

jQuery.Feyn: Drawing Feynman Diagrams with SVG

Zan Pan*

Institute of Theoretical Physics,
Chinese Academy of Sciences, Beijing, 100190, P.R. China

November 27, 2013

Abstract

jQuery.Feyn is a tool for drawing Feynman diagrams with Scalable Vector Graphics (SVG), written in JavaScript and runs in modern browsers. It features predefined propagator styles, vertex types, and symbols. Math formulae can be included as external graphics, or typeset with TeX through MathJax library. The generated SVG code can be easily modified to make fine adjustments and conveniently transferred using copy-and-paste.

1 Introduction

In the field of high energy physics, Feynman diagrams are widely used as pictorial representations of the interaction of sub-atomic particles¹. However, it is not easy to draw such a complicated object in publications (Interestingly, Feynman slash notation is also not easy to typeset). This has led to the development of computer programs. In the framework of LaTeX, there are four approaches: Michael Levine's `feynman`² bundle, Jos Vermaseren's `axodraw`³ package, Thorsten Ohl's `feynmf`⁴ package, and Norman Gray's `feyn`⁵ font. Besides, Binosi *et. al.*'s `JaxoDraw`^{6,7} and Hahn and Lang's `FeynEdit`⁸ are both written in Java which provide graphical user interfaces.

The reason why we designed another program is that publishing with HTML5⁹ has grown to be a very promising technology and requires a tool running in browsers to produce visually pleasing Feynman diagrams for the Web such as Wikipedia or online scientific articles. jQuery.Feyn is an attempt to fill that gap. Compared with the programs mentioned above, our program has great advantages in flexibility and portability.

1.1 Overview

jQuery.Feyn is a [jQuery](#) plugin to facilitate drawing Feynman diagrams with SVG¹⁰. It makes full use of jQuery's succinctness and extensibility to embrace the tagline: *write less, do more*. The following provides a summary of jQuery.Feyn's main features:

- Automatic generation of clean SVG source code
- Easy to use, easy to make fine adjustments
- Predefined propagator styles, vertex types, and symbols
- Support for typesetting labels and including external graphics
- Lightweight, cross-browser, and fully documented

The home of jQuery.Feyn project is <http://photino.github.io/jquery-feyn/>. Please refer to this link for up-to-date documentation and practical examples.

* panzan@itp.ac.cn

1.2 Supported Browsers

Any modern browsers for desktop or mobile with a basic support of inline SVG in HTML5 in the standards mode should be OK to run jQuery.Feyn. However, we do not guarantee that all of them will display the same SVG exactly on your screen due to their disparities in SVG rendering and support level. Also note that mobile browsers often show a lot of quirks that are hard to work around because of their limitations and different UI assumptions. The following provides an incomplete list of supported browsers:

- Firefox 4+
- Chrome 7+
- Opera 11.6+
- Safari 5.1+
- IE 9+

Personally, I recommend Firefox 24+ and Chrome 28+, on which my testing will be conducted continually. There is no doubt that newer browsers always have better support for SVG and hence better support for jQuery.Feyn.

1.3 Bug Reports and Comments

The preferred way to report bugs is to use the GitHub issue tracker:

<https://github.com/photino/jquery-feyn/issues>

Of course, you can also email me at panzan@itp.ac.cn or panzan89@gmail.com. When reporting bugs, you should be as specific as possible about the problem so that we can easily reproduce it. If possible, please test them on Firefox 24+ and Chrome 28+ to get rid of your browser's quirks. Comments, questions, and requests for adding more features are also welcome.

1.4 License

Copyright (C) 2013 by Zan Pan <panzan89@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2 Usage

2.1 Getting Started

You can start your tour from jQuery.Feyn's online demo:

<http://photino.github.io/jquery-feyn/demo.html>

Settings for Feynman diagrams are in the form of JavaScript's liberal object notation, whose simplicity and extensibility has already made it possible for jQuery.Feyn to be both lightweight and powerful. we hope you will enjoy this feature if it is still unfamiliar to you. Knowledge on jQuery is not necessary to use jQuery.Feyn, but getting a feel of JavaScript syntax will make your exploration easier. More importantly, you should be familiar with SVG markup language¹⁰. Unlike PostScript, the SVG code has great readability. It is not a problem to grasp it in a short time.

