Monte Carlo with Madgraph
Personal notes and how-to guide

Flip Tanedo

Institute for High Energy Phenomenology,
Newman Laboratory of Elementary Particle Physics,
Cornell University, Ithaca, NY 14853, USA

E-mail: pt267@cornell.edu

This version: August 3, 2011

Abstract

This is a set of LaTeX'ed notes on collider Monte Carlos using Madgraph based on a course lectured by Maxim Perelstein. It's not really meant for publication, use at your own discretion. Please send comments and corrections to `pt267@cornell.edu`.

Contents

1 Introduction

Reader, this is document. Document, this is reader. Consider yourselves introduced. All of problems come from Maxim's Collider Physics course[1]. For this example we'll use the MG5 web form.

2 Installing Madgraph 5

The first step is installing Madgraph 5. For basic things you can use the online webform at `http://madgraph.hep.uiuc.edu/` (or just search on Google), but we'll go ahead and install a local version. I have a Mac, so that's what we'll focus on. (This should be more or less the same as a Linux installation.) Click the 'Downloads' link on the MadGraph homepage and download and unpack the tarball. You'll need Python version 2.6. This is probably pre-installed on Mac since I didn't have to do anything special.

Go to the directory and run the interface by typing:

```
./bin/mg5
```

from there you can run the tutorial by typing `tutorial` as instructed. Bazinga, you're running Madgraph like a boss.

3 Monte Carlo Cross Sections

The master formula for $2 \to N$ scattering is

$$d\sigma = \frac{1}{2s} \prod_{i=1}^{N} d\Pi_i (2\pi)^4 \delta^{(4)}(p_A + p_B - \sum_i p_i) \cdot |\mathcal{M}|^2 \qquad d\Pi_i = \frac{d^3\mathbf{p}_i}{(2\pi)^3} \frac{1}{2E_i}. \tag{3.1}$$

Instead of analytic formulae, we can use Monte Carlo methods to 'play dice' just like Nature. We use a computer to generate random events, where **random** means that the events are weighted by $|\mathcal{M}|^2$ and **event** means a set of four-vectors in phase space.

Monte Carlo generators follow the 'onion' image of a detector.

- **Short distance**: $E \sim 100$ GeV, $\sim 10^{-16}$ cm. This corresponds to hard scattering parton-level events. We will use MadGraph, another option is CompHEP. This is the level which we tweak for new physics.
- **Showering & hadronization**: $E \sim 1$ GeV $\sim \Lambda_{\rm QCD}$, $\sim 10^{-14}$ cm. Typical options include Pythia and Herwig. They do different things well, but the former has a broader user base.
- **Detector simulation**: μm $- 10m$, macroscopic. Experimentalists have their own proprietary software for their detectors. Theorists are happy using PGS.

[1] `http://www.lepp.cornell.edu/~maxim/P661/`

4 The Madgraph 5 Tutorial

Run the Madgraph tutorial within the MG5 interface. When you first load the interface, it tells you some info about what settings are pre-loaded:

```
load MG5 configuration from input/mg5_configuration.txt
Using default text editor "vi". Set another one in ./input/mg5_configuration.
   txt
Loading default model: sm
models.import_ufo: Restrict model sm with file models/sm/restrict_default.dat .
models.import_ufo: Run "set stdout_level DEBUG" before import for more
   information.
INFO: Change particles name to pass to MG5 convention
Defined multiparticle p = g u c d s u~ c~ d~ s~
Defined multiparticle j = g u c d s u~ c~ d~ s~
Defined multiparticle l+ = e+ mu+
Defined multiparticle l- = e- mu-
Defined multiparticle vl = ve vm vt
Defined multiparticle vl~ = ve~ vm~ vt~
```

Nothing to big to see here, everything is more or less self-explanatory. Go ahead and type in `tutorial` and press enter. The interface will tell you that you've entered tutorial mode.

```
To learn more about the different options for a command, you can use
mg5>help A_CMD
To see a list of all commands, use
mg5>help

The goal of this tutorial is to learn how to generate a process and to
produce the output for MadEvent. In this part we will learn
a) How to generate a process
b) How to create output for MadEvent
c) How to run the MadEvent output

Let's start with the first point, how to generate a process:
mg5>generate p p > t t~
Note that a space is mandatory between the particle names.
```

We can go ahead and follow the tutorial copy the `generate` code to generate a process with two protons going to a top and an anti-top. MG5 will start at lowest order in QED couplings (`QED=0`) and try to generate parton-level processes based on the different partons that are associated with the proton, as MG5 told us when we loaded it. Madgraph then tells us:

```
You can find more information on supported syntax by using:
mg5>help generate
To list all defined processes, type
mg5>display processes
```

```
If you want to know more about particles and multiparticles present, write
mg5>display particles
mg5>display multiparticles

If you want to add a second process, use the add process command:
mg5>add process p p > W+ j, W+ > l+ vl @2
This adds a decay chain process, with the W+ decaying
leptonically.

At this stage you can export your processes to different formats. In
this tutorial, we will explain how to create output for MadEvent.
This is done simply by typing:
mg5>output MY_FIRST_MG5_RUN
```

Asking for more information about the `generate` command gives the following information:

```
syntax: generate INITIAL STATE > REQ S-CHANNEL > FINAL STATE $ EXCL S-CHANNEL /
    FORBIDDEN PARTICLES COUP1=ORDER1 COUP2=ORDER2 @N
-- generate diagrams for a given process
   Syntax example: l+ vl > w+ > l+ vl a $ z / a h QED=3 QCD=0 @1
   Alternative required s-channels can be separated by "|":
   b b~ > W+ W- | H+ H- > ta+ vt ta- vt~
   If no coupling orders are given, MG5 will try to determine
   orders to ensure maximum number of QCD vertices.
   Note that if there are more than one non-QCD coupling type,
   coupling orders need to be specified by hand.
Decay chain syntax:
   core process, decay1, (decay2, (decay2', ...)), ...  etc
   Example: p p > t~ t QED=0, (t~ > W- b~, W- > l- vl~), t > j j b @2
   Note that identical particles will all be decayed.
To generate a second process use the "add process" command
```

Go ahead and export the process to MadEvent following the instructions above. MG5 tells us that it's doing some work and places the process data into a directory `MY_FIRST_MG5_RUN`.

```
Type "launch" to generate events from this process, or see
/Users/doki/Physicsware/MadGraph5_v1_3_2/MY_FIRST_MG5_RUN/README
Run "open index.html" to see more information about this process.
tutorial:
  If you are following the tutorial, a directory MY_FIRST_MG5_RUN has
  been created which can be used in order to run MadEvent exactly as if
  it was coming from MG4.

  Additionally to the MG4 command (see MY_FIRST_MG5_RUN/README), you can also
  generate your events/compute the cross-section from this interface:
  Please Enter:
  mg5> launch MY_FIRST_MG5_RUN
  (you can interrupt the computation to continue the tutorial by pressing Ctrl-
     C)
```

It also informs us that it's generated jpeg diagrams and webpages in this directory. The webpage looks like the one shown in Fig. 1. Nothing too exciting in there yet.

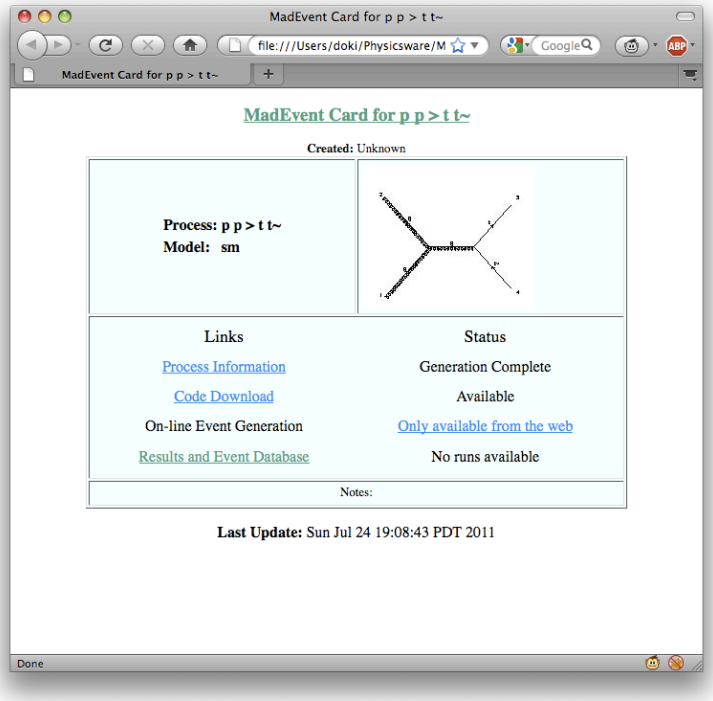


Figure 1: The webpage initially created for our `p p > t t` process.

