Feynman.sty: Feynman diagrams in postscript

François Gelis

December 2, 2002

Service de Physique Théorique,
CEA/DSM/Saclay,
91191, Gif-sur-Yvette Cedex, France

Abstract

This package defines an environment that facilitates the insertion of raw postscript commands within a LaTeX document. It comes with a set of postscript macros that are designed to draw Feynman diagrams. Some elementary knowledge of postscript is needed in order to use this package.

1 Introduction

If you have already written papers about particle physics, you have presumably encountered this problem: how to draw and include Feynman diagrams in a LaTeX document ? The package **feynman.sty** described in this document is one of the possibilities for doing that (but not only that).

In fact, **feynman.sty** allows you to insert postscript commands directly in your LaTeX document. Its use assumes that (i) you know the basics of postscript and (ii) that you know what you are doing. Its principal advantages over alternatives is that the diagrams are embedded in the LaTeX document itself: if you are writing a large document with dozens of Feynman diagrams, you don't need a separate file for each of them, as you would need with conventional postscript figures.

In addition, it provides a set of postscript macros for the main types of propagators one can possibly encounter in particle physics. These macros are the building blocks for complicated diagrams. They can also be used in stand-alone postscript files.

2 Introductory example

Here is a simple example of use of this package. Prepare a LaTeX document containing the following:

```
\documentclass[11pt]{article}
\usepackage[final]{graphics}
\usepackage{feynman}
\begin{document}

\begin{figure}[ht]
\centerline{
\begin{feynman}{3cm}
\diagram{30}{-2 1 translate
  0 0 1 0 beglphoton
  1 -180 0 begcfermion
  1 180 endlphoton
  1 0 180 begcfermion}
\end{feynman}
}
\caption{A simple example}
\end{figure}

\end{document}
```

The figure included in this document will look as follows:

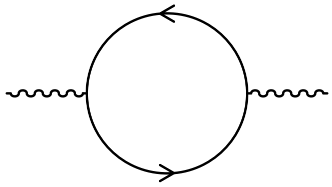

Figure 1: A simple example

As you can see from this example, the document must include the package **feynman** in the header. This means that the file **feynman.sty** must be present in the current directory, or in the search path of latex.

The environment called "feynman" takes a vertical size as argument, and simply reserves some empty vertical space equal to that length. The macro "diagram" takes a first argument which is the scaling to apply before the postscript commands that come next[1]. The second argument of the macro **diagram** is a sequence of postscript commands, that will be included verbatim in the postscript generated by **dvips**. Typically, we will start with a translation command (here: **-2 1 translate**), which is adjusted in order to center the diagram in the figure environment. The following postscript commands of this example make use of the predefined macros in order to draw photon and fermion propagators. As one can see, these macros hide all the intricacy of drawing the propagators. Only s few parameters that describe the size and endpoints of the propagators are necessary. The list of the available macros, as well as their syntax will be described later in this document.

3 Installation

This package is made of two files **feynman.sty** and **feynman.pro**, in addition to the present documentation file. The file **feynman.sty** is needed when LaTeX is run on the file containing Feynman diagrams, and must be in the search path of the LaTeX executable (that can be the current directory). The file **feynman.pro** is a postscript header containing the definitions of all the postscript macros provided by this package. It can be used both by a viewer like **xdvi** (Note that only recent versions of **xdvi** are able to take advantage of the "header=..." special statement, and to display the Feynman diagrams contained in a document. The version shipped with the TeTeX-0.9 is known to work.), and by postscript generators like **dvips**. It must be in the search path of these executables.

[1]Postscript natural unit id the *point*, equal to 1/72th of an inch. The postscript language provides the command **scale** in order to change this unit. For instance, the command **30 30 scale** multiplies the default scale by a factor 30 both on the x and the y axis (NOTE: a scaling factor of 30 makes the unit length approximately equal to a centimeter).

4 Macros naming convention and syntax

4.1 Naming convention

The file **feynman.pro** contains predefined macros for many ordinary types of propagators: scalar, fermions, photons, gluons and ghosts. These propagators can either be linear or circular. Finally, these propagators can either start or end at currentpoint. The name of a macro drawing a propagator obeys the following construction:

$$\left\{\begin{array}{c}\text{beg}\\ \text{end}\end{array}\right\}\left\{\begin{array}{c}\text{l}\\ \text{c}\end{array}\right\}\{\text{name}\}\ ,$$

where "name" is one of "scalar", "fermion", "photon", "gluon" or "ghost"[2]. The prefix "beg" or "end" indicates whether we want a propagator that begins or ends at a given point. The intermediate letter "l" or "c" indicates whether we want a linear or a circular propagator. That makes 20 different macros.