To use jQuery.Feyn, the first thing you should do is to load the scripts found in the distribution:

```
1 <script src="js/jquery-2.0.2.min.js"></script>
2 <script src="js/jquery.feyn-1.0.0.min.js"></script>
```

Please note that jQuery library comes first. After this, you can proceed to configure your desired Feynman diagram like

```
1 <script>
2   $(document).ready(function() {
3     $("#container").feyn({
4       incoming: {i1: "20,180", i2: "180,180"},
5       outgoing: {o1: "20,20", o2: "180,20"},
6       vertex: {v1: "100,140", v2: "100,60"},
7       fermion: {line: "i1-v1-i2,o2-v2-o1"},
8       photon: {line: "v1-v2"}
9     });
10  });
11 </script>
```

The jQuery ID selector `$("#container")` can also be replaced by any other selector that selects a unique block-level element in the document, which serves as the container of jQuery.Feyn's SVG output. The minimal example illustrated above represents a QED process (see Figure 1).

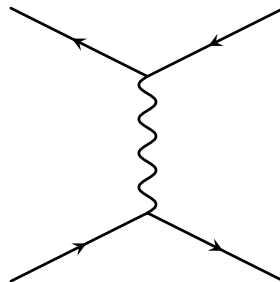


Figure 1. The minimal example of jQuery.Feyn's output

As can be seen, jQuery.Feyn has done the most part of work automatically. If you are unsatisfied with the output of jQuery.Feyn, or if you would like to add a graphics element that are not provided by jQuery.Feyn, you can always modify the SVG code manually. By setting the `standalone` option to `true`, you can edit the source code for your diagram in the textarea.

2.2 Default Options

The default options of jQuery.Feyn's Feyn constructor are listed as follows

```
1 {
2   xmlns: "http://www.w3.org/2000/svg",
3   xlink: "http://www.w3.org/1999/xlink",
4   version: "1.1",
5   x: 0,
6   y: 0,
7   width: 200,
8   height: 200,
9   title: "",
10  description: "Feynman diagram generated by jQuery.Feyn",
11  standalone: false,
12  selector: false,
13  grid: {show: false, unit: 20},
14  color: "black",
15  thickness: 1.6,
16  tension: 1,
17  ratio: 1,
18  clockwise: false,
19  incoming: {},
20  outgoing: {},
21  vertex: {},
22  auxiliary: {},
23  fermion: {arrow: true},
24  photon: {period: 5, amplitude: 5},
25  scalar: {arrow: false, dash: "5 5", offset: 2},
26  ghost: {arrow: true, thickness: 3, dotsep: 8, offset: 5},
27  gluon: {width: 15, height: 15, factor: 0.75, percent: 0.6, scale: 1.15},
28  symbol: {},
29  node: {show: false, thickness: 1, type: "dot", radius: 3, fill: "white"},
30  label: {family: "Georgia", size: 15, face: "italic"},
31  image: {},
32  mathjax: false,
33  ajax: false
34 }
```

As you have already seen, they can be overridden by passing an option object to `feyn` method. However, options are not checked in any way, so setting bogus option values may lead to odd errors. For JavaScript internal errors excluding syntax errors, jQuery.Feyn will write the error information to the container of your Feynman diagram to remind you of the case. A complete list of available options will be given in the [appendix A](#).

2.3 Tips, Tricks, and Troubleshooting

- You can edit the generated SVG code directly to make fine adjustments. The corresponding SVG output will be updated immediately when you trigger a text change event by clicking

outside of the textarea. It is your own responsibility to ensure the validity of your SVG code. Note that reloading the page will discard your editing, so please manually save the change in an external SVG file. You can use the two textareas in this way: one for testing, and the other for producing.