Let's generate events. Follow the instructions and launch. We're given a warning:

```
WARNING: If you edit this file don't forget to modify
            consistently the different parameters, especially
            the width of all particles.
Do you want to edit file: param_card.dat? [y, n, path of the new param_card.dat
   ]
```

We'll go ahead an use the default parameter and run cards. As a rule of thumb one should be careful modifying parameters 'by hand' since the variables are not all independent. For non-trivial runs, one should use the calculator tools available on the MG website. We the webpage in Fig. 2 popping up.

We also get some gibberish, ending in some useful commentary:

```
The total cross-section is 0.57014E+03 +- 0.81587E+00 pb
more information in /Users/doki/Physicsware/MadGraph5_v1_3_2/MY_FIRST_MG5_RUN/
   index.html
tutorial: This step ends the tutorial of the basic commands of MG5. You can
  always use the help to see the options available for different
  commands. For example, if you want to know all valid output formats,
```

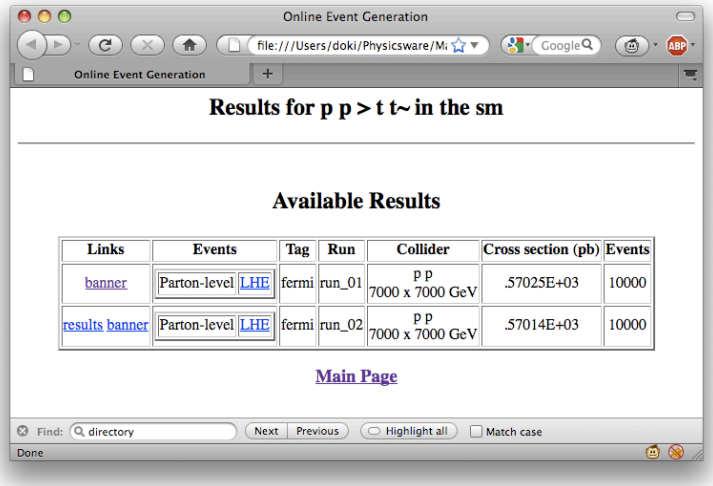


Figure 2: The webpage initially created for our `p p > t t` process.

```
you can enter
mg5>help output

In order to close this tutorial please enter
mg5>tutorial stop
If you want to exit MG5 please enter
mg5>exit

But you can also continue the tutorial to learn some other useful
commands:
d) How to load a model
e) How to define a multi-particle label
f) How to store a history of the commands in a session
g) How to call shell commands from MG5
h) How to draw the diagrams for your processes without generating
   MadEvent output

To import a model, write:
mg5>import model mssm
```

From the webpage we can download the LHE files, view the results, and download 'banners' with all of the input information. The tutorial suggests further things you can do. For now we'll stop and exit.

5 Muon angular distribution at a lepton collider

A good toy example is $e^+e^- \to \mu^+\mu^-$, the 'hello world' of particle physics.

5.1 Some results

Labeling a fermion's helicity by L and R, we have the helicity amplitudes

$$\mathcal{M}(e_L^+ e_R^+ \to \mu_L^- \mu_R^+) = \mathcal{M}(e_L^+ e_R^+ \to \mu_L^+ \mu_R^-) = -e^2(1+\cos\theta) \tag{5.1}$$
$$\mathcal{M}(e_R^- e_L^+ \to \mu_L^- \mu_R^+) = \mathcal{M}(e_R^- e_L^+ \to \mu_L^+ \mu_R^-) = -e^2(1-\cos\theta). \tag{5.2}$$

Usually, however, the colliding electrons are unpolarized and the muon polarization is not measured. This means that we average over the electron polarizations and sum over the muon polarizations, giving

$$\frac{1}{4}\sum_{\text{hel.}} |\mathcal{M}|^2 = e^4(1-\cos^2\theta). \tag{5.3}$$

In the limit $E_{\text{CM}} \gg m_\mu$ we have the differential cross section

$$\frac{d\sigma}{d\cos\theta} = \frac{e^4}{32\pi s}(1+\cos^2\theta). \tag{5.4}$$

A useful quantity is

$$R \equiv \sigma = \frac{4}{3}\frac{\pi\alpha^2}{s} = \frac{868 \text{ nb}}{(E_{\text{cm}} \text{ in GeV})^2}. \tag{5.5}$$

5.2 Dimuons at the ILC, using the MadGraph web interface

Now we can use the MadGraph web interface at `http://madgraph.hep.uiuc.edu/`. MadGraph and MadEvent are a pair of programs which draws tree-level Feynman diagrams, generates an analytic expression for each diagram, and then sums them. The web interface is very straightforward and is a good way to get used to MG before tinkering directly with cards.

Let's start with Part I of the form, see Fig. 3.

1. Pick a model. Later on it'll be easy to modify the files to input your own model. For now use the Standard Model (SM).

2. Specify a process. Click on the 'Model descriptions' or 'Examples/format' links for syntax. For our process, we want $e^+e^- \to \mu^+\mu^-$. Further, we want this not in the Standard Model, but in QED. We thus have to remove the Z. This is easy to do. Input the following process: `e+ e- > mu+ mu- /Z`

3. Previous versions also asked for 'max order' in QCD and QED (how many vertices to include). It appears that MG5 now automatically assumes lowest order. You can impose this in the process, for example, `u u~ > d d~ QED=2`.

4. Proton and jet definitions. You can tweak this, for example, if you only care about valence quarks.

5. Sums over leptons. For example, if you want to sum over all neutrino flavors.

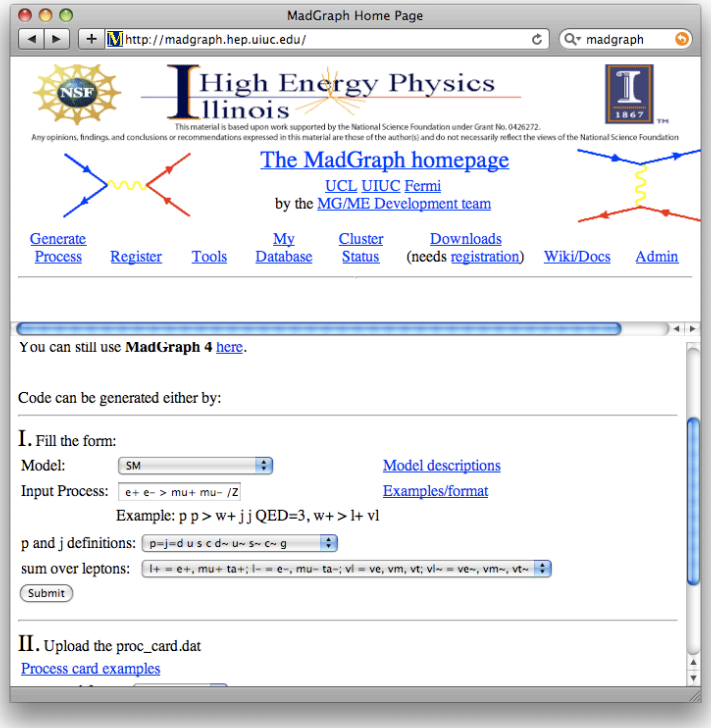


Figure 3: The MG5 webform.

The form basically produced a process card, `proc_card_mg5.dat`. You could have alternately uploaded your own process card. We end up with a webpage that is basically the same as Fig. 1.

