INTRODUCTION TO LINE
A Feynman Diagram Graphic Generator*

by

Bruce LeClaire

*Stanford Linear Accelerator Center
Stanford University, Stanford, California, 94305*

A guide to making Feynman graphs through interactive programming.

This memo serves as an introduction to running the *LINE* exec. *LINE* allows the user to create a Topdraw file containing almost any Feynman graph. As a disclaimer, the author emphatically denies all responsibility for the calculability of such graphs.

* Work Supported by the Department of Energy, Contract DE-AC03-76SF00515.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Acknowledgements

This exec is the sole responsibility of the author, and its strengths and weaknesses lie with him. However, there are a few people without whom this version of the exec could not have been built. First of these are Cwen Saxton, for contributions to the cursor readout (ála GMPLLOT), and Charlotte Hee, who introduced me to the art of menu building. Many thanks are also due Michael Peskin, who in the course of just a few minutes was able to point out many of the shortcomings of an earlier release. I would also like to thank Bruce Sawhill for so bravely (foolishly?) adventuring into the exec as a first time user; the diagram on page 67 was made by him. Lastly, I would like to thank Deborah Lent for creating the plot on the coverpiece, it continues to give me much pleasure.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

Table of Contents

Introduction	1
What You Can Draw	
Part I (Line)	2
(Loop L option)	4
(Loop B option)	5
Summary of choices I	6
Part II (Line)	7
Summary of choices II	8
Example 1	9
Example 2	17
Example 3	24
LINE Menus	34
Example 4	36
Example 4 Exec File	44
Appendices	
I. LINEDEF CTRL File	46
II. LINE Fortran Control Parameters	47
III. LOOPDEF CTRL File	50
IV. LOOP Fortran Control Parameters	51
V. Label Positioning for LINE and LOOP	53
VI. LINE Initialization and Default Settings	54
VII. LINE and LOOP CTRL Menu PF Keys	56
VIII. LINE Exec Master Menu PF Keys	57
IX. Default Topdraw Axes	59
X. Running LINE	60
Form	60
Cursor Selection Options	65
XI. Additional Examples	66

Introduction

This is an exec for making Feynman graphs of amazing complexity with amazing simplicity - once you know how. And knowing how is hopefully quite easy to learn. The *LINE* exec works iteratively, repeatedly prompting you for the elements of the graph. You tell the exec what kinds of lines (or loops) to draw, and where to put them. Each line is plotted in a 10×10 Cartesian coordinate system using two *ordered* pairs of coordinates, (x_1, y_1) and (x_2, y_2) , as endpoints. The types of lines that can be drawn include fermions, scalars, photons, gluons, W or Z bosons. As each line is added the exec allows you to display the updated picture, and to erase the *last* line added (if necessary). In this way you can keep going, drawing and erasing, until you have exactly what you want. Easy.

Your handiwork is put into an output file on your A-disk. The exec prompts you for the file name at the beginning of the session, and the file type is set to TOPDRAW by default. Note that if the file already exists, the exec's output is appended to it, otherwise the exec creates a new file. This feature is handy, for instance, when building a higher order graph from an already existing lower order one. It should also be mentioned that the exec inserts default Topdraw axes at the beginning of the output file unless it finds them there already: see Appendix IX for the exact set of Topdraw commands involved. Throughout the session the Topdraw axes are displayed (unless you turn them off) in order to provide a much needed reference system for drawing. When the graph is completely finished the axes can be removed by a SET AXES OFF Topdraw command. In fact, this command is present in the inserted default axes, and merely needs to be activated by removing the Topdraw comment prefix (left parenthesis). The *LINE* exec also provides for automatic labeling of the lines and loops with a great amount of user control. Furthermore, the output file can be edited during the session allowing the user complete control over the plot. The details of these control features will be discussed later.

The exec allows two basic modes of operation, line by line, or full screen menus. It is most powerful when run from the full screen (43 or 24 VM) menus. However, because of the ease of demonstration, and for teaching purposes, the majority of examples in this manual are done in the line by line mode. The virtues of the menu style will be outlined later.

What You Can Draw: Part I.

This section illustrates all the different types of lines and loops that you can draw for a Feynman graph (with the default settings). They are divided according to whether *LINE* or *LOOP* fortran generates the pictures. The *LOOP* drawings are further divided into two categories: loops (L option) and bubbles (B option). The notation introduced here is exactly the same as used with the exec, so only the mnemonic labels appear.

LINE:

FERM

SCAL

COMP

PHOT

PHOTF

GLUE

GLUEF

WORZ

WORZF

XPART

LOOP: L option

FERM

SCAL

COMP

PHOT

GLUE

WORZ

XPART

LOOP: B option

FERM

SCAL

COMP

PHOT

GLUE

WORZ

XPART

Summary of choices I :

LINE $\left\{ \begin{array}{l} \text{FERM} \\ \text{PHOT} \\ \text{PHOTF} \\ \text{GLUE} \\ \text{GLUEF} \\ \text{WORZ} \\ \text{WORZF} \\ \text{SCAL} \\ \text{COMP} \\ \text{XPART} \end{array} \right\}$

LOOP
(*L or B option*)

$\left\{ \begin{array}{l} \text{FERM} \\ \text{PHOT} \\ \text{GLUE} \\ \text{WORZ} \\ \text{SCAL} \\ \text{COMP} \\ \text{XPART} \end{array} \right\}$

What You Can Draw: Part II.

This section shows the extended capabilities of LINE fortran. Together with the other graphic elements, these additional features allow the user to draw sophisticated physics diagrams going beyond Feynman graphs. It should be added that the user has a tremendous amount of control over the parameters of all the outputs; and an indication of this is attempted here.

LINE:

HELIC

ELLIP

PARAB

HYBER

Summary of choices II :

$LINE \left\{ \begin{array}{l} HELIC \\ ELLIP \\ PARAB \\ HYPER \\ AXES \end{array} \right\}$

Example 1

The best way to learn about the *LINE* exec is to use it, and so, let us begin by studying a couple of examples. The following example is an annotated session with the *LINE* exec, the same session which produced the above graph. You might decide to follow along at the terminal or you might just read along and make your own favorite diagram afterwards. Notations in roman font are the annotations, notations in typewriter font are the computer dialogue, and your responses are denoted by pronupts (->).

-> line go

This command is issued in CMS and starts executing the *LINE* exec. The argument, go, tells the exec not to display its notification message. Since the notification is included Appendix X we do not need to display it here.

Welcome to Line exec -- graphs made to order

The exec does a substantial amount of setup and checking at this time, luckily, most of this need never come to your attention. However, sophisticated applications will require some knowledge of this, and a full discussion is contained in Appendix VI.

Input file (cr=help).

-> example1

The first input required is the file name. You have two options, to create a new file or to append an old file; the exec will check your disk to see which is the case.

EXAMPLE1 TOPDRAW file is being created on your A-disk.

In this case it is a new file.

Do you want line, loop, or optn (end to quit. cr= help)?

-> line

The line and loop options are the standard choices. Here we choose the line option. These two different choices run two different fortran programs: LINE and LOOP. LINE can obviously generate more than just straight line segments, but no matter how fancy the particle, they all 'propagate' along straight lines on the graph paper (with the exception of the extended set; see What You Can Draw: Part II). LOOP contains all the same shapes, but its particles follow curved trajectories, including bubbles. The other choices can be picked at this time, but are not displayed by the prompt. If you choose, you can type in "optn" to be explicitly prompted for one of the additional options. There are three of these options: edit, menu, and ctrl. The edit option simply allows you to edit files during the session. This option can be used to place titles in your output file (the one we are now building). Two other files can directly edited with this option: LINE and LOOP CTRL. These are the two control files used by the fortran programs. The ctrl option also allows you to modify these files by refering to the control variable names directly. Modifying these control parameters allows you a great deal of control over the exact nature of the output. Lastly, the menu option allows you to leave the prompt mode to operate the exec from full screen menus. All of these options will be discussed in greater detail later. A short description of all the choices can be seen by typing in a carriage return.

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

-> ferm

FERM

Here you are talking to the fortran program, and it is prompting for the form of the line. Note that the program expects a word with at most five letters here. In this case a fermion line is choosen. If you forget the mnemonics they can be seen by typing in a question mark, or in a more abbreviated form by typing a carriage return. The program then needs the location of the endpoints, and it prompts you for one coordinate at a time as follows:

INPUT POINT X1:

--> 2

2.000

INPUT POINT Y1:

--> 2

2.000

INPUT POINT X2:

--> 4

4.000

INPUT POINT Y2:

--> 4

4.000

This step goes quickly, and really needs no comment except for the ordering of the points, which can be quite important. The first pair of coordinates specify the starting point, the second set specify the end point. For fermion lines this order tells the arrow which way to point.

Do you want to see graph (yes/no/cr=no) ?