- Simple labels such as particle names, momentum, data, and comments, can be typeset with jQuery.Feyn's `label` option. It supports subscript, superscript, bar and tilde accents by using the `dx` and `dy` attributes of `<tspan>`. For complicated mathematical expressions, they should be included as external SVG images with the `image` option. Troy Henderson's [LaTeX Previewer](#) provides a user-friendly utility for generating LaTeX output¹¹.
- For special characters such as greek letters and mathematical operators, it is recommended to input the unicode entity by citing its decimal number, for example α can be accessed by `α`. A list of frequently used characters can be found at [ascii-code.com/html-symbol](#).
- If you are familiar with Mathematical Markup Language (MathML), you can also include mathematical expressions by adding the `<foreignObject>` element manually. Please check [caniuse.com/mathml](#) to see whether or not your browser has a good support for MathML.
- When [MathJax](#) is available, you can set jQuery.Feyn's `mathjax` option to `true` to typeset mathematics in TeX or LaTeX. This functionality also relies on browsers' support for the `<foreignObject>` element.
- SVG files can be converted to [EPS](#) and [PDF](#) online. If your SVG code has included some external SVG files, please set jQuery.Feyn's `ajax` option to `true` to merge their code directly, or copy, paste, and modify them manually. Before conversion, you should [check](#) the markup validity of your SVG code.
- Chrome does not support loading local files with ajax by default. You should start Google Chrome with the `-disable-web-security` or `-allow-file-access-from-files` option, otherwise you will get a network error.
- Firefox 23 or below has a [bug](#) of rendering the `<image>` element. Please update your browser to 24+.

3 Design

In this section, we will discuss the design (and implementation) of jQuery.Feyn from a user's perspective. Before drawing a Feynman diagram, you are encouraged to draft it on graph paper first.

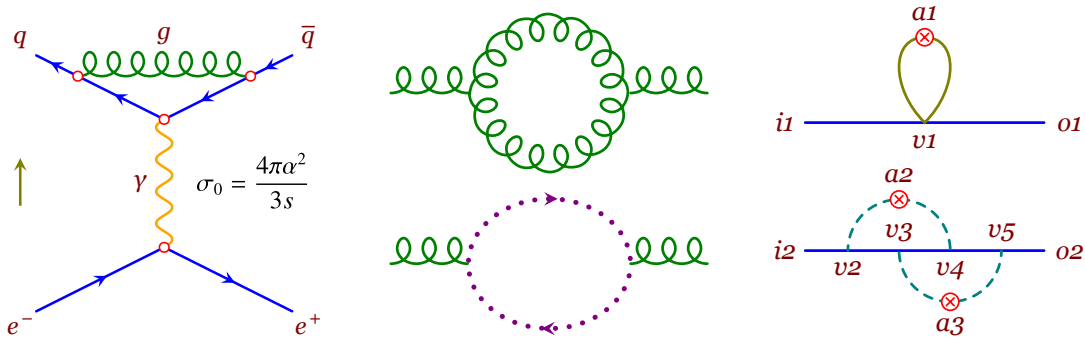


Figure 2. Illustrations on jQuery.Feyn's building blocks

Roughly speaking, a Feynman diagram can be treated as a directed or undirected graph consisting of a set of nodes and edges. As illustrated in Figure 2, jQuery.Feyn has provided five types

of basic building blocks that can be used to construct Feynman diagrams. Each type may contain one or more graphics primitives (*i.e.* the options):

- **Graph node:** sets the coordinates of nodes with the `incoming`, `outgoing`, `vertices` and `auxiliary` primitives, or draws the node marks with the `node` primitive
- **Propagator:** draws the propagators with the `fermion`, `photon`, `scalar`, `ghost`, and `gluon` primitives
- **Symbol:** draws symbols such as arrows, blobs, and so on with the `symbol` primitive
- **Label:** typesets labels with the `label` primitive
- **Image:** includes external graphics with the `image` primitive

For the design of line styles and symbols, we also have references to PSTricks¹², MetaPost¹³, and Asymptote¹⁴. Beyond Feynman diagrams, jQuery.Feyn also excels at drawing some mathematical diagrams with dense connections between nodes (see Figure 3).