Let's explore the **Process Information** link. This is your first sanity check. You can click to see the diagrams that were generated. You can also download the process card, go ahead and take a peek. Any line starting with a `#` is read as a comment. The card starts with a nice header in comments. The meat of the process file is as follows:

```
import model sm
define p = g u c d s u~ c~ d~ s~; define j = p
define l+ = e+ mu+ ta+; define l- = e- mu- ta-; define vl = ve vm vt; define vl
    ~ = ve~ vm~ vt~
generate  e+ e- > mu+ mu- /Z
output madevent -f
```

Pretty straightforward. This is actually much cleaner than the previous MG4 process card, which is why the process cards have the `_mg5` in the filename. The MG4 format takes the form:

```
#*********************************************************************
# Process(es) requested : mg2 input                                  *
#*********************************************************************
# Begin PROCESS # This is TAG. Do not modify this line
e+e->mu+mu-/z   #  Process
```

```
QCD=0   # max qcd order
QED=2   # max qed order
end_coup   # the coupling list is over
done       # the process list is over
# End PROCESS  # This is TAG. Do not modify this line
#*********************************************************************
# Model information                                                  *
#*********************************************************************
# Begin MODEL  # This is TAG. Do not modify this line
sm
# End   MODEL  # This is TAG. Do not modify this line
#*********************************************************************
# Start multiparticle definitions                                    *
#*********************************************************************
# Begin MULTIPARTICLES # This is TAG. Do not modify this line
p uu~cc~dd~ss~g
j uu~cc~dd~ss~g
l+ e+mu+
l- e-mu-
vl vevm
vl~ ve~vm~
# End  MULTIPARTICLES # This is TAG. Do not modify this line
```

The MG4 version also makes use of two files which are note in the MG5 version: `particles.dat` and `interactions.dat`. These files are self-explanatory. In MG4 they live in the `Models` directory in appropriate folder for the particular model (`sm`). The good news is that you can still use this format for MG5, i.e. you can take an MG4 model directory and copy it into the `Models` folder of MG5 and MG5 will be happy to read it. Even better, MG5 implements python versions of these files which are much more intuitive to modify.

Okay. Enough dilly-dallying. Let's generate events. Go to the **On-line Event Generation** link. The generator now asks for several parameter cards. As with the process card, you can upload these manually or use the web form. We'll use the latter.

For **Model parameters** we can go ahead and use the present `param_card.dat`. This includes information about couplings, such as the renormalized value of α_S via PDF data. Note that there is a link to a web form to generate this card because of correlations between the values. Unless you know what you're doing, do *not* modify the card directly.

The **Collider and cuts** part of the form asks us about the machine that we're simulating. Give the run a name, the default appears to be 'fermi.' The number of events is a choice between accuracy and efficiency[2]. Stick to 10,000 events. We'll be using the ILC at 500 GeV, which is a convenient pre-set option. We won't bother with the other options, leave them at their default values. Glancing over the rest of this part of the form you can see how cuts are implemented on the transverse momentum, rapidity, and energy of each type of particle in the lab frame. Most of the basic selection cuts that you read about in hep-ex papers can be implemented right in this form.

[2]Note: MG5 warns us that when running locally, do batches of no more than 10k events at a time. The random number generator is smart enough to use a new seed with each batch.

Remark: You might wonder why we want to impose cuts at the event generation level; why not just generate a bunch of events the way nature does? Recall that the matrix element gives the weighting by which we generate events. The danger is that tree-level matrix elements can be plagued by soft (&) collinear divergences. *Physically* we know that these divergences are taken into account in the evolution of the parton distribution functions, but our tree-level matrix elements can be sharply peaked—even divergent—for soft collinear partons. In order to remove these cases, one can impose cuts at the generator-level to only populate interesting (and physical) events.

We'll ignore the plotting, Pythia, and detector simulation parts of the form for now. Go ahead and send the form (the 'send' button is at the top of the page). This sends you to a page where you can download your cards. Click the 'send' button at the bottom of this page and wait for your events. Gradually the page will update with results as they come in.

Warning: for some reason I have a hard time telling the webform to set the machine to the ILC. This manifests itself as an 'Error generating events' status because there are no $e^+e^- \to \mu^+\mu^-$ events in a pp collider. I went ahead and modified the card directly:

```
#*********************************************************************
# Collider type and energy                                           *
#*********************************************************************
        0     = lpp1  ! beam 1 type (0=NO PDF)
        0     = lpp2  ! beam 2 type (0=NO PDF)
      250     = ebeam1  ! beam 1 energy in GeV
      250     = ebeam2  ! beam 2 energy in GeV
```

The first two parameters tell us about whether the beams are (anti-)protons or electrons.
Update: When using the web form to generate events, remember to select "Fill in the form below" for the relevant sections. In this case, I should have selected this option under the "Choose an option for the run_card.dat" query.

We end up with something like Fig. 4. Some things to note:

- You can click on 'results' for some details on the cross section including the error and luminosity.
- Clicking on 'banner' will give you a text file with all of the cards that we've used. This contains all of the input data.
- The parton-level events (each of the 10,000 events) are available as a Les Houches accord file (LHE) or a rootfile. You can also click on 'plots' for some... well, you know. An example

A theory remark. From (5.5) we expect 347 fb. The particular result I got when running this was 372 fb. This on the order of a 5% difference. Is this correct? Well, the key point is that the expression in (5.5) was based on parameters measured at $\mathcal{O}(1\text{ GeV})$. On the other hand, looking at the 'banner' for our Monte Carlo shows:

Results for e+ e- > mu+ mu- /Z in the sm

Available Results

Links	Events		Tag	Run	Collider	Cross section (pb)	Events	
results banner	Parton-level	LHE plots rootfile	fermi	run_01	e e 500 x 500 GeV	.90942E-01	10000	Remove run
	Hadron-level (Pythia)	Run Pythia						

Main Page

Figure 4: Results from our $e^+e^- \to \mu^+\mu^-$ Monte Carlo. Note that for this run I typed in the wrong beam energy. Whoops. The correct cross section is more like 372 fb.

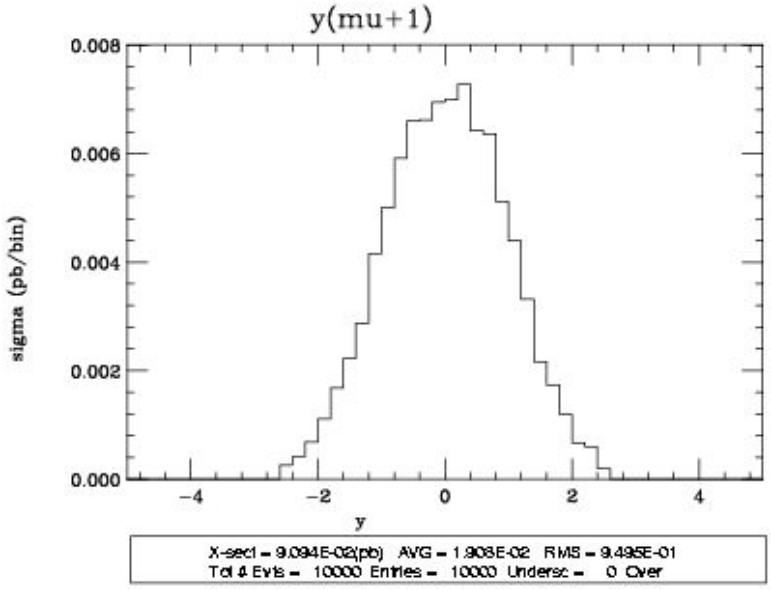


Figure 5: Rapidity distribution of the μ^+ from the 'plots' page of our generated events. The page conveniently includes a legend to interpret the labels and a way to download the files as postscripts.

```
###################################
## INFORMATION FOR SMINPUTS
###################################
Block sminputs
    1 1.325070e+02 # aEWM1
    2 1.166390e-05 # Gf
    3 1.180000e-01 # aS
```

In other words, it is using $\alpha^{-1} = 132.5$ rather than 137. Making this correction, we obtain

$$\left(\frac{132.5}{137}\right)^{-2} \times 347 = 371. \tag{5.6}$$

5.3 More about the Les Houches Accord

Let's take a moment to zoom out and refresh the 'big picture' of the MC program. This section borrows heavily (often verbatim) from Michael Peskin's 2007 West Coast LHC meeting talk `http://hep.ps.uci.edu/~wclhc07/PeskinIrvine.pdf`.