--> no

Now you are back in the exec and it provides you with an opportunity to see your handiwork. The above option allows you to view the plot, but we won't use it here since just one line is rather dull. In fact, we'll wait until the entire graph is drawn before we plot it. This assumes that we won't be making any mistakes.

Line is done.

The exec tells you that you've completed the line, and then iterates through the entire process again. The next three lines drawn are exactly like the first, other than the change in coordinates, and so won't be

commented on. For reference, the coordinates of the line segments are:

$(2,2) \rightarrow (4,4) \rightarrow (2,6)$ and $(8,6) \rightarrow (6,4) \rightarrow (8,2)$. Onward.

Do you want line, loop, or optn (end to quit, cr= help)?

-> line

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

-> ferm

FERM

INPUT POINT X1:

-> 4

4.000

INPUT POINT Y1:

-> 4

4.000

INPUT POINT X2:

-> 2

2.000

INPUT POINT Y2:

-> 6

6.000

Do you want to see graph (yes/no/cr=no) ?

-> no

Line is done.

Do you want line, loop, or optn (end to quit, cr= help)?

-> line

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

Line is the choice, begin executing line fortran.

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

-> ferm

FERM

INPUT POINT X1:

-> 8

8.000

INPUT POINT Y1:
-> 6
6.000
INPUT POINT X2:
-> 6
6.000
INPUT POINT Y2:
-> 4
4.000
Do you want to see graph (yes/no/cr=no) ?
-> no
Line is done.
Do you want line, loop, or optn (end to quit, cr= help)?
-> line
Line is the choice, begin executing line fortran.
EXECUTION BEGINS...
WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)
-> ferm
FERM
INPUT POINT X1:
-> 6
6.000
INPUT POINT Y1:
-> 4
4.000
INPUT POINT X2:
-> 8
8.000
INPUT POINT Y2:
-> 2
2.000
Do you want to see graph (yes/no/cr=no) ?
-> no
Line is done.

Now all the fermion lines are done. We need just one more line to complete the graph.

Do you want line, loop, or optn (end to quit, cr= help)?

-> line

Line is the choice, begin. executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

-> phot

PHOT

This is all you have to do to get a photon line; the program does the rest (and other lines are just as easy). One important feature of LINE should be noted at this time; LINE puts a fixed number of wavelengths between any two points you give it, and it scales the amplitude of the oscillations according to the distance between these points. Of course, you can set parameters and adjust this as you wish (see example 3), but if you don't, then don't be surprised to get wildly different shapes for wildly different scales.

INPUT POINT X1:

-> 4

4.000

INPUT POINT Y1:

-> 4

4.000

INPUT POINT X2:

-> 6

6.000

INPUT POINT Y2:

-> 4

4.000

These coordinates connect the fermion lines.

Do you want to see graph (yes/no/cr=no) ?

-> yes

This option is now exercised. Inside the exec, a temporary file is being built using your 'brushstrokes'. Now you have decided to see the picture,

and the exec sends the Topdraw file to a graphic device. Whether this is your terminal, a public 'hotscreen' or a hardcopy device depends upon what option, if any, you have given *LINE*. In this example we have not provided *LINE* with any options. Thus the graph will be generated by Topdraw as a 4013 sequential file, and then displayed at the terminal by GMPLLOT. This assumes that the terminal has graphic capabilities. See Appendix X for how to run *LINE* with a plotting output.

Do you wish to keep this (yes/no/cr=yes)

-> yes

Of course you do. This time. But suppose you're goofing around and experimenting with something new, or suppose you made a typo, and the line didn't come out the way you wanted it to. Well, the exec gives you a second chance. It works like this: the very last line is not put into the output file until after you answer this question. If you say no then this line is erased and back to the top of the program loop you go. Of course, you have to look at the picture to get asked this question, otherwise the exec assumes you know what you're doing and automatically keeps the line. Anyway, if you did this example so far, you should have a fairly professional looking Feynman graph of electron annihilation into quarks. (Why not into muons, or taus, or so some other final state? Because you need quarks to radiate off gluons as we shall see in the next example.)

Line is done.

Do you want line, loop, or optn (end to quit, cr= help)?

-> end

At last, you're done. You just type in end to finish and claim your masterpiece.

Session is over, output file is in EXAMPLE1 TOPDRAW

R;

The output is left on your A-disk as EXAMPLE1 TOPDRAW. You might want to look at it, and observe that most of it is a series of points which are to be fed into Topdraw. But also notice that the exec annotates the lines rather well, telling you the kind of line, its endpoints, its length (s_{12}), and its angle. In fact, the exec even tells you when the file was created (or appended).

At this point you may have mixed feelings about the exec because it seems (at least from the length of the writeup) to be very complicated. But, with a little practice you'll whip these off in no time, and the techniques for super-complicated graphs are the same as the ones you've just learned for the simple graph here.

Example 2

This example utilizes an already existing plot to create a more involved diagram. It also introduces gluon lines as well as free gluons. Furthermore, the loop option is used to create fermion bubbles. Finally, the keep option is used to erase a misplaced line.

```
-> copy example1 Topdraw a example2 = =
R;
```

We begin by using the output file from example 1 as the starting point for example 2.

```
-> line go
Welcome to Line exec -- graphs made to order
Input file (cr=help).
-> example2
```

Since this file already exists the exec will append the session's output to it. Note that the default axes are already contained in this file since it was made with the *LINE* exec. If they weren't, the exec would put them in and notify you as such.

Do you want line, loop, or optn (end to quit, cr=help)?

```
-> line
```

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

-> phot

PHOT

INPUT POINT X1:

-> 3.25#3.25#3.25#4.75

Here we are putting in a QED vertex correction. All four coordinates are input at once using the CMS line end symbol #. This trick allows you to type the entries ahead of the prompts, in an extremely time saving fashion. When the exec has exhausted the stack it will once again wait up for you. But beware - don't end the line with the pound sign unless you want the last answer to be a carriage return. The numbers that follow are the fortran program echoing the input.

3.250

INPUT POINT Y1:

3.250

INPUT POINT X2:

3.250

INPUT POINT Y2:

4.750

Do you want to see graph (yes/no/cr=no) ?

-> yes

In this example we get to see the graph being built each step of the way.

Do you wish to keep this (yes/no/cr=yes)?

-> yes

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> line#glue#6.75#4.75#6.75#3.25

Here we are being extremely bold, and are typing five prompts ahead. This is quite reasonable to do once you become familiar with all the exec's questions. But notice that we still pause to observe the outcome.

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

GLUE

INPUT POINT X1:

6.750

INPUT POINT Y1:

4.750

INPUT POINT X2:

6.750

INPUT POINT Y2:

3.250

Do you want to see graph (yes/no/cr=no) ?

-> yes

And the gluon vertex correction works out quite nicely, (these lines are indeed quarks after all). Again notice that the input ordering of the points affects the line; here the pigtails face in toward the vertex. If the points were entered in reverse order the pigtails would have faced in the opposite direction. The input order also affects the photon lines. The general rule of thumb is to orient the open space to the left of the direction of travel (starting point to finishing point) when doing a vertex correction.

Do you wish to keep this (yes/no/cr=yes)?

-> yes

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

line#phot#2.5#6.5#4.5#6.5

Again we try the type ahead style.

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

2.500

INPUT POINT Y1:

6.500

INPUT POINT X2:

4.500

INPUT POINT Y2:

6.500

Do you want to see graph (yes/no/cr=no) ?

-> yes

But this time the output is not what we intended. Not to worry however, there are many ways to recover. First of all, we could edit the file and remove this particular line segment since *LINE* allows you to edit the output file during the session. This is not so painful to do, either, since the output is well annotated and you can locate each line quickly. However, *LINE* is even more clever, and has anticipated such an occasional mistake. The next prompt is usually an annoyance to answer, but in this case, it allows us to erase this *last* line and resume construction of the graph. Onward.

Do you wish to keep this (yes/no/cr=yes)?

-> no

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> line#phot#2.5#5.5#4.5#5.5

We try again.

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

2.500

INPUT POINT Y1:

5.500

INPUT POINT X2:

4.500

INPUT POINT Y2:

5.500

Do you want to see graph (yes/no/cr=no) ?

-> yes

Do you wish to keep this (yes/no/cr=yes)?

-> yes

This time we got it right.

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> line#phot#5.5#5.5#7.5#5.5

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

5.500

INPUT POINT Y1:

5.500

INPUT POINT X2:

7.500

INPUT POINT Y2:

5.500

Do you want to see graph (yes/no/cr=no) ?

-> yes

Do you wish to keep this (yes/no/cr=yes)?

-> yes

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> loop

This is new, and for the first time we draw a loop.

Loop is the choice, begin executing loop fortran.

EXECUTION BEGINS...

DO YOU WANT A LDDP OR A BUBBLE? (L OR B, FORMAT A1):

-> b

It may seem a little redundant having to ask for a loop only to be asked if you want a loop again (even though we ask for a bubble here). The reason for this lies inside the fortran: a bubble is just a loop which doesn't stop, and so loops and bubbles are built with the same code (almost). Thus, you must distinguish between the two possibilities.