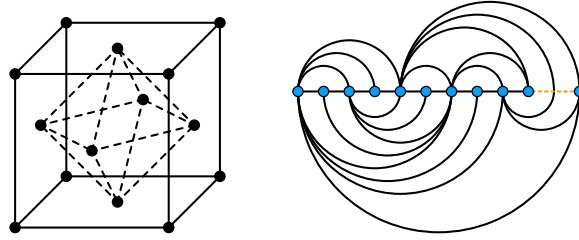


Figure 3. The cube with an octahedron inside and the Goldner-Harary graph drawn by jQuery.Feyn

3.1 Graph Nodes

A graph node is a coordinate pair extracted from the position string such as "20, 180" with a name with the prefix `i`, `o`, `v`, or `a`. They are respectively the first-letter of `incoming`, `outgoing`, `vertices`, and `auxiliary`. These terminologies should be self-explanatory and are not discussed further. In fact, all graphics nodes will be merged into one object in the base implementation. We retain this separation in the user interface just for semantic clarity.

To draw node marks, you should use the `node` primitive. Five types of node marks are provided. Of course, you can also specify different filled colors to denote different vertices (see Figure 4).



Figure 4. Examples of node marks provided by jQuery.Feyn

3.2 Propagators

We support five types of propagators: `fermion`, `photon`, `scalar`, `ghost`, and `gluon`. According to the conventions by Peskin and Schroeder¹⁵, only `fermion` and `ghost` propagators show arrows by default. In practice, boson propagators are represented by the sine curves and gluon propagators by elliptical arcs. They can be approximated by cubic Bézier paths in SVG, which will be discussed in the appendix B.

Each type of the propagators has three shapes: `line`, `arc`, and `loop`. The `tension` parameter controls the shape of arc radius for arc propagators; whereas the `ratio` parameter controls the shape of elliptical arc for `fermion`, `scalar`, and `ghost` loop propagators, *i.e.* the ratio of y-radius to x-radius. Their geometrical meanings are shown in Figure 5.

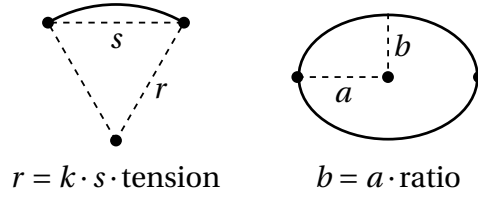


Figure 5. The geometrical layout of the arc and loop propagator (k is a constant)

3.3 Symbols

For the convenience of drawing complex diagrams, a small set of predefined symbols are provided: arrow, blob, bubble, condensate, hadron, and zigzag. Some of them can also support variants. Examples are illustrated in Figure 2 and Figure 6.

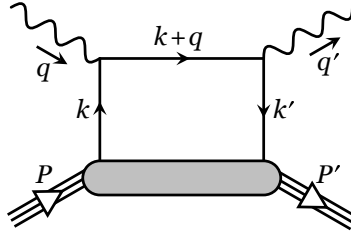


Figure 6. Feynman diagram for deeply virtual Compton scattering drawn by jQuery.Feyn

3.4 Labels

Labels are somewhat annoying. The `label` primitive has a nice support for subscript, superscript, and accents. For simple text or annotations such as particle names and momentum, this is enough. But it is not competent to typeset math formulae. Two solutions are available: to include as external graphics (see the first diagram in Figure 2), or to use the MathJax library (see Figure 7). The first method makes use of the `image` primitive and always works, while the other depends on browsers' support for the `foreignObject` element.

$$b, \nu \text{ } \text{ooooooo} \text{ } a, \mu \text{ } \underset{p}{=} \frac{-i\delta^{ab}}{p^2 - i\epsilon} \left(\eta^{\mu\nu} - (1 - \xi) \frac{p^\mu p^\nu}{p^2} \right)$$

Figure 7. Feynman diagram for the gluon propagator drawn by jQuery.Feyn with MathJax support

3.5 Images

The `image` primitive is provided to enhance jQuery.Feyn's extensibility. You can use it to include any external graphics whatever you like. As seen in Figure 2, we have embeded the math formula as an SVG file in the first diagram. It is your own duty to ensure that proper width and height values are assigned to the image object.