Testing a model require detailed evaluation of its signatures and an estimate of detector acceptance and response. This is especially true at a hadron collider, where signatures with quark jets resemble QCD background and signatures with leptons depend strongly on the detector geometry. It is even more true at the LHC, for which we are proposing complex models with many new particles. For an analysis at this level, experimenters need explicit events that can be passed through a simulated detector geometry. So, how do you make them?

The complete simulation of an LHC event in a theory of physics beyond the Standard Model requires many ingredients. In principle, all of these should be included for appropriate realism. It is necessary to decide which of these ingredients will be included in the simulation and, for each that is included, who will take responsibility. For example:

- **Model level:** detailed spectrum, production processes from initial state partons, decay chains, partial widths, production cross sections and dynamics, decay distributions and dynamics, polarization dependence and spin correlations.
- **Perturbative QCD level:** QCD showers for final state partons, QCD showers for intermediate colored states, initial state radiation.
- **Non-perturbative level:** 'underlying event' (ith possible extra jet production), ahdronization of all radiated partons, decays of hadrons and τs.

All of these effects are already included in Pythia and Herwig. The problem is that Pythia and Herwig do not include your particular pet model. Our goals is to include the model-level information about your arbitrary-ass pet model into these programs and let them do the heavy lifting. The general procedure we'd like is

1. Generate four-vectors for pair-production of your pet particle using a standard parton distribution (CTEQ or MRST)

2. Decay each particle into a random two-body final state

3. Put the resulting vectors into Pythia to generate other features of the event (assuming Pythia recognizes the final particles).

This program is achieved through the **Les Houches Accord**. The basic Les Houches Accord allows insertion of events from a parton level event generator into a parton shower generator such as Pythia or Herwig. Originally, the Accord was a common block to transfer information between FORTRAN subroutines. But now, a stand-alone generator like MagGraph/MadEvent can write a text file that can then be read by one of these programs.

The file type for the Les Houches Accord is 'lhe' (Les Houches Event) which is really just an XML file. The documentation can be found in hep-ph/0609017. The idea is that we generate the final state parton-level four-vectors using MG5 and output to an LHE file, which we can then simply feed into Pythia to carry out the rest of the process. MadGraph can automate the entire chain and even feed the Pythia output to PGS for detector simulation.

5.4 From LHE to Mathematica

And then there's this column called 'six'... I have no idea what that is.

– Maxim Perelstein

Great, let's get back to our analysis of $e^+e^- \to \mu^+\mu^-$. What we really want is the power to play with the data ourselves and create our own plots beyond the 'out of the box' plots that MadEvent gives us. There are a few options here. You can use the Root file that MadEvent outputs, but that requires knowing a thing or two about Root. Alternately, you can use the MadGraph plotting software. We'll do something different and use *Mathematica* to import the data from the Les Houches Accord file (LHE). In order to do this, we'll use a *Mathematica* notebook adapted from Chameleon (developed by the Harvard LHC Olympics BlackBox team). The text of the entire file is in Appendix B.

You can find out more about the Les Houches Event format in hep-ph/0609017. However, it's informative enough to go ahead and take a peek into one of the LHE files. The general format is:

```
<LesHouchesEvents version="1.0">
<!--
  # optional information in completely free format,
  # except for the reserved endtag (see next line)
-->
<header>
  <!-- individually designed XML tags, in fancy XML style -->
</header>
<init>
  compulsory initialization information
  # optional initialization information
</init>
<event>
  compulsory event information
  # optional event information
```

```
</event>
(further <event> ... </event> blocks, one for each event)
</LesHouchesEvents>
```

You'll find that the first chunk of text is a copy of all of our cards. All of this is under XML tags such as `<MGRunCard>` or `<MGGenerationInfo>`. We can skip all that. Next there's a short section under a `<init>` tag which describes the initial state particles.

Particle code. Note that particle identities are encoded by integers following the PDG-approved conventions. For example, 1 refers to a down quark while -1 refers to an anti-down quark. You can find the entire list in the 'Monde Carlo Particle Numbering Scheme' section of the PDG. When you make your own models, you're responsible for defining your own integers for new particles; as a rule of thumb, pick large numbers that are unlikely to be used by the Standard Model.

Ok, now we get to the good stuff. Each event looks roughly like Fig. 6 (note that this event is for Drell-Yan production). You can see that there is a line for each particle involved in the

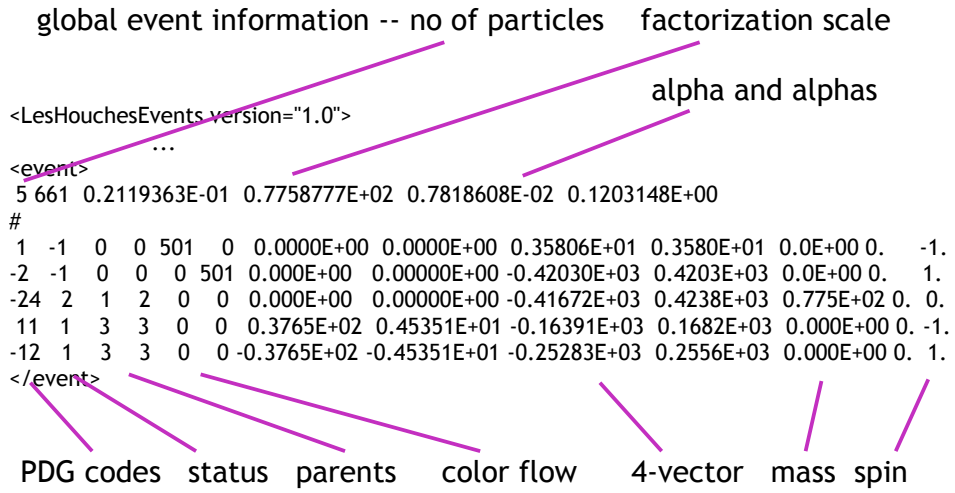


Figure 6: An LHE event, image from `http://hep.ps.uci.edu/~wclhc07/PeskinIrvine.pdf`.

process. Column-by-column, we have

- The particle identity given by its PDG code
- Whether the particle is an initial ($+1$), final (-1), or intermediate (2) state
- The particle's parents, listed as a row number
- Information about the particle's color structure
- The particle's momentum four-vector and its invariant mass

- The particle's spin.

We would like to import all of this information into a big table in *Mathematica* which we can then use to generate histograms.

Here's how it works. Go ahead and open `MadEvent analysis.nb`. This file contains definitions for functions that we'll use to import the LHE file data into *Mathematica*. Go ahead and evaluate this notebook; you can close it afterward, we just needed those definitions loaded. Next, open a new notebook—this will be our workspace.

The first thing you'll need to do is to tell this notebook where to find the LHE file. Start with the `SetDirectory` command with **menu → insert → file path**. If the folder is called **/Users/yourname/MCfolder**, you'll end up with something like:

```
SetDirectory["/Users/yourname/MCfolder"];
```

Next we'd like to import the LHE data. First we'll import the entire file. Suppose the file is called **lhefilename.lhe**, located in the directory above.

```
lhefile = Import["lhefilename.lhe", "Table"];
```

Good. That should take several seconds; it's a big file. Now we want to strip off all of the meta-data. Fortunately, the function `ReadME[]` does that. We can store the events in a *Mathematica* list which we'll call **eventListp**. Why the 'p'? I don't know, it's probably for 'Perelstein' or something. Load the list from **lhefilename.lhe** as follows:

```
eventListp = ReadME[testp];
```

Great. Now everything is loaded into **eventListp**, we just need to learn how to use it. First, let's define a variable **Nev** for the total number of events in **eventListp**.

```
Nev = Length[eventListp] - 1
```

This should output 10,000. (Add a semi-colon if you don't want *Mathematica* to talk back.) Note the crucial -1; for some reason it's always off by one. (There's an extra element at the end which is empty.) Let's test this out. Let's define a truncated list of two events, **tsp** and use the `EventPrint` function to see what these events look like.

```
tsp = {eventListp[[1]], eventListp[[2]]};
EventPrint /@ %;
```

