw is equipped with a curve fitting facility that can draw smooth curves through a set of coordinates. Coupled to this is a set of commands to draw the axes that are typically needed for the use of graphs and histograms. An example of a complete picture would be

```
\begin{center} \begin{picture}(360,440)(0,0)
\SetOffset(40,30)
\LinAxis(0,0)(300,0)(3,10,5,0,1.5)
\LinAxis(0,400)(300,400)(3,10,-5,0,1.5)
\LogAxis(0,0)(0,400)(4,-5,2,1.5)
\LogAxis(300,0)(300,400)(4,5,2,1.5)
\SetScale{100.} \SetWidth{0.005}
\SetColor{Blue}
\Curve{(.1057001,1.2997)(.1057003,1.5399)
(.1057006,1.6908)(.1057010,1.8019)(.1057030,2.0406)
(.1057060,2.1911)(.1057100,2.3020)(.1057300,2.5403)
(.1057600,2.6904)(.1058000,2.8007)(.1060000,3.0365)
(.1080000,3.4512)(.1100000,3.5600)(.1200000,3.6950)
(.1300000,3.6969)(.1500000,3.6308)(.1800000,3.5024)
(.2200000,3.3413)(.3000000,3.0788)(.5000000,2.6374)
(.8000000,2.2295)(1.0000000,2.0357)(1.3000000
```

```
,1.8078)(1.6000000,1.6275)(2.0000000,1.4336)
(2.5000000,1.2398)(3.0000000,1.0815)}
\SetColor{Red}
\DashCurve{(1.7853600,.0111)(1.7853800,.0228)
(1.7854000,.0339)(1.7856000,.1218)(1.7860000,.2324)
(1.7870000,.3821)(1.7900000,.5786)(1.8000000,.8089)
(1.8200000,.9765)(1.8500000,1.0869)(1.9000000
,1.1718)(2.0000000,1.2335)(2.1000000,1.2468)
(2.2000000,1.2413)(2.4000000,1.2064)
(2.7000000,1.1340)(3.0000000,1.0574)}{0.05}
\SetScale{1.}\SetWidth{0.5}
\SetColor{Blue}
\Line(200,360)(270,360)
\Text(195,360)[r]{\large$e^+e^-\rightarrow\mu^+\mu^-$}
\SetColor{Red}
\DashLine(200,330)(270,330){5}
\Text(195,330)[r]{\large$e^+e^-\rightarrow\tau^+\tau^-$}
\SetColor{Black}
\Text(0,-10)[]{0} \Text(100,-10)[]{1}
\Text(200,-10)[]{2} \Text(300,-10)[]{3}
\Text(150,-25)[]{\large Beam energy in GeV}
\Text(-10,70)[]{$1$} \Text(-10,170)[]{$10$}
\Text(-10,270)[]{$10^2$} \Text(-10,370)[]{$10^3$}
\rText(-25,220)[][l]{\Large$\sigma$ in nb}
\ArrowLine(190,270)(160,300)
\ArrowLine(160,240)(190,270)
\ArrowLine(270,300)(240,270)
\ArrowLine(240,270)(270,240)
\Photon(190,270)(240,270){4}{4.5}
\Vertex(190,270){1.5} \Vertex(240,270){1.5}
\end{picture}  \\ {\sl \hskip 10 pt Threshold
effects for $\mu$ and $\tau$} \end{center}
```

and the resulting picture would be



Threshold effects for μ and τ

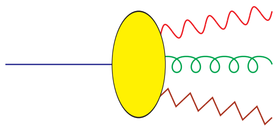
Of course one can scale these pictures further, but because the scale factor has been used to enter the data points these should then be adapted too. Note that when the scale is blown up by a factor 100, the linewidth has to be scaled down or disasters will take place.

Finally a playful example:

```
\begin{center}\begin{picture}(300,56)(0,0)
\SetColor{Blue}
\Line(100,25)(150,25)
\SetColor{Green}
\Gluon(150,25)(200,25){3}{6}
\SetColor{Red}
\Photon(150,35)(200,45){3}{6}
```

```
\SetColor{Mahogany}
\ZigZag(150,15)(200,5){3}{6}
\IfColor{\COval(150,25)(20,10)(0){Black}{Yellow}
     }{\GOval(150,25)(20,10)(0){0.5}}
\end{picture} \end{center}
```

which results in



Acknowledgement: The author wishes to thank G.J.van Oldenborgh for help with some of the TEX macros.

Axodraw can be obtained from the authors homepage: http://www.nikhef.nl/∼t68/axodraw. Alternatively it is available by means of anonymous ftp from ftp.nikhef.nl. There it is located in the directory pub/form/axodraw. Commentary and suggestions should be sent to the author at t68@nikhef.nl.