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HEL?)


```
-> ferm
FERM
INPUT POINT X1:
-> 4.5#5.5
```

Here is the first point.

```
4.500
INPUT POINT Y1:
5.500
INPUT POINT X2:
-> 5.5#5.5
```

And here is the second point. Notice that the bubble actually uses the separation between the two input points in establishing its diameter. Furthermore, the arrows are oriented along the input direction, but are located on the perpendicular bisector of the line between the starting and finishing point.

```
5.500
INPUT POINT Y2:
5.500
Do you want to see graph (yes/no/cr=no) ?
-> yes
Do you wish to keep this (yes/no/cr=yes)?
-> yes
```

```
Loop is done.
Do you want line, loop, or optn (end to quit, cr=help)?
Do you want line, loop, or optn (end to quit, cr=help)?
line
Line is the choice, begin executing line fortran.
EXECUTION BEGINS...
WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP )
gluef
GLUEF
```

This choice gives us a free gluon, that is, a gluon which has an arrow at its end. This line has a fixed length, and the second set of coordinates is used merely to point the line. The free gluon takes its scale to be roughly

1.5 units. This is also adjustable; see the next example for a discussion of the program controls. But notice that throughout this example the length scale was kept at between 1 and 2 units.

INPUT POINT X1:

-> 7.5#2.5#8.5#3.0#yes

7.500

INPUT POINT Y1:

2.500

INPUT POINT X2:

8.500

INPUT POINT Y2:

3.000

Do you want to see graph (yes/no/cr=no) ?

Do you wish to keep this (yes/no/cr=yes)?

yes

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> end

Session is over, output file is in EXAMPLE2 TOPDRAW

R;

This ends the session, and in the output file you have a nice looking graph which would make even the most respectable theorist shudder. The next example shows how to make loops and to adjust the control parameters to make the lines and loops exactly as you want them.

Example 3

This next example is by far the most involved. It demonstrates some advanced techniques for controlling the output. Not only are loops introduced (which are simple to use), but the exact geometries of the loops are tailor-made using the control files provided by *LINE*. The simplicity of arrow control is demonstrated. Also, notice that the scaling nature of line and loop size allows for three separate graphs to be made on one plot.

-> line go

Welcome to Line exec -- graphs made to order

Input file (cr=help).

-> example3

EXAMPLE3 TOPDRAW file is being created on your A-disk.

The first thing to notice about the above graph is that the arrows are on the ends of the fermion lines. The default setting places the arrows in the middle, but that is where we want the external photons to couple. You actually have a great deal of control over the geometry of all the lines and loops, but it is usually hidden from view by the default settings. Hopefully, the defaults are sufficiently satisfying so that you never have to change them, but if you want to, they are all put in plain view inside two control files, *LINE* and *LOOP CTRL*. These files can be directly edited from within the exec. This is nice because the *CTRL* files are annotated and you need not remember the exact spelling of the

mnemonics. However, you must be careful because the fortran reads the file using tab commands. This explains the prof' sion of brackets. In short, all data must be contained within the d' ameters. The best approach to inputting new values to these parameters are the full screen menus. Their input fields automatically delimit the formats properly, plus they display the mnemonics in plain view. And they're fast. The one disadvantage is that they cannot be used from all terminals. The last approach is the one adopted in this example. It involves using the ctrl option of *LINE* . This option is very fast as well; but it demands that you know the option names. It first prompts you for wheter *LINE* or *LOOP* control parameters are to be changed. Then it puts you into a loop, prompting you for the name of the variable to be adjusted, and its new value. When you are finished adjusting parameters you just type in "end".

Do you want line, loop, or optn (end to quit, cr=help)?

-> ctrl

Here we go directly to the ctrl optn, although we could have also entered optn and gotten another, more explicit, prompt.

Do you want LINE or LOOP ?

-> line

Enter variable name and new value (end to quit).

-> arfac 1.0

Enter variable name and new value (end to quit).

-> end

In this instance we only adjust the arrow position for lines, which is done by setting the variable ARFAC to 1.0. Normally it is 0.5, and this puts the arrow half way down. All the effects of these parameters are contained in an online help program as well as in the appendix; only those parameters relevant to the above diagrams will be discussed here. The control files with default values can be found in the appendix.

Save defaults file in output (yes/no/cr=no) ?

-> no

This feature can be quite useful if you customize a very complicated line. It allows all the defaults to be stored in the Topdraw file so that you

may refer to them later. We won't be taking advantage of this feature now.

Do you want line, loop, or optn (end to quit, cr=help)?

-> line#ferm#2#3#2#5#no#line#phot#1#4#2#4#yes

This line of input is sufficient to create the lowest order graph.

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

FERM

INPUT POINT X1:

2.000

INPUT POINT Y1:

3.000

INPUT POINT X2:

2.000

INPUT POINT Y2:

5.000

Do you want to see graph (yes/no/cr=no) ?

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

1.000

INPUT POINT Y1:

4.000

INPUT POINT X2:

2.000

INPUT POINT Y2:

4.000

Do you want to see graph (yes/no/cr=no) ?

Do you wish to keep this (yes/no/cr=yes)?

-> yes

So far so good, the next diagram is the same, except for the addition of a loop.

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> line#ferm#4#3#4#5#no#line#phot#3#4#4#4#no

This is the first graph translated over by 2 units.

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

FERM

INPUT POINT X1:

4.000

INPUT POINT Y1:

3.000

INPUT POINT X2:

4.000

INPUT POINT Y2:

5.000

Do you want to see graph (yes/no/cr=no) ?

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

3.000

INPUT POINT Y1:

4.000

INPUT POINT Y2:

4.000

INPUT POINT Y2:

4.000

Do you want to see graph (yes/no/cr=no) ?

line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> loop#1#phot#4#4.5#4#3.5#yes

Now we make the loop. We need to ask for the loop twice, by "loop" and then "l", i.e. loop#1 in the above, in order to distinguish against the possibility of making a bubble. The next four entries are the coordinates of the loop. Please notice that the loop is to the *left* of the direction of travel from starting point to finishing. The actual separation of the endpoints is one unit, which is the same as the length of the external photon line. This is no accident. Because of the scaling of amplitudes and wavelengths inherent in the program algorithms, it is best to keep all lines approximately the same length throughout the drawing. Of course, you can adjust the amplitudes and wavelengths in the control files, but most graphs allow enough symmetry so you do not have to resort to this.

Loop is the choice, begin executing loop fortran.

EXECUTION BEGINS...

DO YOU WANT A LOOP OR A BUBBLE? (L OR B, FORMAT A1):

LOOP

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

4.000

INPUT POINT Y1:

4.500

INPUT POINT X2:

4.000

INPUT POINT Y2:

3.500

Do you want to see graph (yes/no/cr=no) ?

Do you wish to keep this (yes/no/cr=yes)?

-> yes

And it works out quite nicely.

Loop is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> line#ferm#8#2.5#8#5.5#no

Because of all the complications hanging off of this third fermion line, it is necessary to make it a little longer than the other two.

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A6, ? FOR HELP)

FERM

INPUT POINT X1:

8.000

INPUT POINT Y1:

2.500

INPUT POINT X2:

8.000

INPUT POINT Y2:

5.500

Do you want to see graph (yes/no/cr=no) ?

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> ctrl#loop#iarrot 1#end

Notice that the arrows on the fermion loop are rotated off the horizontal, yet if you look ahead you see that the endpoints used for the bubble are on the vertical. How is this? It is because the defaults weren't used. This time, a variable called IARROT (arrow rotation) is turned on (set to 1) inside LOOP CTRL. This allows you to use a simple coordinate scheme to enter the lines (and loops, etc.), while moving the arrows out of the way.

Save defaults file in output (yes/no/cr=no) ?

-> no

Do you want line, loop, or optn (end to quit, cr=help)?

-> loop#b#ferm#7#3.5#7#4.5#yes

Again, the endpoint coordinates are simple.

Loop is the choice, begin executing loop fortran.

EXECUTION BEGINS...

DO YOU WANT A LOOP OR A BUBBLE? (L OR B, FORMAT A1):

BUBBLE

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

FERM

INPUT POINT X1:

7.000

INPUT POINT Y1:

3.500

INPUT POINT X2:

7.000

INPUT POINT Y2:

4.500

Do you want to see graph (yes/no/cr=no) ?

Do you wish to keep this (yes/no/cr=yes)?

-> yes

Loop is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> line#phot#5.5#4#6.5#4#yes#yes

And now the photon hangs off an internal line. The fermion loop is 1 unit in diameter which makes locating the photon's endpoint easy. The advice being- keep the units simple. Use integers and half-integers if possible.

Line is the choice, begin executing line fortran.

EXECUTION BEGINS...