This will output the same info as Fig. 6 but in *Mathematica*. Note that this tells you about *all* of the in and out (and decayed) particles. We already know what the in-states are and we're only concerned about the out-states. To peel off the in-state information, we have a handy function `SetOut` which strips all particles from an event that aren't designated as 'out' states. Go ahead and see for yourself:

```
eventListOutLp = SetOut[eventListp];
{eventListOutLp[[1]], eventListOutLp[[2]]};
EventPrint /@ %;
```

Excellent. Now we're all set. Let's get down to business. For this particular process, we are interested in the angular distribution, $d\sigma/d\cos\theta$. Let's now construct a list of $\cos\theta$ values for each event. We will use the function `theta` to calculate the azimuthal angle between a vectors and the beam axis. Note that we only need to look at the angle of one muon.

```
Cost = Range[Nev];
For[i = 1, i <= Nev, i++,
  Cost[[i]] = Cos[theta[eventListOutLp[[i]]][[1]]]];
```

Now for the important part. Let's make histograms. Unfortunately, this is rather dependent on your particular version of *Mathematica*. Here's how it works for *Mathematica* 6.

```
data = Histogram[Cost, HistogramCategories -> Table[i, {i, -1.0, 1.0, 0.1}]]
```

That should giveyou a nice histogram with the distribution of $\cos\theta$ values from -1 to 1 with bins of size 0.1. As of *Mathematica* 7 and onward, the `HistogramCategories` option was superceded by an optional argument in `Histogram` called `bspec`. For *Mathematica* 8 (and probably 7) you would use something like this:

```
data = Histogram[Cost, {-1.0, 1.0, 0.1}]
```

Ah, that's much simpler, isn't it? For the remainder of this document I'll try to do everything following the *Mathematica* 8 specifications. With any luck you should get a nice histogram like

If you want to extract the specific counts for each bin, that's also pretty easy:

```
data = BinCounts[Cost, {-1.0, 1.0, 0.1}].
```

Finally, you might want to compare with theory. A useful way is to make a table of 'bin counts' from an analytical formula such as (5.4) that the number of counts per bin is proportional to $(1+\cos^2\theta)$. We can figure out the overall prefactor just by normalizing with respect to the total

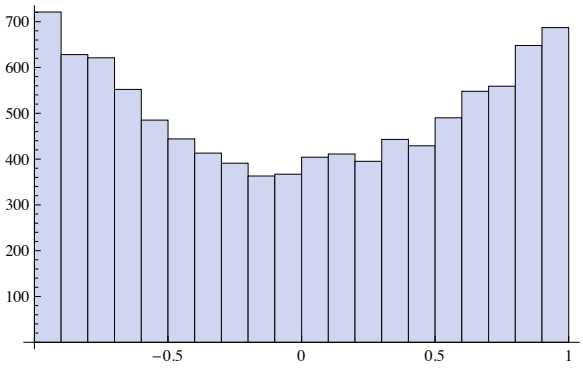


Figure 7: A nice histogram. You *did* get a nice histogram, right?

number of events. For example, we can pull out a factor of 500 (since there are 20 bins of a total of 10,000 events) and then normalize $(1 + \cos^2\theta)d\cos\theta$, which gives us:

```
theory = Table[(1 + (x + 0.05)^2)*(3/4)*500, {x, -1.0, 0.9, 0.1}]
```

Note that we shifted $x = \cos\theta$ by 0.05 to sample in the middle of each bin. A good sanity-check that the overall shapes match is to calculate and plot the residues,

```
residues = (data - theory)/Sqrt[(data + theory)/2]
```

You should end up with a nice random distribution about zero. Congrats, we've Monte Carlo'd the angular distribution of $e^+e^- \to \mu^+\mu^-$.

5.5 Homework 1.2: Muon Spin Determination

Note: I'm skipping non-Monte Carlo assignments.

> **Muon Spin Determination: Monte Carlo**. Using MadGraph, generate 10000 events for the process $e^+e^- \to \widetilde{\mu}^+\widetilde{\mu}^-$ at $\sqrt{s} = 500$ GeV, with no Z and no $\widetilde{\mu}$ mass. Plot the angular distribution of $\widetilde{\mu}$. Compare with the analytic results obtained in Problem 1, and with spin-1/2 results obtained in the Lecture 2 tutorial.
>
> **HINTS:** A scalar muon $\widetilde{\mu}$ is part of the Minimal Supersymmetric Standard Model, MSSM, where it's called a "smuon". There are actually two of them, $\widetilde{\mu}_L$ and $\widetilde{\mu}_R$; either one would be ne for the purposes of this problem, as their couplings to the photon are the same. The MSSM is part of standard Madgraph, so you do not need to create a "new MG model" to do this problem. You will need to set the smuon mass to zero though. Edit "paramcard.dat" file directly to achieve this. Ignore the warnings telling you to never do this: it's OK for this little toy problem.

For simplicity we'll go ahead and use the web interface. In the past, 'grown ups' would just modify the cards directly and run MG4 locally—however, the new features in MG5 make this

somewhat outdated. While the new method is elegant, it's a bit involved the first time through so we'll save that for a later example. As explained in the hint, this will be nearly identical to the previous $e^+e^- \to \mu^+\mu^-$ example except we have to use the MSSM and modify the model a bit.

The first step is to fill out the web form to generate processes. Set the model to MSSM and input `e+ e- > mul- mul+`. Go ahead and submit. Poke around with the generated files and cards if you so please, then click on 'generate events.'

Under 'Model parameters' select 'Go to the param_card web form.' Set up the 'Collider and cuts' section exactly as in the previous example. Don't forget to select 'Fill in the form below.' Click 'submit.'

At the time that I wrote this the MG5 web interface was being a dick, so I ended up just modifying the cards directly. The PDG code for the left-handed muon is `1000013` so we go to that line in `param_card.dat` and change the mass to zero:

```
1000013 0 # mul- : Msl1.
```

Anyway, this was supposed to work, but the MG5 web interface kept being a dick so I went ahead and did this locally. I went to the MG5 directory and into the `models` folder and made a duplicate of the `mssm_v4` folder. I named the copy `mssm_HW1`. This contains the MSSM in MG4 format (i.e. as cards rather than as UFO Python files). Go ahead and modify the param_card directly. Now go to the MG5 directory in the terminal and run `./bin/MG5` to get to the MG5 interface that we used in the tutorial. In MG4 you'd have to modify a bunch of files manually; screw *those* days, eh?

At the interface, we need to tell MG5 to load our newly modified MSSM model:

```
import model_v4 mssm_HW1
```

Note that we had to use a funny import command, `import model_v4`, in order to use our MG4 model definition. If we'd just used `import model`, MG5 would complain that we don't have enough UFOs and would send agents Moulder and Scully to your house. We can generate diagrams (a `proc_card.dat` files) by typing in

```
generate e+ e- > mul+ mul-
```

We now store these to a folder, let's say `Run_HW1`:

```
output Run_HW1
```

MG5 tells us that the directory has been set up and directs us to an html page identical to what you would get using the web form, except that you can't click for 'on-line event generation.' The next step is to enter `launch`. You'll now be prompted to modify the param and run cards. Here's the tricky thing for Linux newbies: the default editor is vi. If you don't like this, you might want

to consider changing the `/input/mg5_configuration.txt` file so that the `text_editor` value has a friendlier shell-based editor: perhaps emacs or nano.

Important vi commands: vi starts out in command mode, which means it is ready to accept commands but *not* text to be inserted into the document (as one would expect from a GUI text editor). To get to something you're more familiar with, type in `a` to go to insert mode. Now you can move the cursor around and type as you would expect. Press the escape key to return to command mode. To save a file from command mode, type in `:w` (then press enter) for 'write.' To quit, type `:q`. You can combine these last two commands to `:wq`. One benefit of an advanced editor like vi is that it is easy to use command mode to search for a string. Use forward slash to search for an expression after the cursor, or question mark to search before. For example: `/Pikachu` will search for the next occurrence of the term 'Pikahu', while `?Raichu` will search for the previous occurence of the term 'Raichu.'

Go ahead and leave the param card as it is (or open it, I don't care) and modify the run card appropriately, MG5 will open up the results html page. Download the LHE file and fire up *Mathematica* as described in the previous section. I went ahead and renamed the LHE file to something handy, like `HW1.lhe`. Now you can go ahead and run all of the same commands as in the $e^+e^- \to \mu^+\mu^-$ example. You should end up with something like Fig. 8. Confirm that this is the shape that you expected.