4.2 Macros syntax

4.2.1 Macros of type "l"

The macros of type "l" need four arguments present in the stack prior to being called, and return two numbers. The syntax is the following:

$$x_i\ y_i\ l\ \theta \quad \{\mathbf{beg},\mathbf{end}\}\mathbf{l}\{\mathbf{name}\}\ \rightarrow\ x_f\ y_f\ ,$$

where "**{beg,end}l{name}**" denotes any macro of type "l". This notation means that the four parameters x_i, y_i, l and θ must be present on the stack (θ being the last one to be pushed on the stack) before calling the macro. After returning, the macro leaves the parameters x_f and y_f at the two bottom levels of the stack. The meaning of these parameters is explained in figure 2.

x_i and y_i are the coordinates of the point where the propagator must begin or end, depending on the case. In both cases, l is the length of the propagator and θ is the angle (in degrees) between the propagator and the horizontal axis. The returned values x_f and y_f are always the coordinates of the point at which the propagator terminates. In the case of a propagator of type "end", they are identical to x_i and y_i.

[2] There is also a variant of "scalar", called "scalar1", which has an arrow at the end point. This type of propagator exists only in the linear version.

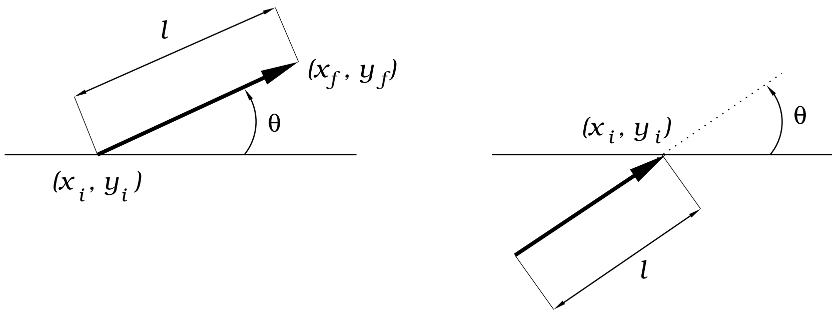

Figure 2: Left: the parameters of a macros of type **begl{name}**. Right: the parameters of a macro of type **endl{name}**.

Having these coordinates returned in the stack is very convenient to draw sequences of propagators, when the endpoint of a propagator is used as the starting point of the next propagator. In this case, only the parameters l and θ must be supplied to the next propagator command, since the first two arguments are already present in the stack.
For cases where it is not convenient to work with this "polar representation" of a linear propagator, we provide the macro **getlparam** that converts from $x_i\ y_i\ x_f\ y_f$ to $x_i\ y_i\ l\theta$:

$$x_i\ y_i\ x_f\ y_f \quad \textbf{getlparam} \ \rightarrow \ x_i\ y_i\ r\ \theta$$

Therefore, one can use the sequence:

$$x_i\ y_i\ x_f\ y_f \quad \textbf{getlparam}\ \ \textbf{begl}\{\textbf{name}\}$$

in order to draw a linear propagator from the knowledge of the coordinates of its starting point and endpoint.

4.2.2 Macros of type "c"

The macros of type "c" need five arguments, and return two numbers:

$$x_i\ y_i\ r\ \theta_1\ \theta_2 \quad \{\textbf{beg},\textbf{end}\}\textbf{c}\{\textbf{name}\} \ \rightarrow \ x_f\ y_f\ ,$$

where "**{beg,end}c{name}**" denotes any macro of type "c". The meaning of these parameters is illustrated in figure 3.
x_i and y_i are the coordinates of the point where the propagator must begin or end, depending on the case. In both cases, r is the radius of the circle that supports the propagator. θ_1 is the angle between the horizontal axis and the