4 Summary

jQuery.Feyn is a tool to draw Feynman diagrams in browsers, which utilizes the power of SVG. We have explained how to use it and presented some examples of its building blocks. It is always hard to reconcile ease-of-use with expressiveness. We encourage the users to modify the generated code manually to make fine adjustments. As an open standard with the history over a decade, SVG has rich graphic features and effects, which makes it full of fun to learn in depth. This also contributes to the reason why we introduce jQuery.Feyn to physicists for preparing their publications.

Acknowledgements

We are grateful to GitHub to host our project. We also acknowledge all the people who send us feedback during the testing phase.

A Complete Reference of Options

In the following, we provide a complete list of jQuery.Feyn's options. `String`, `Number`, `Boolean`, `Array`, and `Object` are JavaScript's data types. Also note that the bold text in the parentheses is the corresponding default value.

xmlns : `String ("http://www.w3.org/2000/svg")`

Sets the `xmlns` attribute of `<svg>` which binds the SVG namespace

xlink : `String ("http://www.w3.org/1999/xlink")`

Sets the `xlink:href` attribute of `<svg>` which defines an IRI reference type as a URI

version : `String ("1.1")`

Sets the `version` attribute of `<svg>` which indicates the SVG language version

x : `Number (length | 0)`

Sets the `x` attribute of `<svg>` which indicates a x-axis coordinate in the user coordinate system

y : `Number (length | 0)`

Sets the `y` attribute of `<svg>` which indicates a y-axis coordinate in the user coordinate system

width : `Number (length | 200)`

Sets the `width` attribute of `<svg>` which indicates a horizontal length in the user coordinate system

height : `Number (length | 200)`

Sets the `height` attribute of `<svg>` which indicates a vertical length in the user coordinate system

title : `String (text)`

Sets the content for the `<title>` element which displays a title for the Feynman diagram

description : `String (text | "Feynman diagram generated by jQuery.Feyn")`

Sets the content for the `<desc>` element which describes the Feynman diagram

standalone : `Boolean (true | false)`

Enables or disables the SVG code editor to make fine adjustments and save as a standalone SVG file

selector : Boolean (true | **false**)
 Determines whether or not to set `id` and `class` attributes for SVG elements

grid : Object

show : Boolean (true | **false**)
 Determines whether or not to display a grid system to facilitate your drawing

unit : Number (length | **20**)
 Sets the length of subdivision for the grid system

color : String (paint | **black**)
 Sets global `stroke` attribute for SVG elements which defines the color of the outline

thickness : Number (length | **1.6**)
 Sets global `stroke-width` attribute for SVG elements which specifies the width of the outline

tension : Number (parameter | **1**)
 Sets global parameter of arc radius and the zigzag amplitude

ratio : Number (parameter | **1**)
 Sets global parameter of elliptical arcs for fermion, scalar, and ghost loop propagators

clockwise : Boolean (true | **false**)
 Sets global `clockwise` parameter for propagators

incoming : Object

i1, i2, i3, ... : String (position)
 Sets the coordinate pairs of graph nodes for incoming particles

outgoing : Object

o1, o2, o3, ... : String (position)
 Sets the coordinate pairs of graph nodes for outgoing particles

vertex : Object

v1, v2, v3, ... : String (position)
 Sets the coordinate pairs of graph nodes for vertices

auxiliary : Object

a1, a2, a3, ... : String (position)
 Sets the coordinate pairs of graph nodes for miscellaneous symbols

fermion : Object

color : String (paint | **inherit**)
 Sets the `stroke` attribute for `<g>` into which fermion propagators are grouped

thickness : Number (length | **inherit**)
 Sets the `stroke-width` attribute for `<g>` into which fermion propagators are grouped

tension : Number (parameter | **inherit**)
 Sets the parameter of arc radius for fermion propagators

ratio : Number (parameter | **inherit**)
 Sets the parameter of elliptical arcs for fermion propagators

arrow : Boolean (**true** | false)
 Determines whether or not to show arrows for fermion propagators

clockwise : Boolean (true | **false**)
 Sets the direction of arrows for arc and loop fermion propagators