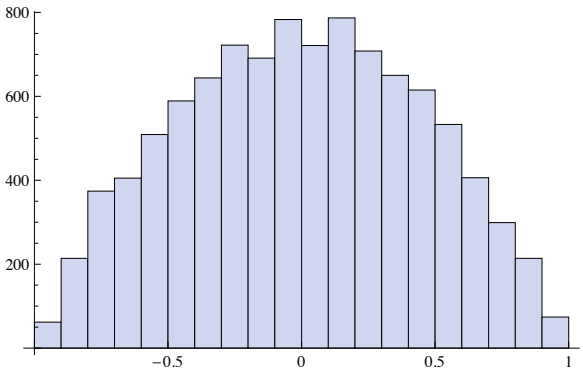


Figure 8: HW1 histogram. This *is* what you expected, right?

5.6 Homework 1.3: Reproducing real $e^+e^- \to \mu^+\mu^-$ data

Using Madgraph, reproduce the experimental results shown in Fig. 9, i.e. cross section and forward-backward asymmetry of the muons as a function of $\sqrt{s}$. Note: you will need to run MG/ME at a series of $\sqrt{s}$ values, so you will need to edit the "runcard.dat" le directly, instead of using the web form. Of course, both Z and γ exchange diagrams need to be included!

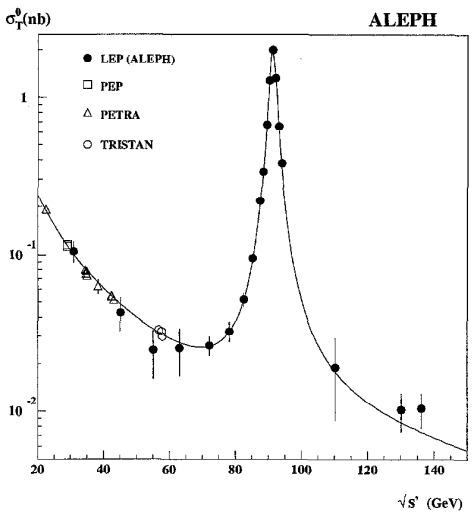


Fig. 2. Measured cross sections of muon-pair production compared with the fit results. The ALEPH measurements below 60 GeV correspond to the exclusive hard ISR selection that are not used in the fit. For comparison the measurements at lower energies from PEP, PETRA and TRISTAN are included.

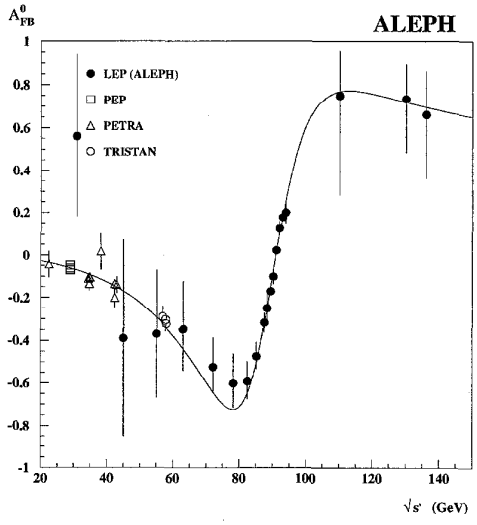


Fig. 3. Measured forward-backward asymmetries of muon-pair production compared with the fit results. The ALEPH measurements below 60 GeV correspond to the exclusive hard ISR selection that are not used in the fit. For comparison the measurements at lower energies from PEP, PETRA and TRISTAN are included.

Figure 9: Plots from a paper by the ALEPH collaboration, Phys. Lett. B 399, 329 (1997)

Recall that the **forward-backward asymmetry** is a measure of how many events are in the forward direction versus the backward direction (e.g. net electron flow versus net positron flow),

$$A_{\rm FB} = \frac{\sigma(\cos\theta > 0) - \sigma(\cos\theta < 0)}{\theta(\cos\theta > 0) + \sigma(\cos\theta < 0)}. \tag{5.7}$$

Recall that for $e^+e^- \to \mu^+\mu^-$ this quantity is nonzero in the Standard Moddel due to the chiral couplings of the Z boson. Indeed, for Z-mediated events, the angular distribution for each helicity amplitude are all proportional to $(1 \pm \cos\theta)$, as in the case of photon-mediated events, except that the prefactors are now different. For Z-mediated events this leads to

$$\frac{d\sigma}{d\cos\theta} \propto (g_L^2 - g_R^2)^2 \cos\theta. \tag{5.8}$$

Compare this to the $\cos^2\theta$ and $\sin^2\theta$ distributions that we found for $e^+e^- \to \gamma \to \mu^+\mu^-$ and $e^+e^- \to \gamma \to \widetilde{\mu}_L^+\widetilde{\mu}_L^-$ respectively. The relative contributions of the parity-conserving photon diagrams and the parity-violating Z diagrams depend on the scale s.

Before jumping in, let's decide to generate 1000 events each for $\sqrt{s}$ values of: 60, 70, 80, 85, 88, 90, 91, 92, 95, 100, 110, 120. Okay. Let's fire up MG5. Since we're working with the Standard Model, we can just go ahead and input our process. Go ahead and generate the $e^+e^- \to \mu^+\mu^-$ diagrams. Have MG5 spit everything into a new folder, say `Run_HW1_3`.

```
output Run_HW1_3
```

Now enter `launch`. Don't modify the param card, but we'll need to modify the run card for each $\sqrt{s}$ value, so go ahead and make those. Be sure to change the tag name and beam energies for each card. (And don't forget to change the total number of events and the beam types from their default values.) Here's the relevant section of the first run card:

```
#*********************************************************************
# Tag name for the run (one word)                                    *
#*********************************************************************
  'run60'     = run_tag ! name of the run
#*********************************************************************
# Run to generate the grid pack                                      *
#*********************************************************************
  .false.     = gridpack  !True = setting up the grid pack
#*********************************************************************
# Number of events and rnd seed                                      *
# Warning: Do not generate more than 100K event in a single run      *
#*********************************************************************
   1000       = nevents ! Number of unweighted events requested
      0       = iseed   ! rnd seed (0=assigned automatically=default))
#*********************************************************************
# Collider type and energy                                           *
#*********************************************************************
        0     = lpp1  ! beam 1 type (0=NO PDF)
        0     = lpp2  ! beam 2 type (0=NO PDF)
      30     = ebeam1  ! beam 1 energy in GeV
      30     = ebeam2  ! beam 2 energy in GeV
```

This will pop up a webpage. Ignore it for now. We have other data to take. Back in the MG5 window, launch once again. Repeat for each energy. Somewhat annoyingly MG5 opens a new tab for each launch. By the end of it all, the webpage will tell you everything you need to know to recreate the total cross section plot. You'll end up with *beaucoup* LHE files. For simplicity, I renamed them `run1.lhe`, `run2.lhe`, etc.

Now back to *Mathematica*. Run the notebook with all the definitions and start a new notebook. Load all of the events into a single list. This involves loading each file individually and then concatenating the contents into the event list. Here's the code I ended up writing:

```
SetDirectory[
  "/Users/doki/Documents/Work/ColliderMadgraph/Homeworks/HW1/"];
sqrts = {60, 70, 80, 85, 88, 90, 91, 92, 95, 100, 110, 120};
j = 1;
eventListOutbyE = Range[Length[sqrts]];
Nev = Range[Length[sqrts]];
Afb = Range[Length[sqrts]];
While[j <= 12,
 testp = Import["run" <> ToString[j] <> ".lhe", "Table"];
 eventListOutbyE[[j]] = SetOut[ReadME[testp]];
 Nev[[j]] = Length[eventListOutbyE[[j]]] - 1;
 Afb[[j]] =
  Sum[Sign[Cos[thetaOf[eventListOutbyE[[j]][[k]][[1]]]]], {k,
```

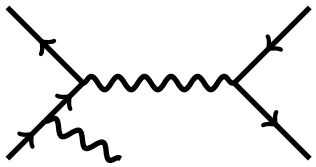
```
    Nev[[j]]}]/Nev[[j]];
 j++]

ListPlot[Transpose[{sqrts, Afb}]]
```

It should be straightforward to glean the meaning of the relevant Chameleon commands (e.g. **thetaOf**) or to otherwise look them up in the driver notebook.