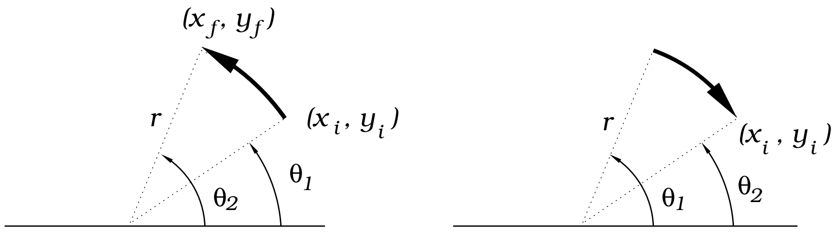

Figure 3: Left: the parameters of a macros of type **begc{name}**. Right: the parameters of a macro of type **endc{name}**.

radius that points to the beginning of the propagator, while θ_2 is the angle between the horizontal axis and the radius that points to the end point of the propagator. The returned values x_f and y_f are always the coordinates of the point where the propagator ends. In the case of a propagator of type "end", they are identical to x_i and y_i.

4.3 Setting the width

The width of the propagators is stored in a global variable called **Width**, whose default value is 0.03. If one or several propagators must be traced with a different width, this can be changed by the command:

/**Width** *newvalue* **def**

Note that this modification of the width affects all the subsequent propagators: one should therefore reset the width to its default value when this change should cease.

5 Examples

In this section, we provide several examples of Feynman diagrams, accompanied with their source code, in order to illustrate the usage of the macros described in the previous section.

5.1 Example 1

```
\begin{figure}[htb]
\centerline{
\begin{feynman}{3cm}
\diagram{20}{-7 1 translate
```

```
gsave
-0.5 0 translate
0 0 1 0 beglghost
/Width 0.1 def
1 -180 0 begcfermion
/Width 0.03 def
2 copy
1 0 beglphoton
pop pop
/Width 0.1 def
1 0 180 begcfermion
/Width 0.03 def
pop pop
grestore
gsave
5 0 translate
0 0 1 0 beglghost
1 -180 0 begcfermion
2 copy
1 0 beglphoton
pop pop
1 0 90 begcfermion
1 -90 endlghost
1 90 180 begcfermion
pop pop
grestore
gsave
10 0 translate
0 0 1 0 beglghost
1 -180 0 begcfermion
2 copy
1 0 beglphoton
pop pop
1 0 60 begcfermion
1 -120 endlghost
1 60 120 begcfermion
1 -60 endlghost
1 120 180 begcfermion
pop pop
grestore}
```

```
\end{feynman}
}
\caption{\label{fig:expansion} Example 1.}
\end{figure}
```

produces the output displayed in figure 4.

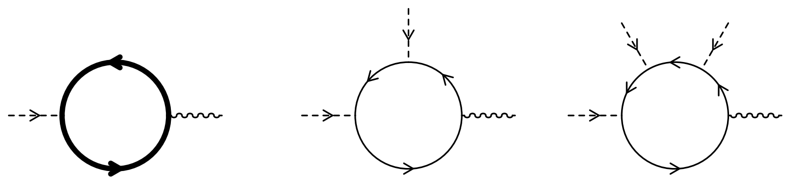

Figure 4: Example 1.

This example illustrates also the usage of the very useful postscript command **nop** which discards the lowest level of the stack, and of the command **2 copy** which duplicates the last two levels of the stack. It also shows the usage of *local translations*, whose action is limited to the graphic primitives enclosed by the same **gsave**...**grestore** pair. This is useful in order to move around a given Feynman diagram in a figure that contains several diagrams.

5.2 Example 2

```
\begin{figure}[htb]
\centerline{
\begin{feynman}{3cm}
\diagram{30}{-2 1 translate
        0 0 1 0 beglghost
        1 -180 0 begcfermion
        2 copy
        1 0 beglphoton
        pop pop
        1 0 180 begcfermion
        pop pop
        0.5 ti
        0 setgray
        0.3 0.2  moveto
        (K) show
```

```
        1.9 0.45 moveto
        (L) show
        0.4 symb
        -0.2 -0.5 moveto (p) show
        0.3 ti 0 0.15 rmoveto (0) show}
\end{feynman}}
\caption{\label{fig:2point} Example 2.}
\end{figure}
```

generates the output of figure 5. This example shows how to insert labels

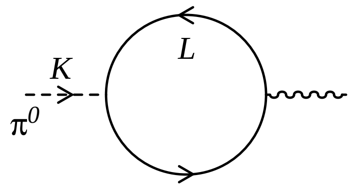

Figure 5: Example 2.

on a postscript diagram.