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

5.500

INPUT POINT Y1:

4.000

INPUT POINT X2:

6.500

INPUT POINT Y2:

4.000

Do you want to see graph (yes/no/cr=no) ?

Do you wish to keep this (yes/no/cr=yes)?

Line is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> ctrl#loop#ang 90.0#end

This last modification allows us to obtain the desired graph versus the one shown below. Inside LOOP CTRL there is a variable ANG which controls the sagitta of the loop (this variable is not used when making bubbles). Normally, it is set to 180.0° , but here we do not want this much curvature. Instead, we set ANG to 90.0° , which is just one half 180.0° since each loop takes its share of 180.0° . Note that the variables in the control files need to have decimal points if they are not character or integer, thus the input is 90.0 above.

Save defaults file in output (yes/no/cr=no) ?

-> no

Do you want line, loop, or optn (end to quit, cr=help)?

-> loop#1#phot#7#4.5#8#5#yes

Here again, the coordinate scheme is simple. The endpoints join up in a straightforward fashion.

Loop is the choice. begin executing loop fortran.

EXECUTION BEGINS...

DO YOU WANT A LOOP OR A BUBBLE? (L OR B, FORMAT A1):

LOOP

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

7.000

INPUT POINT Y1:

4.500

INPUT POINT X2:

8.000

INPUT POINT Y2:

5.000

Do you want to see graph (yes/no/cr=no) ?

Do you wish to keep this (yes/no/cr=yes)?

-> yes

Loop is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> loop#1#phot#8#3#7#3.5#yes

This time the left hand rule tells us to start the loop on the fermion line.

Loop is the choice, begin executing loop fortran.

EXECUTION BEGINS...

DO YOU WANT A LOOP OR A BUBBLE? (L OR B, FORMAT A1):

LOOP

WHAT KIND OF LINE IS DESIRED? (FORMAT A5, ? FOR HELP)

PHOT

INPUT POINT X1:

8.000

INPUT POINT Y1:

3.00

INPUT POINT X2:

7.000

INPUT POINT Y2:

3.500

Do you want to see graph (yes/no/cr=no) ?

Do you wish to keep this (yes/no/cr=yes)?

-> yes

Loop is done.

Do you want line, loop, or optn (end to quit, cr=help)?

-> end

Session is over, output file is in EXAMPLE3 TOPDRAW

R;

Now you are hopefully educated enough to make almost any figure you desire with *LINE*. While the discussion of the control features is admittedly scant, these features are documented online as well as in the appendix, and you will find them as you need them. The geometries of the lines can be individualized extensively, or the defaults can be exclusively used, depending on your preferences. Never fail to remember that your output is a Topdraw file and can be edited accordingly. This not only allows you to put titles and labels in the file, but also allows you to remove unwanted features at any time. For instance, the arrows are just titles in Topdraw that use the gradient symbol (which

unfortunately necessitates your filling them in with ink if you like solid arrows). At this point you have mastered the machinery sufficiently to learn how to use it in its most powerful form, with the *LINE* menu option, see the next section.

Line Menus

By invoking the *LINE* exec with the argument menu you can forego the constant prompting of the exec and have all the options displayed on your screen at once. This style of operation is by far and away the most powerful. For instance, since you have the option of not erasing the input from the previous line (or loop), you can save yourself the trouble of repeating duplicate responses (e.g. see example 4). The exec keeps you informed of the execution status without ever having to leave the menu. This lets you operate the program in a very efficient manner. Furthermore, the PF keys are programmed to allow you great control over your environment, as well as the control parameters for the program. You are able to enter xedit, use abacus, get help, look at other menus, or end the session just with the touch of a key. This gives you great versatility. For instance, the parameters of the lines and loops can be changed in seconds in the control menus. By utilizing the input area you can put titles into your output file without ever having to edit it. Furthermore, this input area can also be used to enter CMS or CP commands without leaving the exec (e.g. to clear some disk space, to send messages, to reference other files, etc.). When you hit the enter key the exec then stacks your answers into its buffer and executes the fortran internally. You need never leave the menus unless the exec becomes confused (e.g. by incorrect input). Then the exec switches back into prompt mode. But you can recover from this by asking to reenter the menu mode of operation (and correcting your input). The best advice for learning about the menu feature is to try it by typing "line menu"; the rest of its operation should be self-explanatory.

With the exception of the stack option. This option is used for storing your output file in a very condensed form. Instead of saving a Topdraw file with thousands of numbers in it, the stack option creates an exec (with the same filename as the output Topdraw file) that has all your commands in it. So, each time you draw a line from the menu with the stack option, your commands are saved in the exec. After you generate a hard copy of the Topdraw plot you can discard the bulky Topdraw file and save the stacked exec. This exec is capable of generating the original Topdraw plot in seconds, including the titles entered from the input area, and all your changes to the control parameters. Thus, the exec is truly a way to store your Topdraw file in condensed form, with a tremendous savings of space. The stack exec will have the same filename as your output file, and you can execute it as you would any ordinary exec. It stacks up all the commands from your session and then calls the *LINE* exec with the special stack option. The exec will work silently regenerating the original topdraw file, and may require a little time to redraw a complicated graph. However, it should be much faster than having to duplicate the original session.

Some details of the stack operation should be mentioned here. The stack option must be on (i.e. "yes") throughout any portion of the session in order to save it. As long as it is on it will save subsequent entries to the output file, even those from the input area. However, if you decide to erase the last entry during the session, the stacked exec file will have that entry deleted from it, much like the Topdraw file. The actual operation of the commands in the exec file should be quite clear since they bear such a direct relation to the commands you enter. The different lines and loops are also separated by comment statements for readability. As you become more skilled with *LINE* you will also realize that you can directly modify this file in order to recreate a modified session, and thus save time. In fact, you can even create a new topdraw file from a stacked exec file, and forego even the prompting and menus of *LINE*.

One word of warning when using the stack option; *LINE* exec expects the stacked exec file to be newly created during a session. If an exec with the same name already exists the session's output is appended to this exec file. You might therefore get unexpected results when subsequently running the stacked exec. However, the *LINE* exec will inform you during the session if this situation occurs. For completeness it should be added that the stacked exec will create a Topdraw plot with the same name as the original session's.

Bhabha Scattering

Example 4

This example serves to demonstrate running *LINE* with menus. It also shows *LINE*'s ability to label the graphs internally. Because both these features are new, and especially in order to contrast the line by line mode with the menu mode, this example simply redraws the first example's graph †.

There are three menus used in *LINE*, and all of them are organized the same. Data is entered, or changed, within fields defined for each menu. The fields are delimited parentheses. The function of the PF keys are denoted at the bottom of the menu, and help files are available for every menu. Each menu is quit by way of the PF03 key. The entire session is quit if the PF03 key is used in the master menu. The *LINE* exec's master menu has the greatest capacity, and the user is meant to run the session from it, with occasional forays into the *LINE CTRL* and *LOOP CTRL* menus. Because the menus are dynamically controlled and have so many branch points, this manual cannot completely demonstrate their capabilities. Instead, the following example serves merely as an indication of their abilities.

The complete session is not shown for this example. For instance, the menu on the next page is gotten only after entering the name of the output file, taken to be **EXAMPLE4** here. The menus are shown at selected times during the session. I should mention that the PF keys are extremely important for operating the menus and are described in the appendixes. In the following the annotations are in roman font, the menus are in the usual typewriter font, and your input are in boldface font (e.g. **line**).

† This time the final state is chosen to be electrons, however.

===== Line Fortran Control Menu =====

Data card 1: Default control

IDEF(0)

Data card 2: Line control

WIGL(5.50) AMP(0.10) ASP(2.20)

Data card 3: Arrow control

ARFAC(0.50) DELSIZ(2.25) IARROW(1)

Data card 4: Free line control

FLEN(1.40) DLEN(0.30)

Data card 5: Scalar and composite line control

DUTY(0.50) DBLSEP(0.022)

Data card 6: X-particle control

XSIZE(1.00) XSEP(0.120) XCHAR(1) XCASE(0)

Data card 7: Misc control

CONIC(1.00) HWIDTH(1.00) HTAIL(1) AXCTRL(1)

Data card 8: Title control

TOFF(0.75) TSIZE(2.00) TANGLE(0.) TFONT(D)

Data card 9: Additional title control

IPOS(9) TITLE(e2-3,1p) CASE(lx x ul)

1= File 2= Item Help 3= Quit 4= Restore Defaults 5= Menu Help

6= Output Line Ctrl 7= Abacus 8-12 Unused PF keys

This is the LINE CTRL menu after being modified. Normally, the value of IPOS is 0 (i.e. no titles), TFONT is N (i.e. no font command), while TITLE and CASE are blank. Also, the offset from the line, TOFF, has been increased from 0.50 to 0.75 to accomodate the length of the title. Here we have put the label in position 9. The positioning algorithms are described in Appendixes II and V. Basically, IPOS numbers positions going counter-clockwise around the line. The title is from the Unified Graphics standard character set. TFONT is changed to D in order to give us the duplex font. We use PF01 to leave this menu in order to file the changes (PF03 quits without updating LINE CTRL). This then returns us to the main menu.

```
===== Line Exec Master Menu =====
```

Type of line or loop (ferm)	Execution begins...
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

```
Input coordinates....  X1 ( 2      )
```

X2 (4)

Y2 (4)

Do you want to view the graph ? (no)

Stack option (yes)

Press enter to draw the above

1= Help 2= Clear 3= Quit 4= Line Ctrl Menu 5= Loop Ctrl Menu 6= Abacus
7= Xedit Output 8= Help Ctrl Menu 9= Plot 10= Xedit Stack 11= Next 12= CMS Input

Now the line has been generated and both the output file and the stacked exec file have been updated. The messages appearing here (e.g. Execution begins..., etc.) inform you of the status of execution. They are typically used to notify you of execution completion, although they are also used as error messages on occasion. At this point the LINE CTRL menu is used to set IPOS to 11 and TITLE to "e2+3,1q". Both TOFF and CASE remain the same as for the previous line. We do not bother to show the LINE CTRL menu again.