6 Initial and Final State Radiation

Looking back at Fig. 9 you might wonder how ALEPH took data below the Z peak when LEP never ran at this low energy range. The answer is **initial state radiation** (ISR), in this case the emission of a photon off of one of the electrons. ISR becomes more important at a high-energy lepton collider, but it is *always* important in QCD collisions. Now that we've exhausted $2 \to 2$ processes, let's turn to ISR (and FSR) and $2 \to 3$ processes of the form



Remark: the set of ISR diagrams form a gauge invariant subset of the total set of diagrams, and hence it's 'fair' to treat them independently for now.

6.1 ISR Theory

Naively, we expect that the cross section for ISR diagrams is suppressed by a factor of $e^2/4\pi^2 \approx 0.3\%$, coming from the additional photon vertex and the difference in the 3-body versus 2-body phase space. However, the additional electron propagator introduces a factor of

$$\frac{1}{(p_e - p_\gamma)^2 - m_e^2} = \frac{1}{2p_e \cdot p_\gamma}. \tag{6.1}$$

In the $m_e \to 0$ limit, we may write (writing A for the electron)

$$p_A = (E, 00, E) \tag{6.2}$$

$$p_\gamma = \left(zE, \mathbf{p}_\perp, \sqrt{z^2E^2 - \mathbf{p}_\perp^2} \right). \tag{6.3}$$

This, however, means that the denominator of the electron propagator vanishes if

(a) $z \to 0$, i.e. a **soft singularity**, or if

(b) $\mathbf{p}_\perp/z \to 0$, i.e. a **collinear singularity**.

These singularities are not actual physical divergences, since restoring a small finite m_e removes the collinear singularity and the soft singularity is cancelled by the divergence in the QED vertex function. Regardless, these 'singularities' represent a *physical* enhancement in the probability of soft, collinear photon emission.

Note that these soft, collinear photons are not observable; soft things don't register well in the calorimeter and collinear things disappear down the beam pipe. We thus have to set a minimum energy and a minimal transverse momentum Q for a 'detectable' photon. A $2 \to 3$ process where $p_\perp < Q$ is recorded as a $2 \to 2$ process, so that the soft and collinear singularities are absorbed into the correction to the $2 \to 2$ cross section.

Let's work out the size of this correction. In the small $p_\perp$ limit, we may write $p_\gamma \approx (1-z)p_e$ up to $\mathcal{O}(p_\perp)$. The electron propagator thus has a denominator proportional to $p_A \cdot p_\gamma \approx (1-z)p_A$. Indeed, in the $m_e \approx 0$ limit, this mean that the $(p_A \cdot p_\gamma)^2 = 0$ and for low p_T^γ electron remains on-shell.

When we have an on-shell intermediate particle, we may replace the numerator with a spin sum,

$$\mathcal{M}_{2\to3} = \bar{v}_B \gamma^\mu \frac{\not{p}_A - \not{p}_\gamma - m_e}{-2p_A \cdot p_\gamma} (e\gamma^\alpha) u_A \times (\cdots) \tag{6.4}$$

$$\approx \bar{v}_B \gamma^\mu \sum_s \frac{u^s\left((1-z)p_A\right) \bar{u}^s\left((1-z)p_A\right)}{-2(p_\perp^2/2z)} (e\gamma^a) u_A \times (\cdots) \tag{6.5}$$

We can then write this in terms of a **soft collinear factorization**

$$\mathcal{M}_{2\to3} = \frac{-z}{p_\perp^2} \sum_s \left[\bar{u}^s\left((1-z)p_A\right)\gamma^\alpha u_A \epsilon_\alpha^a(p_\gamma)\right]\left[\bar{v}_B \gamma^\mu u^s\left((1-z)p_A\right)\right] \times (\cdots) \tag{6.6}$$

$$= \frac{-z}{p_\perp^2} \sum_s \left[\mathcal{M}_{e\to e\gamma}(z)\right]\left[\mathcal{M}_2((1-z)p_A, p_B \to p_1, p_2)\right]. \tag{6.7}$$

The key here is that the first bracket represents the **splitting amplitude** for an electron while the second bracket is just a normal $2 \to 2$ amplitude. The cross section is

$$d\sigma_{2\to3} \approx \frac{1}{2s} d\Pi_\gamma \left(\frac{z}{p_\perp^2}\right)^2 \frac{1}{2} \sum_{s,s',h,a} \mathcal{M}_{e_h \to e_s \gamma_a}(z) M^*_{e_h \to e_{s'} \gamma_a}(z)$$
$$\times\, d\Pi_1 d\Pi_2 \mathcal{M}^s_{2\to2}(s(1-z)) \mathcal{M}^{s'*}_{2\to2}(s(1-z)) (2\pi)^4 \delta^{(4)}(p_A + p_B - p_\gamma - p_1 - p_2). \tag{6.8}$$

We can simplify this since we know that an electron cannot flip helicity by emitting a photon, so that $s = s' = h$. Further...

It is useful to define the CM energy of the $2 \to 2$ amplitude,

$$\hat{s} = (\hat{p}_A + p_B)^2 = 2\hat{p}_A \cdot p_B = 2(1-z)p_A \cdot p_B = (1-z)s. \tag{6.9}$$

6.2 Homework

Simulate 10000 events each for $e^+e^- \to u\bar{u}$ and $e^+e^- \to u\bar{u}g$, at the same $\sqrt{s}$. To get sensible results in the second simulation, impose appropriate cuts to avoid soft and

collinear divergences. (A simple choice is invariant mass cuts, $s_{gu} > s_{\min}$, $s_{g\bar{u}} > s_{\min}$, where $s_{\min} \ll s$. Feel free to come up with your own cuts, though!) Plot the two-dimensional distribution of events in x_u and $x_{\bar{u}}$ where $x_i = 2E_i/\sqrt{s}$. Compare your results with the analytic formula (use your own result from problem 2, or Eq. (3.11) in ESW, or the equation at the bottom of PS p. 261 with $\alpha_g \to \frac{4}{3}\alpha_s$.

7 Pythia

From Peskin slides: PYTHIA will decay known particles isotropically. You might think that this is not adequate, that you would like to include decay matrix elements and spin correlations. If so, you need to simulation more stages of the decay on your side before writing the lhe file. That is, you have to take more of the responsibility onto your side. Ill say more about spin correlations in a moment.

It is possible in the Les Houches accord to define new particles not known to PYTHIA, specific their decay modes and branching fractions, and let PYTHIA generate the decay chains with isotropic decays at each stage. The lhe file contain the spins of final quarks and leptons. PYTHIA does not use this information, but, still, you should think about whether it might be useful. New physics models typically produce highly polarized taus, and tau polarization effects in decay can be large.

Special cases to consider polarization effects: τ and b. (See peskin slides)

8 FeynRules and MG5

Universal FeynRules Output.

Acknowledgements

This work is supported in part by the NSF grant number PHY-0355005, an NSF graduate research fellowship, and a Paul & Daisy Soros Fellowship For New Americans. The contents of this article do not necessarily represent the views of either institution.

A Notation and Conventions

4D Minkowski indices are written with lower-case Greek letters from the middle of the alphabet, $\mu, \nu, \cdots$. 5D indices are written in capital Roman letters from the middle of the alphabet, $M, N, \cdots$. Tangent space indices are written in lower-case Roman letters from the beginning of the alphabet, $a, b, \cdots$. Flavor indices are written in lower-case Roman letters near the beginning of the alphabet, $i, j, \cdots$.

We use the particle physics ('West Coast,' mostly-minus) metric for Minkowski space, $(+, -, -, -)$.