5.3 Example 3

```
\begin{figure}[htb]
\centerline{
\begin{feynman}{4cm}
\diagram{30}{-4 -1.7 translate
        gsave
        1 2 translate
        0 0 1 60 beglghost
        1.5 30 beglfermion
        2 copy
        1 0 beglphoton
        pop pop
        1.5 150 beglfermion
        1 -60 endlghost
        1.5 -90 beglfermion
        0.5 tr
        3.4 1.45 moveto (+) show
```

```
        0.4 symb
        -0.35 -0.1 moveto (s) show
        -0.5 3 moveto (p) show
        0.3 ti 0 0.15 rmoveto (0) show
        0.5 ti
        0.35 0 moveto (S) show
        0.45 2.8 moveto (P) show
        0.05 1.5 moveto (L) show
        2.7 1.8 moveto
        -0.7 0 rlineto currentpoint stroke 180 arrow stroke
        2.3 1.95 moveto (K) show
        grestore
        gsave
        5.2 2 translate
        0 0 1 60 beglghost
        1.5 30 beglfermion
        2 copy
        1 0 beglphoton
        pop pop
        1.5 150 beglfermion
        1 -60 endlghost
        1.5 -90 beglfermion
        0.4 symb
        -0.35 3 moveto (s) show
        -0.5 -0.1 moveto (p) show
        0.3 ti 0 0.15 rmoveto (0) show
        0.5 ti
        0.35 0 moveto (P) show
        0.45 2.8 moveto (S) show
        0.05 1.5 moveto (L) show
        2.7 1.8 moveto
        -0.7 0 rlineto currentpoint stroke 180 arrow stroke
        2.3 1.95 moveto (K) show
        grestore}
\end{feynman}
}
\caption{\label{fig:3point} Example 3.}
\end{figure}
```

Yet another example. See the output in figure 6. Nothing really new here.

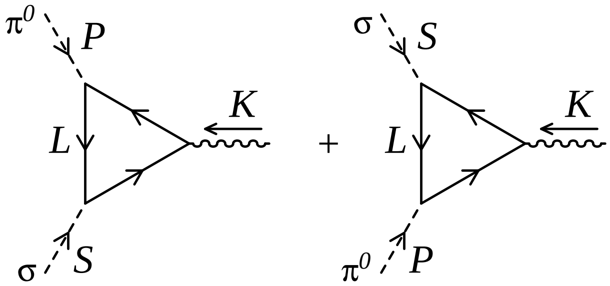

Figure 6: Example 3.

5.4 Example 4

```
\begin{figure}[htb]
\centerline{
\begin{feynman}{4cm}
\diagram{30}{-3.5 0.4 translate
gsave
0 0 7 0 beglscalar1
0 2.2 7 180 endlscalar1
0.5 2.2 0.7 0.7 getlparam beglphoton
0.5 0 0.7 0.7 getlparam beglphoton
0.7 0.7 2.5 1.1 getlparam beglphoton
1.1 2.2 1.8 1.7 getlparam beglphoton
1.8 1.7 2.5 1.1 getlparam beglphoton
1.8 1.7 1.8 1.05 getlparam beglphoton
1.8 0.8 1.8 0.0 getlparam beglphoton
2.5 1.1 3.5 1.1 getlparam beglphoton
3.5 0 3.5 1.1 getlparam beglphoton
3.5 1.1 4.5 1.1 getlparam beglphoton
4.5 1.1 5.5 0.55 getlparam beglphoton
5.5 0.55 6.5 0 getlparam beglphoton
5.5 0.55 5.5 1.3 getlparam beglphoton
5.5 1.3 5.5 2.2 getlparam beglphoton
5.5 1.3 6.5 2.2 getlparam beglphoton
4.5 1.1 4.5 2.2 getlparam beglphoton
grestore}
\end{feynman}
}
```

```
\caption{\label{fig:4} Example 4.}
\end{figure}
```

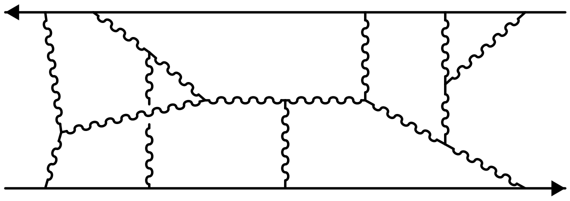

Figure 7: Example 4.

This example illustrates the usage of the macro **getlparam** in order to convert the coordinates of the starting point and endpoint into the parameters needed for the macros of type "l". This macro is useful in order to draw complicated networks of propagators.