The outline border was then entered from this input area in a similar fashion. The job is finished by using PF07 to edit the file and set the topdraw axes off. Next, PF09 was used to make a hard copy of the topdraw plot. This was because *LINE* was run with the exec output option, see Appendix X. And finally, PF03 was used to end the session. For completeness I should mention that the input area can be used to issue commands directly to CMS by using the PF12 key. Thus, you can type "q disk", and hit PF12 in order to see how full your disks are. Other uses are hopefully obvious.

The two graphs above show the results of two sessions with *line*. They differ only in the choice of position of the titles. This involved only changing the choice of IPOS. They are included as demonstrations of *LINE*'s versatility. The graph on the left's stacked exec is on the following page. Note that it is written in the REXX language, as is *LINE*. Hence, the */* */* entries are comments. Hopefully, the functioning of each statement is readily apparent.

EXAMPLE4 EXEC FILE

```
/*  
  LINE EXEC OUTPUT FOR BWL   CREATED  18:54:42 ON 02/21/85  
*/  
queue  EX4  
queue  CTRL$SET  
queue  LINE  
queue  TOFF " 0.75"  
queue  IPOS " 9 "  
queue  TITLE "E2-3,1P"  
queue  CASE "LX X UL"  
queue  TFONT "D"  
queue  CTRL$END  
/* */  
queue  LINE  
queue  FERM  
queue  2  
queue  2  
queue  4  
queue  4  
queue  NO  
/* */  
queue  CTRL$SET  
queue  LINE  
queue  IPOS "11"  
queue  TITLE "E2+3,1Q"  
queue  CTRL$END  
/* */  
queue  LINE  
queue  FERM  
queue  4  
queue  4  
queue  2  
queue  6  
queue  NO  
/* */  
queue  CTRL$SET  
queue  LINE  
queue  TOFF " 0.50"  
queue  IPOS " 10"  
queue  TITLE "G1,1P+Q"  
queue  CASE "GU UL L"  
queue  CTRL$END  
/* */  
queue  LINE  
queue  PHOT  
queue  4  
queue  4  
queue  6  
queue  4  
queue  NO  
/* */
```

```

queue CTRL$SET
queue LINE
queue TOFF " 0.75"
queue IPOS " 11"
queue TITLE " E2-3,1P'"
queue CASE " LX X UL"
queue CTRL$END
/* */
queue LINE
queue FERM
queue 6
queue 4
queue 8
queue 2
queue NO
/* */
queue CTRL$SET
queue LINE
queue IPOS " 9 "
queue TITLE " E2+3,1Q'"
queue CTRL$END
/* */
queue LINE
queue FERM
queue 3
queue 6
queue 6
queue 4
queue NO
/* */
queue CMD$START
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING TITLE 3 7.5 DATA SIZE 2.5 'BHABHA SCATTERING'"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING CASE ' LLLLL LLLLLLLL'"
queue CMD$END
queue CMD$START
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING 1 7"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING 9 7"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING JOIN"
queue CMD$END
queue CMD$START
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING 1 1"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING 1 7"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING JOIN"
queue CMD$END
queue CMD$START
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING 1 1"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING 9 1"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING JOIN"
queue CMD$END
queue CMD$START
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING 9 7"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING 9 1"
queue "EXECIO 1 DISKW EX4 TOPDRAW A (STRING JOIN"
queue CMD$END
queue END
call LINE 'STACK'

```

Appendix I: LINEDEF CTRL FILE†

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
Data card 1: Default Control
  < >
  IDEF <0>
  < >
Data card 2: Line control
  < > < > < > < >
  WIGL < 5.50 > AMP < 0.10 > ASP < 2.20 > IPHI <0>
  < > < > < > < >
Data card 3: Arrow control
  < > < > < >
  ARFAC < 0.50 > DELSIZ < 2.25 > IARROW <1>
  < > < > < >
Data card 4: Free line control
  < > < >
  FLEN < 1.40 > DLEN < 0.30 >
  < > < >
Data card 5: Scalar and composite line control
  < > < >
  DUTY < 0.50 > DBLSEP < 0.022 >
  < > < >
Data card 6: X-particle line control
  < > < > < > < >
  XSIZE < 1.00 > XSEP < 0.120 > XCHAR <1> XCASE <0>
  < > < > < > < >
Data card 7: Misc control
  < > < > < > < >
  CONIC < 1.00 > HWIDTH < 1.00 > HTAIL <1> AXCTRL <1>
  < > < > < > < >
Data card 8: Title control
  < > < > < > < >
  TOFF < 0.50 > TSIZE < 2.25 > TANGLE < 0 > TFONT <N>
  < > < > < > < >
Data card 9: Additional title control
  < > < > < > < >
  IPDS < 0 > TITLE< > CASE< >
  < > < > < > < >
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...

```

† The lines with the column markings do not belong to the file, they are used here to delimit the actual file records, and to locate the column positions.

Appendix II: Line Fortran Control Parameters

Here are the important parameters for the lines made with *LINE* Exec.

- IDEF - this parameter is used to control the source of the values of the following parameters. If it is 1 then the fortran uses an internal data statement and the default values are exclusively used. If it is 0 then the *LINE* CTRL file is read in. $IDEF = 0,1$.
- WIGL - is obviously the number of wiggles. $WIGL > 0$.
- AMP - gives the amplitude for WORZ lines and PHOT lines, (however, things are a little more tricky for the gluon lines because of its more complicated structure; there WIGL helps determine the amplitude).
 $0.001 < AMP < 10$.
- ASP - is the aspect ratio of the two circles which make the glue. (Controls tightness of loops). $0.001 < ASP < 100$.
- FLEN - is used for the free photon, gluon and worz lines. FLEN gives the length of the wiggles. $FLEN < 10$.
- DLEN - is also used for the free photon, gluon and worz lines. DLEN gives the length of the straight line section with arrow. $DLEN < 10$.
- IARROW - flags for arrows for photon, gluon, and worz lines. It places the arrow at the midpoint of the line, depending on the following variable.
 $IARROW = 1$ (arrow on) $IARROW = 0$ (arrow off).
- ARFAC - this places the arrow on the line (including fermions) a fraction ARFAC from the starting point. Normally this is 0.5 but it can be adjusted between 0 and 1. $0 < ARFAC < 1$.
- DELSIZ - this gives the size of the arrow. $0 < DELSIZ < 5$. (Topdraw title size).
- DUTY - this is the on fraction for the scalar dotted line. $0 < DUTY < 1$.
- DBLSEP - is the distance between the double lines for a composite particle line. $DBLSEP < 10$.
- XSIZE - this sets the size of the character for an X-particle line. $0 < XSIZE < 5$. (Topdraw title size).
- XSEP - determines the separation between characters in an X-particle line. $0.001 < XSEP < 1$.

- XCHAR - gives the user complete control over the choice of character for an X-particle. This is the Unified Graphics code as is standard in Topdraw.
- XCASE - This is the Unified Graphics case specification.
- CONIC - This parameter controls all the conic sections. For ellipses it is the eccentricity. For parabolas it is the coefficient of the quadratic term. For hyperbolas, (which actually should require two control parameters) it is the tangent of the opening angle between the asymptotes. (The hyperbolas are adjusted so that their slope at the endpoints with 20% of the asymptotic value). $CONIC > 1.E - 6$.
- HWIDTH - controls how wide the helicity arrow is. The arrowhead width also scales with this parameter. $0 < HWIDTH < 10$.
- HTAIL - is used to delete ($HTAIL = 0$) the tail of the helicity arrow. $HTAIL = 0, 1$.
- AXCTRL - controls the labeling of the axes. Its value modulo 3 determines which axis points along the input direction (which is cyclical starting with x). For values ≤ 3 the labels are simply x , y , and z ; for values $\leq 2 \times 3$ they are x' , etc.; for values $\leq 3 \times 3$ they are $\tilde{x}$, etc. $AXCTRL = 1, 2, \dots, 9$.
- TOFF - this parameter controls how far off the lines and loops the labels are. Note that the labels use the Topdraw center option for titles. The units for TOFF are in terms of the DATA coordinates. For the default title size (see below) the value of a character size is approximately 0.25. $0 < TOFF < 10$
- TSIZE - is the standard Topdraw title size parameter. $0 < TSIZE < 5$.
- TANGLE - is the angle at which the Topdraw title is set. $0 < TANGLE < 360$.
- TFONT - is the font choice. Its value must be one of the following: N (the default) means to not set the font, B is for the basic font set (not typically used), E is the extended font set (the Topdraw default), D is for the duplex font (best quality, slowest output time).
- IPOS - controls the position of the title relative to the line drawn (together with TOFF). For lines it goes around counterclockwise starting at the endpoint of the line. There are twelve positions; 1 and 7 are at the endpoints, 4 and 10 are the midpoints. The line is divided into fifths. See Appendix V. $IPOS = 1, 2, \dots, 12$.

- TITLE - is the Topdraw title in terms of the Unified Graphic's character set. No quotes are necessary, the title is to be entered directly.
- CASE - is the Topdraw title case in terms of the Unified Graphic's character set. No quotes are necessary, the title case is to be entered directly.

The above range of parameters are overly generous; the values should always be near the default (except for flags and characters). All variables which are not integer or character need to have a decimal place present.

Note added

- IPHI - This variable is used to reverse the phase of the oscillation for photons and worz particles. For gluons it reverses the direction of the pigtaills. This feature is used to keep the lines from overlapping other lines, and is mostly used for the free photon, etc., lines. $IPHI = 0,1$ (0 is the default).

Appendix III: LOOPDEF CTRL FILE†

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
Data card 1: Default control
    < >
    IDEF <0>
    < >
Data card 2: Angle control
    < >
    ANG < 180.0 >
    < >
Data card 3: Line control
    < > < > < >
    WIGL < 7.00 > AMP < 0.07 > ASP < 2.20 >
    < > < > < >
Data card 4: Arrow control
    < > < > < >
    DELSIZ < 2.25 > IARROT <0> IARROW <1>
    < > < > < >
Data card 5: Scalar and composite line control
    < > < >
    DUTY < 0.50 > DBLSEP < 0.022 >
    < > < >
Data card 6: X-particle line control
    < > < > < >
    XSIZE < 1.00 > XSEP < 0.100 > XCHAR <1> XCASE <0>
    < > < > < >
Data card 7: Title control
    < > < > < >
    TOFF < 0.50 > TSIZE < 2.00 > TANGLE < 0. > TFONT <N>
    < > < > < >
Data card 8: Additional title control
    < > < > < >
    IPOS < 0 > TITLE< > CASE< >
    < > < > < >
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...