line : String (connections)
 Sets the directed edges between graph nodes for fermion lines
arc : String (connections)
 Sets the directed edges between graph nodes for fermion arcs
loop : String (connections)
 Sets the directed edges between graph nodes for fermion loops
photon : Object
 color : String (paint | **inherit**)
 Sets the stroke attribute for <g> into which photon propagators are grouped
 thickness : Number (length | **inherit**)
 Sets the stroke-width attribute for <g> into which photon propagators are grouped
 tension : Number (parameter | **inherit**)
 Sets the parameter of arc radius for photon propagators
 clockwise : Boolean (true | **false**)
 Determines whether the first wiggle starts up or down for photon propagators
 period : Number (parameter | 5)
 Sets the period parameter for photon propagators
 amplitude : Number (parameter | 5)
 Sets the amplitude parameter for photon propagators
 line : String (connections)
 Sets the directed edges between graph nodes for photon lines
 arc : String (connections)
 Sets the directed edges between graph nodes for photon arcs
 loop : String (connections)
 Sets the directed edges between graph nodes for photon loops
scalar : Object
 color : String (paint | **inherit**)
 Sets the stroke attribute for <g> into which scalar propagators are grouped
 thickness : Number (length | **inherit**)
 Sets the stroke-width attribute for <g> into which scalar propagators are grouped
 tension : Number (parameter | **inherit**)
 Sets the parameter of arc radius for scalar propagators
 ratio : Number (parameter | **inherit**)
 Sets the parameter of elliptical arcs for scalar propagators
 arrow : Boolean (true | **false**)
 Determines whether or not to show arrows for scalar propagators
 clockwise : Boolean (true | **false**)
 Sets the direction of arrows for arc and loop scalar propagators
 dash : String (dasharray | "5 5")
 Sets the stroke-dasharray attribute for <g> into which scalar propagators are grouped
 offset : Number (length | 2)
 Sets the stroke-offset attribute for <g> into which scalar propagators are grouped
 line : String (connections)
 Sets the directed edges between graph nodes for scalar lines
 arc : String (connections)
 Sets the directed edges between graph nodes for scalar arcs

loop : String (connections)
 Sets the directed edges between graph nodes for scalar loops

ghost : Object

color : String (paint | **inherent**)
 Sets the stroke attribute for <g> into which ghost propagators are grouped

thickness : Number (length | **inherent**)
 Sets the stroke-width attribute for <g> into which ghost propagators are grouped

tension : Number (parameter | **inherent**)
 Sets the parameter of arc radius for ghost propagators

ratio : Number (parameter | **inherent**)
 Sets the parameter of elliptical arcs for ghost propagators

arrow : Boolean (**true** | false)
 Determines whether or not to show arrows for ghost propagators

clockwise : Boolean (true | **false**)
 Sets the direction of arrows for arc and loop ghost propagators

dotsep : Number (length | **8**)
 Sets the stroke-dasharray attribute for <g> into which ghost propagators are grouped

offset : Number (length | **5**)
 Sets the stroke-offset attribute for <g> into which ghost propagators are grouped

line : String (connections)
 Sets the directed edges between graph nodes for ghost lines

arc : String (connections)
 Sets the directed edges between graph nodes for ghost arcs

loop : String (connections)
 Sets the directed edges between graph nodes for ghost loops

gluon : Object

color : String (paint | **inherent**)
 Sets the stroke attribute for <g> into which gluon propagators are grouped

thickness : Number (length | **inherent**)
 Sets the stroke-width attribute for <g> into which gluon propagators are grouped

tension : Number (parameter | **inherent**)
 Sets the parameter of arc radius for gluon propagators

clockwise : Boolean (true | **false**)
 Determines whether the first wiggle starts up or down for gluon propagators

width : Number (length | **15**)
 Sets the coil width of gluon propagators

height : Number (length | **15**)
 Sets the coil height of gluon propagators

factor : Number (parameter | **0.75**)
 Sets the factor parameter for gluon propagators

percent : Number (parameter | **0.6**)
 Sets the percent parameter for gluon propagators