We will also use the conformally flat AdS_5 metric,

$$ds^2 = \left(\frac{R}{z}\right)^2 \left(\eta_{\mu\nu} dx^\mu dx^\nu - dz^2\right). \tag{A.1}$$

Dirac spinors Ψ are related to left- and right-chiral Weyl spinors ($\chi, \bar{\psi}$ respectively) via

$$\Psi = \begin{pmatrix} \chi \\ \bar{\psi} \end{pmatrix}. \tag{A.2}$$

Our convention for σ^0 and the three Pauli matrices $\vec{\sigma}$ is

$$\sigma^0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \sigma^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma^2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma^3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{A.3}$$

with the flat-space γ matrices given by

$$\gamma^\mu = \begin{pmatrix} 0 & \sigma^\mu \\ \bar{\sigma}^\mu & 0 \end{pmatrix} \qquad \gamma^5 = \begin{pmatrix} i\mathbb{1} & 0 \\ 0 & -i\mathbb{1} \end{pmatrix}, \tag{A.4}$$

where $\bar{\sigma}^\mu = (\sigma^0, -\vec{\sigma})$. This convention for γ^5 gives us the correct Clifford Algebra. (Note that this differs from the definition of γ^5 in Peskin & Schroeder.)

B MadEvent Analysis.nb

This is the content of the file `MadEvent analysis.nb`. It is based on the Chameleon package for the LHC Olympics developed by Philip Schuster, Jesse Thaler, and Natalia Toro. It was modified by Maxim Perelstein and Andi Weiler in 2008-2009, and has been slightly changed by Flip Tanedo in 2011. It provides a set of definitions for functions to input an LHE file and analyze its events.

```
(* Based on the Chameleon package by Philip Shuster, Jesse Thaler, \
Natalia Toro *)
(* Modified by Maxim Perelstein and Andi Weiler, \
2008-09 *)

<< Histograms`
(*format of the compulsory event vector*)

eventInfo = {_, _, _, _, _, _};

ClearAll[particle, finout];

(*PDG numbers see e.g. \
http://home.fnal.gov/~maeshima/alignment/ORCA/PYTHIA_particle_codes.\
ps*)

particle[2] = u;
particle[-2] = uBar;

particle[1] = d;
```

```
particle[-1] = dBar;

particle[3] = "s";
particle[-3] = "sBar";

particle[4] = "c";
particle[-4] = "cBar";

particle[5] = b;
particle[-5] = bBar;

particle[6] = t;
particle[-6] = tBar;

particle[11] = e^-;
particle[-11] = e^+;

particle[12] = "\!\(\*SubscriptBox[\"\[Vee]\", \"e\"]\)";
particle[-12] =
  "\!\(\*SuperscriptBox[SubscriptBox[\"\[Vee]\", \"e\"], \"c\"]\)";

particle[13] = "\!\(\*SuperscriptBox[\"\[Mu]\", \"-\"]\)";
particle[-13] = "\!\(\*SuperscriptBox[\"\[Mu]\", \"+\"]\)";

particle[14] = "\!\(\*SubscriptBox[\"\[Vee]\", \"\[Mu]\"]\)";
particle[-14] =
  "\!\(\*SuperscriptBox[SubscriptBox[\"\[Vee]\", \"\[Mu]\"], \
\"c\"]\)";

particle[16] = "\!\(\*SubscriptBox[\"\[Vee]\", \"\[Tau]\"]\)";
particle[-16] =
  "\!\(\*SuperscriptBox[SubscriptBox[\"\[Vee]\", \"\[Tau]\"], \
\"c\"]\)";

particle[21] = gluon;

particle[24] = W^+;

particle[1000006] = Subscript[OverTilde[t], 1];
particle[-1000006] =
\!\(\*SuperscriptBox[
SubscriptBox[
RowBox[{"OverTilde", "[", "t", "]"}], "1"], "*"]\);
particle[1000022] = Subscript[OverTilde[\[Chi]], 0];

finout[-1] = in;
finout[1] = out;
finout[2]  = decayed;

ReadME[rawInput_ ] :=
  (
   (*Search for beginning of events*)
```

```
   pos = Position[rawInput, {"</init>"}][[1, 1]];
   (*Throw away crap at start of file,
   combine events in {} array and remove the XML commands and \
compulsory eventinfo *)

   DeleteCases[
    Split[Drop[ rawInput,
      pos], # =!= {"</event>"} &], {"<event>"} | \
{"</LesHouchesEvents>"} | {"</event>"} | eventInfo, 2]
   );

EventPrint[ event_ ] := (
   Print[Join[{{pid, in/out, mother1, mother2, color1, color2, px, py,
         pz, p0, mass, c\[Tau], hel}},
      event /. {x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_, x9_, x10_,
         x11_, x12_, x13_} -> {particle[x1], finout[x2], x3, x4, x5,
         x6, x7, x8, x9, x10, x11, x12, x13}] // MatrixForm]);

(* those are input or decayed particles*)

decayedOrIn = {_, -1, _, _, _, _, _, _, _, _, _, _, _}  | {_,
    2, _, _, _, _, _, _, _, _, _, _, _};

EventOut[ event_ ]  := (DeleteCases[event, decayedOrIn]);

SetOut[ event_ ]  := (DeleteCases[event, decayedOrIn, 2]);

(* EffMassAll[objList_]  :=Plus @@ Map[ptOf,objList//Transpose,{2}]; *)
\

EffMass[event_]  := Plus @@ Map[ptOf, event, {1}];

pT[event_]  := Map[ptOf, event, {1}] // Flatten;
eta[event_] := Map[etaOf, event, {1}] // Flatten;
theta[event_] := Map[thetaOf, event, {1}] // Flatten;
threeVector[event_] := Map[ThreeVectorFrom, event, {1}];


MissingEnergy[event_] := Plus @@ Map[enOf, event, {1}];

ptOf[{_, _, _, _, _, _, px_, py_, ___}] := Sqrt[px^2 + py^2];
enOf[{_, _, _, _, _, _, _, _, _, En_, ___}] := En;
thetaOf[{_, _, _, _, _, _, px_, py_, pz_, ___}] :=
  ArcCos[pz/Sqrt[px^2 + py^2 + pz^2]];
etaOf[{_, _, _, _, _, _, px_, py_, pz_, ___}] := -
   Log[ Abs[Tan[ ArcCos[pz/Sqrt[px^2 + py^2 + pz^2]]/2]]];
FourVectorFrom[{_, _, _, _, _, _, px_, py_, pz_, En_, ___}] := {En,
   px, py, pz };

FourLength[{pe_, pz_, px_, py_}] :=
  Sqrt[Max[pe^2 - pz^2 - px^2 - py^2, 0.0]];

ThreeVectorFrom[{_, _, _, _, _, _, px_, py_, pz_, ___}] := { px, py,
```

```
   pz };

CosthetaTwoJet[{jet1_, jet2_}] := (a = ThreeVectorFrom[jet1];
   b =  ThreeVectorFrom[jet2]; a.b/Sqrt[a.a b.b]);

GetAll[evt_] := evt;
all[evt_] := True;

Freq[evtList_, crit_, objsel_, plotfunc_, {min_, max_, nbins_}] :=
  BinCounts[
   Flatten[
    plotfunc /@ Flatten4[objsel /@ Select[evtList, crit]]], {min,
    max, (max - min)/nbins}];

Hist[freqcombo__, {min_, max_, nbins_}, HistogramOptions___] :=
  Histogram[freqcombo /. Bin[args__] -> Freq[args, {min, max, nbins}],
    FrequencyData -> True,
   HistogramCategories -> Table[i, {i, min, max, (max - min)/nbins}],
   HistogramOptions];

Flatten4[List_] := Flatten[List, Max[0, Depth[List] - 4]];

HGet[patt_, All][evt_] := Cases[evt, patt];

oJetlhe =  {2, 1, _, _, _, _, _, _, _, _, _, _, _}  | {-2,
    1, _, _, _, _, _, _, _, _, _, _, _} | {21,
    1, _, _, _, _, _, _, _, _, _, _, _}  ;

oMissingEnergy = {8888, 1, _, _, _, _, _, _, _, _, _, _, _}  ;

PTSort[event_] := Sort[event, Greater[ptOf[#1], ptOf[#2]] &];


and[f_][evt_] := f[evt];
and[f_, g__][evt_] := f[evt] && and[g][evt];
or[f_][evt_] := f[evt];
or[f_, g__][evt_] := f[evt] || or[g][evt];

join[f_][evt_] := f[evt];
join[f_, g__][evt_] := Join[f[evt], join[g[evt]]];
```