```

† The lines with the column markings do not belong to the file, they are used here to delimit the actual file records, and to locate the column positions.

Appendix IV: Loop Fortran Control Parameters

Here are the important parameters for the loops made with *LINE* Exec.

- IDEF - this parameter is used to control the source of the values of the following parameters. If it is 1 then the fortran uses an internal data statement and the default values are exclusively used. If it is 0 then the *LINE CTRL* file is read in. $IDEF = 0,1$.
- ANG - is the angle of the loop on a circle. $0 < ANG < 360$. (Decrease for lower saggitta).
- WIGL - is obviously the number of wiggles, or at least almost always; the general rule is that increasing WIGL increases the number of oscillations. $WIGL > 0$.
- AMP - gives the amplitude for WORZ and PHOT oscillations (however things are a little more tricky for the gluon lines because of its more complicated structure; there WIGL helps determine the amplitude).
 $0.001 < AMP < 10$.
- ASP - is the aspect ratio of the two circles which make the glue (controls tightness of loops). $0.001 < ASP < 100$.
- IARROW - this flag turns the arrow(s) on except for photons, worz's, and gluons (which never have arrows). One arrow for loops, two for bubbles.
 $IARROW = 1$ (on), $IARROW = 0$ (off)
- IARROT - is a flag which rotates the arrows on bubbles with arrows. This is useful for $g-2$ graphs, etc. $IARROT = 1$ (on), $IARROT = 0$ (off).
- DELSIZ - gives the size of the arrows drawn. $0 < DELSIZ < 5$. (Topdraw title size).
- DUTY - this is the on fraction for the scalar dotted line. $0 < DUTY < 1$.
- DBLSEP - is the distance between the double lines for a composite particle line.
 $DBLSEP < 10$.
- XSIZE - this sets the size of the character for an X-particle line. $0 < XSIZE < 5$. (Topdraw title size).
- XSEP - determines the separation between characters in an X-particle line.
 $0.001 < XSEP < 1$.

- XCHAR** - gives the user complete control over the choice of character for an X-particle. These are the Unified Graphics code as is standard in Topdraw.
- XCASE** - is the Unified Graphics case argument.
- TOFF** - this parameter controls how far off the lines and loops the labels are. Note that the labels use the Topdraw center option for titles. The units for TOFF are in terms of the DATA coordinates. For the default title size (see below) the value of a character size is approximately 0.25.
 $0 < \text{TOFF} < 10$
- TSIZE** - is the standard Topdraw title size parameter. $0 < \text{TSIZE} < 5$.
- TANGLE** - is the angle at which the Topdraw title is set. $0 < \text{TANGLE} < 360$.
- TFONT** - is the font choice. Its value must be one of the following: N (the default) means to not set the font, B is for the basic font set (not typically used), E is the extended font set (the Topdraw default), D is for the duplex font (best quality, slowest output time).
- IPOS** - controls the position of the title relative to the loop drawn (together with TOFF). For loops it goes around counterclockwise starting at the endpoint of the input line segment. The number of positions depend on whether a loop or a bubble is being drawn. For a loop there are nine positions, and the loop segment is divided into 8 pieces. Position 1 is at the endpoint, 9 is at the starting point, and 5 is at the midpoint. For a bubble the numbering scheme is based on a eightfold symmetry (i.e. $\pi/4$). The numbering scheme wraps around the circle twice; 1-8 have polar angle $\frac{n}{4}\pi$ while IPOS 9-16 have angle $\frac{n-8}{4}\pi - \frac{1}{8}\pi$ (which are related to positions 1-8 by a rotation of $\frac{1}{8}\pi$). See Appendix V. IPOS = 1,2,...,16.
- TITLE** - is the Topdraw title in terms of the Unified Graphic's character set. No quotes are necessary, the title is to be entered directly.
- CASE** - is the Topdraw title case in terms of the Unified Graphic's character set. No quotes are necessary, the title case is to be entered directly.

The above range of parameters are suggestions only, the values should always be near the default (except for flags and characters). All variables which are not integer or character need to have a decimal place present.

Appendix V: Label Positioning for Line and Loop†

† The numbers shown here are the values of IPOS.

Appendix VI: Line Initialization and Default Settings

LINEDEF	MENUCTRL	}	Control files
LOOPDEF	MENUCTRL		
LINE	CTRL		
LOOP	CTRL		
ONETIME	TOPDRAW	}	Topdraw files
LTEMP	TOPDRAW		
GETDEF	XEDIT	}	Xedit macro

Table I: Temporary files used by *LINE*.

The *LINE* exec begins by searching for files which it needs to operate. These include the fortran text files for Line and Loop (or if they are missing, the fortran files themselves) as well as LINEDEF and LOOPDEF CTRL. The latter files contain the default settings of all the control parameters. They must be present, either on the user's disks, or on the default disk (which is presently the W-disk). Only those disks for which the user has read/write access are searched to ensure execution of the proper routines. The user's disks are searched first, in case the user wishes to have their own permanent control files. If either the plotting or fortran options are used (see Appendix X), then *LINE* will search through the user's disks for the input files.

LINE also searches for several files on the user's A-disk which shouldn't be there. These files are used as scratch files and are listed in Table I above. If the exec finds any of these files it prompts the user to allow it to erase them. If the user does not give their approval the exec exits. The control files, LINE and LOOP CTRL, are copied from the master default files, LINEDEF and LOOPDEF CTRL. The menu control files, LINEDEF and LOOPDEF MENUCTRL, are also created from LINE and LOOPDEF CTRL. Changes to the control parameters made during the session are stored in LINE and LOOP CTRL, since these are the files that are actually used by the fortran programs. These are also the files which are modified by the user (e.g. when in LINE MENU or when editing). At the end of the session the CTRL files are erased, as are the MENUCTRL files. If the user

wishes to save a copy of the CTRL files used to make a line (or loop), they can dump a copy of them to the output file. The user can also use the edit option to copy the files wherever they wish. However, the preferred method of storing any default changes for individual lines is to use the stack option from the full screen menu.