scale : Number (parameter | **1.15**)
 Sets the scale parameter for gluon arcs and loops

line : String (connections)
 Sets the directed edges between graph nodes for gluon lines

arc : String (connections)
 Sets the directed edges between graph nodes for gluon arcs

loop : String (connections)
 Sets the directed edges between graph nodes for gluon loops

symbol : Object

color : String (paint | **inherent**)
 Sets the stroke attribute for <g> into which symbols are grouped

thickness : Number (length | **inherent**)
 Sets the stroke-width attribute for <g> into which symbols are grouped

s1, s2, s3, ... : Array

sn[0] : String (position)
 Sets the coordinates of graph nodes for symbol

sn[1] : Number (angle)
 Sets the x-axis-rotation angle for symbol

sn[2] : String ("arrow" | "blob" | "bubble" | "condensate" | "hadron" | "zigzag")
 Sets the symbol type

sn[3] : Number (parameter | **20**)
 Sets the distance parameter for the symbol

sn[4] : Number (parameter | **4**)
 Sets the height parameter for the symbol

sn[5] : Boolean (true | **false**)
 Enables or disables a variant for the symbol

node : Object

color : String (paint | **inherent**)
 Sets the stroke attribute for <g> into which nodes are grouped

thickness : Number (length | **inherent**)
 Sets the stroke-width attribute for <g> into which nodes are grouped

show : Boolean | String (**false** | "i" | "o" | "v" | "a" | ... | "iova")
 Determines whether or not to show nodes

type : String ("box" | "boxtimes" | "cross" | **"dot"** | "otimes")
 Sets the node type

radius : Number (length | **3**)
 Sets the radius parameter of nodes

fill : String (paint | **"white"**)
 Sets the fill attribute for <g> into which nodes are grouped

label : Object

color : String (paint | **inherent**)
 Sets the stroke attribute for <g> into which labels are grouped

thickness : Number (length | **0**)
 Sets the stroke-width attribute for <g> into which labels are grouped

fill : String (paint | **"white"**)
 Sets the fill attribute for <g> into which labels are grouped

family : String (family-name | **"Georgia"**)
 Sets the font-family attribute for <g> into which labels are grouped

size : Number (length | **15**)
 Sets the font-size attribute for <g> into which labels are grouped

weight : String ("normal" | "bold" | "bolder" | "lighter")
 Sets the font-weight attribute for <g> into which labels are grouped
face : String ("normal" | "italic" | "oblique")
 Sets the font-style attribute for <g> into which labels are grouped
align : String ("start" | "middle" | "end")
 Sets the text-anchor attribute for <g> into which labels are grouped
t1, t2, t3, ... : Array
tn[0] : String (position)
 Sets the coordinates of graph nodes for label
tn[1] : String (text)
 Sets the text of label as the content of <tspan>
tn[2] : Number (length | 18)
 Sets the width attribute for <foreignObject>
tn[3] : Number (length | 30)
 Sets the height attribute for <foreignObject>
image : Object
m1, m2, m3, ... : Array
mn[0] : String (position)
 Sets the coordinates of position for including external image
mn[1] : String (file)
 Sets the path for external image file
mn[2] : Number (length | 32)
 Sets the width attribute for image
mn[3] : Number (length | 32)
 Sets the height attribute for image
mathjax : Boolean (true | false)
 Determines whether or not to use MathJax to typeset mathematics in labels
ajax : Boolean (true | false)
 Determines whether or not to merge the code of external SVG image directly

B Cubic Bézier Splines

SVG can produce graceful curves by graphing quadratic and cubic equations. As mentioned before, in jQuery.Feyn we use cubic Bézier segments to approximate the photon and gluon propagators (for the line type, gluon propagators are drawn as elliptical arc paths supported by SVG directly). Now, we come to answer the problem how to approximate a sine curve and an ellipse by cubic Bézier splines. By using the symmetry, we only need to consider one-fourth of its period.