The GETDEF XEDIT file is a xedit macro which is used for getting the default control values from LINEDEF and LOOPDEF CTRL and storing them in internal arrays inside the exec. The CTRL files have a very fixed format, and the fortran programs demand that the values be stored in definite columns in the CTRL files. The angular brackets used throughout the CTRL files serve as important delimits (see Appendices I and II). The GETDEF macro uses the angular brackets for parsing the input values. It is copied onto the A-disk for the session, and is erased at the end.

The other temporary files are used to enable the exec to erase misplaced lines and loops. Their function is as follows: for each line drawn a copy of it, as a single line, is saved in ONETIME TOPDRAW. If the user wishes to view the plot of the file built up to that moment, then the ONETIME file, and the permanent output file, are merged into a temporary file, LTEMP TOPDRAW. This file is the plotted, either internally, or with an optional external procedure supplied by the user. Upon viewing the plot, the user can then decide whether ONETIME should be appended to the output file. If not, then the session continues and ONETIME is overwritten, otherwise it is instead copied to the output. If the user decides not to view the graph then ONETIME is directly appended to the output file upon the fortran program's completion. At the end of the session both ONETIME and LTEMP TOPDRAW are erased.

Appendix VII: LINE and LOOP CTRL Menu PF Keys

The following list shows the function of the PF keys for the above menus.

- PF01 - is used to update the values of the control parameters in the CTRL files. The values on the screen are stored in the control files and effect any subsequent output (until the menu is again used). This key, and PF06, are the only keys which update the control file. PF01 then returns to either the master menu, or directly to the *LINE* exec (when in prompt mode)†.
- PF02 - enters a CMS help file which describes the role of the different control parameters. These help files are similar to Appendixes II and IV.
- PF03 - this serves to quit the menu *without* updating the control parameter values. The program then returns either to the master menu or to the line by line prompting.
- PF04 - is used to restore all the original default values. These are the values which were initially gotten from LINEDEF or LOOPDEF CTRL. These values are stored in the MENUCTRL files throughout the session. Note that the control parameter IDEF serves a similar function in some cases.
- PF05 - enters a CMS help file which describe the role of LINE and LOOP CTRL menus under the *LINE* exec (e.g. how they are used to interface the control parameters to the fortran programs). It is very basic in its description.
- PF06 - serves a similar function as PF01 in that it also updates the CTRL files with new values of the control parameters. It also returns from the menu in a similar fashion. It differs in that the updated CTRL file is written to the output file in its entirety as Topdraw comments. This is useful for storing the control setting but is somewhat cumbersome; therefore, the recommended procedure is to use the stack option in the master menu.
- PF07 - this is the last PF key which functions in these menus. It simply transfers program control to Abacus, SLAC's very powerful online calculator. When Abacus is quit (by entering quit or exit), program control is transfered back to the menu.

† A word of warning to users in prompt mode who change values from via menus. Since you have the option of directly editing the control files let it be known that filing the menu values away will change the CTRL files, however, editing LINE or LOOP CTRL will not update the menus. This is not a problem if you modify the control parameters in a consistent fashion.

Appendix VIII: LINE EXEC Master Menu PF Keys

The following list shows the function of the PF keys for the *LINE* exec master menu. The function of the ENTER key is described at the end. Note: PF01-12 ↔ PF13-24.

- PF01 - enters a CMS help file which describes the operation of *LINE* from this menu.
- PF02 - is the clear key. It simply clears all the fields on the screen, so that the initial menu is displayed again.
- PF03 - this serves to quit the menu and end the session with *LINE* . The output file is left on the user's A-disk.
- PF04 - transfers control to LINE CTRL menu, allowing the user to change the control parameters for LINE fortran. Notification of changes are made upon return to the master menu. Note that the stack option must be on *before* the key is used if the changes are to be saved.
- PF05 - serves the same role for LOOP as PF04 does for LINE.
- PF06 - transfers program control to Abacus, SLAC's very powerful online calculator. When Abacus is quit (by entering quit or exit), program control is transferred back to the menu.
- PF07 - xedits the output file. Program control returns to the menu.
- PF08 - enters a CMS help file describing the role of the LINE and LOOP CTRL menus. It is the same file gotten by PF05 in the CTRL menus.
- PF09 - is used to plot the output file in its current state. If no plot exec has been given *LINE* as an option (see Appendix X), then an internal procedure is used to create a Topdraw 4013 sequential file which is then displayed at the user's terminal with GMPLLOT. Otherwise, the user's exec is called with the filename and filetype of the output file as arguments.
- PF10 - xedits the exec file being created by the stack option.
- PF11 - readies the menu for the next set of inputs. All the coordinate pairs (vertexes) entered through the menu are stored in an internal array; PF11 rolls back through the array putting the previous coordinate pairs into the X1 and Y1 fields and clearing the X2 and Y2 fields for the next vertex. The cursor is then positioned on the X2 field as a convenience It should be mentioned that if PF11 is hit enough

times the array is cycled through again. Also note that PF11 is repositioned at the top of the array each time the master menu is entered.

PF12 - is used to pass, as literals, the data entered in the direct input area to CMS. Thus, the user is able to branch control to other processes while in the exec (e.g. to send messages to other users). When this key is used the data is cleared from the direct input area.

The last key which functions in the menu is the ENTER key. It has a dual role depending on whether any data is present in the direct input area. If data is present the exec ignores any input from other fields and processes the direct input area fields instead. *LINE* takes the data, as literals, and appends it to the output Topdraw file. It also puts commands to do this into the stacked exec, if the stack option is on. The fields are cleared and notification is given the user that the outfile has been modified.

If no data is present in the direct input area, *LINE* then precedes to process the other input fields. A certain amount of error checking is done. The first error halts processing and *LINE* returns an error message and waits for another ENTER, giving the user a chance to correct the error. If no error is found then the exec builds an internal stack from the responses and then executes the fortran, optionally displays the plot, etc. As each step is completed messages are sent to the user. The menu remains displayed throughout execution.

Appendix IX: Default Topdraw Axes

```
( LINE EXEC OUTPUT FOR BWL   CREATED   15:30.44 ON 02/17/85
(
( DEFAULT AXES CTRL CARD
(
SET WINDOW X 2.0 10.0 Y 1.0 9.0
SET LIMITS X 0 10
SET LIMITS Y 0 10
SET SCALE  Y 10 -5
SET SCALE  X 10 -5
( SET AXES OFF
( SET SIZE 13 BY 10 REDUCE = 4.0
(
PLOT AXES
```

The above Topdraw commands are used to define the default axes used by *LINE*. The first line is a notification of when the file was created by the user. The left parenthesis is a Topdraw comment control. The line with "DEFAULT AXES CTRL CARD" is inserted as a tag in case this file is appended by *LINE* in future sessions. *LINE* inserts the above axes at the top of the output file unless it finds the default card in the file already. The coordinate system is set up to be 10 x 10 (by SET LIMITS and SET SCALE) units in a 8 by 8 window. The window was chosen to be square so that the bubbles look like circles. If the user wanted their bubbles to be ellipses with semi-major axes along the x-axis they could widen the dimensions of the window in x; similarly for y. The PLOT AXES command is explicitly used to guarantee that the axes are plotted in extended font. Otherwise, the axes might be plotted in duplex font. This would unnecessarily require more time for the graphic device to display. The SET AXES OFF command will force the axes to *not* be displayed even with the PLOT AXES command present. Thus, the user merely need remove the parenthesis from this command to eliminate the axes. The last command which has been commented out determines the absolute size of the plot. The window size has been set to 8 by 8, and the size of the plot is by default 8 by 8 inches. However, the SET SIZE 13 BY 10 REDUCE = 4.0 command will reduce the absolute size of the plot (for versatec and imagen output) to 2 by 2 inches (which happens to be the size of the small plots in this manual). This command has been included to save the user from having to make reductions of their plot.

Appendix X: Running LINE†

The LINE EXEC is an interactive program used to create physics diagrams. It can create a magnitude of different shapes (photons, gluons, scalars, fermions, etc.) as lines, loops, or bubbles. The exec creates a Topdraw file as output. The output file may be plotted from within the exec. By running LINE with the cursor argument the user can directly position the elements of the plot (e.g. titles). The exec is invoked as follows:

```
+-----+
| FORM: LINE Arguments ( Options | Arguments & Options may be omitted.
+-----+
```

Arguments- control internal operation of LINE.

Options- are used to interface external user files with LINE.

Arguments = (none)	- runs exec in line by line prompt mode.
Go	- runs exec without this notification.
Menu	- runs exec in full screen menu mode.
X	- runs exec in menu mode with option to read in graphic coordinates with cursor.

Options = (none)	- use default plotting procedure.
Plot fn1	- use user supplied routine, fn1 EXEC.
Fort fn2	- use user supplied routine, fn2 FORTRAN.

The graphical elements are specified by a short mnemonic and positioned by two endpoints. The exact geometries of the graphical elements can be controlled by the user. A manual on the use of this exec is available from BWL, Bin 95, x2706. The manual contains examples for first time users as well as the details of LINE's operation. The notes which follow contain some of the technical details of operation that the experienced user should be aware of.

NOTES:

1. The program is very new and so changes can be expected.
The support for the program may also be unreliable at times.

† This appendix begins with the online help file for LINE.

2. The MENU OPTION is valid for 24 and 43 VM. It is the recommended style of operation. Line by line mode is supported, but some features are not.
3. The CURSOR OPTION can only be used with the full screen menu. It is only supported for the Ann Arbor Ambassador and XL series terminals. It utilizes the cursor keyboard and terminal readout directly without going through Unified Graphics (much like GMPLLOT). It must be specified as an option in order to appear on the menu. When used the updated coordinate values appear on the menu. The cursor can be used without having to regenerate the plot by using PFO9 and having NO in the View the graph ? prompt (before hitting the PF key). The options can be viewed by answering YES to the cursor prompt, or the option number can be directly selected. In order to prevent the series 1 emulator from hanging it was necessary to reprogram the return key. In the extremely unlikely event of a system crash while waiting for the ASCII READ this key will need to be reprogrammed: the procedure is very simple - consult the Ann Arbor terminal manual (under DCS key programming). The break key is also reprogrammed - full restoration is done with:
 SETUP T ESC "Og ESC P'+~M ESC P'(y~[{E) ESC \
4. The PLOT OPTION allows the user to specify their own plotting procedure. The default option is run an internal procedure which creates a Topdraw SEQ4013 file which is then plotted by GMPLLOT. GMPLLOT is immediately exited, but the user can view the graph by using the SEND key. If the plot option is used the user must have an exec with that filename on a R/W disk. Furthermore, the exec must accept two arguments; the filename and filetype of the Topdraw file to be plotted. The cursor option may still be used provided that the exec is being run from a graphics terminal.
5. The FORT OPTION allows the user to utilize the structure of the exec to run their own fortran to generate the graphic output. The exec provides for this by having the user type USER when asking for the fortran program (typically LINE or LOOP). When run from the menu the user must conform to the standard protocol. This is as follows: five entries will be put onto the program stack before fortran execution, they are the choice of line (character*5) and four coordinates (real*8); x1, y1, x2, y2. These are the endpoints for the typical line, the user can do with them what they

wish here. Provided that they pull these values off the program stack, otherwise the exec will drop the stack and go into prompt mode. The output of the fortran program must be to unit 7. If any input is needed (besides the five standard entries), it can be done through unit 8, where the input file is has the same filename as the fortran, but has filetype CTRL. If the TEXT file is not on a R/W disk the exec will compile the fortran program with the FORT exec.

The previous three pages show the basic syntax for running *LINE* with its various options (in fact, it is the online help file for *LINE*). The external options allow the user to supply their own routines, either for plotting the output file or for running *LINE* with customized fortran routines. The latter allows the user to extend the capacity of *LINE* to draw just about anything (to the extent the user wishes to work). The fortran option is run by entering *USER* to the input prompt, *Line* or *Loop*. The fortran must pull five items off the stack if the option is to be used in the menu mode. The code should ask for the input as if it were doing I/O with the terminal (see subroutine *GET* in *LINE* or *LOOP FORTRAN* for an example). The five inputs are the standard protocol of *LINE* - one character*5 choose (for the kind of line) and four endpoint coordinates. Because the fortran programs are run with *CMSTYPE HT* you cannot prompt for additional input from the terminal. Instead, if you need more input to your program you can use unit 8. The name of your input file must be the same as the fortran program, and its filetype must be *CTRL*. Lastly, your fortran program must write to unit 7, and conform to *TOPDRAW*'s conventions.

The cursor option allows the user to exactly position the graphic elements, and to rotate and translate the endpoint coordinates trivially. This option is supported only for the full screen menu operation on either the Ann Arbor Ambassador or XL series terminals (the latter are supposedly destined to become the SLAC standard). When *LINE* is run in this mode an additional prompt appears on the master menu. This prompt asks whether the crosshairs are to be displayed and is referenced after the plot option is chosen (either from the prompt on the menu or via the PF09 key). After a line is drawn the default plotting routine exits and the program returns to the menu to determine whether the last line drawn is to be kept. After the menu is fully displayed the send key (upper right hand corner of the keyboard) can be used to toggle back and forth between the alphanumeric and alphagraphic displays (the graph is on the alphagraphic screen, the menu is on the alphanumeric screen). If the user toggles into the alphagraphic display prematurely, part of the menu might appear on the graphic display. This can be confusing but is easily fixed by clearing the screen locally with *CTRL L* (when done in the alphanumerical display). After pausing to get the keep option *LINE* then proceeds onward, and it is then that the crosshair prompt is examined.

The other time when the crosshair option is examined is when the PF09 key is entered. The normal response to this key is for *LINE* to plot the *Topdraw* file. However, when run with the crosshair option the plot prompt (*Do you want to view the graph ?*) is used to decide if the graph is to be replotted. This allows the user access to the crosshair readout

without having to constantly redraw the plot. After examining this prompt the exec next examines the crosshair option, and will put up the crosshair option list if the prompt is not blank or nc:

The cursor (or crosshair) option list allows the user to update the values in the coordinate input fields for the master menu (these are the X1, Y1, X2, and Y2 fields). Thus, the fields can be used strictly as output; as, for instance, when the user merely wants to use the readout in order to locate a point for positioning a Topdraw title. But, the points can subsequently be used as inputs for the next line drawn. The latter style allows for the simulation of interactive positioning of the graphic elements (if the line is incorrectly placed then it is not kept, and the cursor option is used to position the endpoints for the next try). The cursor selection options offer a great deal of versatility in the latter style of operation. The most straight forward application is to use the cursor to locate the starting point and end point. This is done by selecting option 1. The crosshairs are displayed and the position is read out after the user hits any character key (or return). The crosshairs are redisplayed to get the second point. After these coordinates are obtained the exec returns to the menu and the cursor position is given by the values in the coordinate fields. Other cursor options allow for only one point to be moved, or for translations of the endpoints (in which case you enter two points via the cursor which are used to construct a vector, you need not position the cursor on the element to be moved), and lastly, rotations can also be done. The latter are done about the centerpoint of the line under the option Translation and Rotation (5), or about an endpoint under the Pivot option (3). As an aside, the pivot option is used to move a single point, and need not rotate the figure; on the other hand, if the endpoint is to be rotated the cursor need not even be used. A short list of the cursor options follows on the next page.

The last feature of the cursor option is choice 0, Initialization. The coordinate system used by the graphic terminal is, roughly, a 4100 by 3200 set of coordinates with their origin at the lower right hand corner of the screen. *LINE*, on the other hand, utilizes the data coordinate system (the 10 by 10 axes) to locate positions. This necessitates keeping a set of conversion factors within the program. These conversion factors assume the plot is being made with the default axes. However, the user is not restricted to this setup. In fact, the cursor option can be used to locate positions on plots which *LINE* did not make. To do so requires a different set of conversion factors, and these are made with the Initialization option. Basically there are four conversion factors: a x scale, a y scale, and the two coordinates of the data coordinate system (wrt the terminal's). In order to fix these factors the user must enter in four values; which are the coordinates (wrt the

data system) of two points. This allows the exec to calculate and store the conversion factors internally, which in turn allows the user to exclusively reference the data system coordinates. The input points cannot have a degeneracy in x or y values (since there are two scale factors), and are best chosen along the diagonal with as much separation between them as possible. Note that this option can be used many times during a session. This enables the user to reference positions for multiple plots on a page.

Cursor selection options:

Choose from the following (cr= 1):

- 0. Initialization
- 1. Normal ordering
- 2. Reverse ordering
- 3. Pivot
- 4. Translation
- 5. Translation and Rotation
- 6. Return
- 7. Help

Help message:

Crosshairs are used to input coordinates to menu.
They are moved by cursor controls on the terminal.
Hit any character key in order to input the position.
Pick an integer according to the following list:

- 0. Initialization : To establish conversion factors.
 : Needed if default axes aren't used.
 : Be careful to pick points diagonally.
- 1. Normal ordering : Starting point, finishing point.
- 2. Reverse ordering : Finishing point, starting point.
- 3. Pivot : Only one coordinate is changed.
 : You can do it with the cursor
 : or by entering a rotation angle.
- 4. Translation : Element's endpoints are translated
 : according to your input vector.
- 5. Translation and Rotation : Similar to 3, only you are also
 : prompted for a rotation angle(in deg).
 : Rotation(ccw) about midpoint of element.
- 6. Return : Returns you to the menu.

Appendix XI: Additional Examples†

Meson Production in e^+e^- Collisions

† These examples are illustrations of *LINE*'s abilities, and are not annotated. One comment should be made however, they were done in menu mode and utilized the cursor positioning, sometimes extensively.

TYPE I

TYPE II

TYPE III

TYPE IV