A cubic Bézier spline needs four control points P_0 , P_1 , P_2 and P_3 . It is obvious that the end-points and their tangents should be accurate. Then, we need another constraint to determine all the control points. For approximating the sine curve, we require the curvature of the Bézier spline in the endpoints to be accurate as well. According to the derivation in <http://mathb.in/1447>, we can obtain the following control points

$$P_0 = (0, 0), \quad P_1 = (\lambda p / \pi, \lambda a / 2), \quad P_2 = (2p / \pi, a), \quad P_3 = (p, a), \quad \lambda = 0.51128733$$

where p is one-fourth of the period of the sine curve and a is the amplitude. For approximating an ellipse with semi-major axis a and semi-minor axis b , we can approximate a circle first and then

make scaling transformations. The control points for the ellipse in the first quarter are given by

$$P_0 = (0, b), \quad P_1 = (\kappa a, b), \quad P_2 = (a, \kappa b), \quad P_3 = (a, 0), \quad \kappa = 0.55191502$$

Here we have used the result in Spencer Mortensen's article:

<http://spencermortensen.com/articles/bezier-circle/>

where the constant κ is determined from the constraint that the maximum radial distance from the circle to the Bézier curve must be as small as possible.

References

- ¹ D. Kaiser, *Physics and Feynman's Diagrams*, American Scientist **93** (2005), 156–165.
- ² M.J.S. Levine, *A LaTeX Graphics Routine for Drawing Feynman Diagrams*, Comput. Phys. Commun. **58** (1990), 181–198.
- ³ J.M. Vermaseren, *Axodraw*, Comput. Phys. Commun. **83** (1994), 45–58.
- ⁴ T. Ohl, *Drawing Feynman Diagrams with L^AT_EX and METAPOST*, Comput. Phys. Commun. **90** (1995), 340–354, [arXiv:hep-ph/9505351](https://arxiv.org/abs/hep-ph/9505351).
- ⁵ N. Gray, *The feyn Font*, <http://www.ctan.org/tex-archive/fonts/feyn>.
- ⁶ D. Binosi, L. Theussl, *JaxoDraw: A Graphical User Interface for Drawing Feynman Diagrams*, Comput. Phys. Commun. **161** (2004), 76–86, [arXiv:hep-ph/0309015](https://arxiv.org/abs/hep-ph/0309015).
- ⁷ D. Binosi, J. Collins, C. Kaufhold, L. Theussl, *JaxoDraw: A Graphical User Interface for Drawing Feynman Diagrams. Version 2.0 Release Notes*, Comput. Phys. Commun. **180** (2009), 1709–1715, [arXiv:0811.4113 \[hep-ph\]](https://arxiv.org/abs/0811.4113).
- ⁸ T. Hahn, P. Lang, *FeynEdit — A Tool for Drawing Feynman Diagrams*, Comput. Phys. Commun. **179** (2008), 931–935, [arXiv:0711.1345 \[hep-ph\]](https://arxiv.org/abs/0711.1345).
- ⁹ S. Kleinfeld, *HTML5 for Publishers*, O'Reilly & Associates (2011), <http://chimera.labs.oreilly.com/books/1234000000770/>.
- ¹⁰ J.D. Eisenberg, *SVG Essentials*, O'Reilly & Associates (2002), http://commons.oreilly.com/wiki/index.php/SVG_Essentials.
- ¹¹ T. Henderson, *User-friendly Web Utilities for Generating L^AT_EX Output and METAPOST Graphics*, TUGboat **33** (2012), 48–52, <http://www.tug.org/TUGboat/tb33-1/tb103henderson.pdf>.
- ¹² T. van Zandt, *PSTricks — PostScript Macros for Generic T_EX: User's Guide*, <http://www.ctan.org/tex-archive/graphics/pstricks/base/doc/pstricks-doc.pdf>.
- ¹³ J. Hobby, *METAPOST: A User's Manual*, <http://www.tug.org/docs/metapost/mpman.pdf>.
- ¹⁴ J. Bowman, A. Hammerlindl, T. Prince, *Asymptote: the Vector Graphics Language*, <http://asymptote.sourceforge.net/asymptote.pdf>.
- ¹⁵ M.E. Peskin, D.V. Schroeder, *An Introduction to Quantum Field Theory*, Addison-Wesley Publishing Company (1995).