SLAC-PUB-8290
October 1999

Pandora: an Object-Oriented Event Generator for Linear Collider Physics

Michael E. Peskin[a]
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94309 USA

ABSTRACT

I desribe a new event generator, pandora, which uses the C++ class structure to allow a modular treatment of beams and particle production and decay.

presented at the International Workshop on Linear Colliders
Sitges, Barcelona, Spain, 28 April – 5 May 1999

[a] Work supported by the Department of Energy, contract DE–AC03–76SF00515.

PANDORA: AN OBJECT-ORIENTED EVENT GENERATOR FOR LINEAR COLLIDER PHYSICS

MICHAEL E. PESKIN
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94309 USA

I describe an new event generator, pandora, which uses the C++ class structure to allow a modular treatment of beams and particle production and decay.

1 Introduction

The ideal LC event generator needs to fulfill a number of requirements. It should provide the basic standard and nonstandard processes in e^+e^- annihilation. It should also include beamstrahlung and initial state radiation, include initial-state polarization, and include the full set of final-state correlations associated with polarization effects. It should also allow input of any parton-level cross section and should be able to generate events from this cross section with reasonable efficiency.

As a step toward these goals, I would like to introduce a new generator, called pandora. The idea of pandora is to package the various stages of a LC event as distinct C++ classes which interact through a simple interface. This makes it possible to include new parton-level processes in a simple way, allowing the larger system to take care of the initial beams and final interactions.

Pandora is implemented in C++ as a `pandora` class, whose constructor depends on two `beam` classes and one `process` class. Utility classes provide the distributions for W, Z, and top quark decay. All of the classes make use of an `LVector` (Lorentz vector) class which provides a variety of 4-vector operations. A complete main program for parton-level $e^+e^- \to t\bar{t}$ events is shown in Table 1. The current software distribution and documentation can be found at ref. 1.

2 Event selection

The basic concept of pandora is to represent a cross section for a complete e^+e^- process—from the initial electron and positron to the final partons—as an integral over N variables x_i which run over $[0,1]$. The x_i parametrize all relevant variables in the process, from beamstrahlung energy loss to resonance decay angles. The expression $d^n\sigma/dx^n$ is handed to a general purpose program

Table 1: A simple program with pandora

```
#include "pandora.h"
#include "eetottbar.h"
#include "beams.h"
int main(){
    double ECM = 500.0
    ebeam B1(ECM/2.0, electron, electron, 0.9, 1, 1, NLC500);
    ebeam B2(ECM/2.0, positron, positron, 0.0, 1, 1, NLC500);
    eetottbar R;
    pandora P(B1,B2,R);
    int Ncalls = 100000;
    VegasGrid V = P.prepare(Ncalls);
    for (int i =1;i <= 10;i++)
       {LEvent LE = P.getEvent(V);cout << LE ;}
    return 0;}
```

to select weight-1 events. Then each chosen value of $\{x_i\}$ is converted to an event handed back in a standard format specified by the `LEvent` class.

In pandora, event selection is done using the VEGAS algorithm to optimize a grid in the N-dimensional space and then choosing weight-1 events in the metric defined by this grid. This is the algorithm used in Kawabata's BASES/SPRING program.[2] VEGAS does do useful adaptation to the function being integrated, but the algorithm is inefficient if the peaks of the function being integrated are not aligned with the grid. Methods recently proposed by Ohl [3] and Jadach [4] may ameliorate this problem. For the moment, we accept a loss in speed of event generation as the price of generality.

3 Beam class

The parametrizations of beamstrahlung and initial-state radiation are contained in classes which are derived from the abstract class `beam`. The formulae for beamstrahlung used in pandora are based on the 'consistent Yokoya-Chen' approximation explained in ref. 5. They are somewhat simpler than earlier analytic formulae in the literature and agree just as well (or poorly) with simulation data. Both e^+e^- and e^-e^- (the latter for round beams only) are considered. The `ebeam` class constructor used in Table 1 takes as arguments, the nominal beam energy, the beam species, the species initiating the considered reaction (which might, for example, be a photon from an electron beam), the beam polarization, flags for the inclusion of beamstrahlung and initial-state

Table 2: Definition of the pandora process class

```
class process{ public:
  process(int N): n(N){}
  virtual int validEvent(DVector & X, double s, double beta) = 0;
  virtual DVector crosssection(DVector & X, double  s) = 0;
  virtual LEvent buildEvent(DVector & X, double s) = 0;
               private:
  int n;  /* number of integration variables */   };
```

radiation, and the name of a reference machine design. A more complicated constructor allows input of arbitrary beam parameters.

4 Process class

The reaction cross sections and decay distributions are contained in classes which are derived from the abstract class `process`. A subclass of `process` must define three functions, which give the allowed domain of the variables x_i, the value of the differential cross section, and the event corresponding to a chosen set $\{x_i\}$. The definition of `process` is given in Table 2. The differential cross section is returned as a 4-component vector for the four possible orientations of initial helicities.

More generally, pandora returns amplitudes as matrices indexed by polarization, which are multiplied to obtain cross sections with full spin correlations. As an example, Figure 1 shows the distributions of top quark decay angles returned by pandora. The helicity amplitudes for particle production and decay are the basic raw materials for linear collider physics studies. It is part of my plan to provide a compilation of the amplitudes needed for the most imporant LC processes.

5 Interface to PYTHIA

Pandora returns parton-level final states and does not carry out hadronization. This is acceptable if the final states can be hadronized by a general-purpose simulation program such as PYTHIA.[6] Since PYTHIA lives in the FORTRAN world, its coupling to pandora must be somewhat inelegant. However, Barklow and Iwasaki have written a general interface which includes pandora processes as subprocesses in PYTHIA event generation. Tau leptons are decayed using TAUOLA,[7] taking account of their longitudinal polarization, before the event

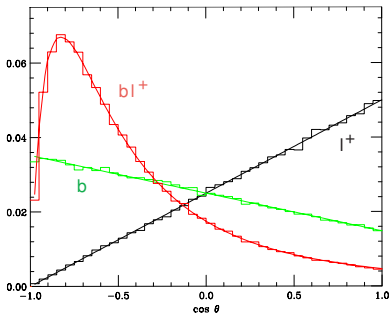

Figure 1: Distribution of decay angles of the ℓ^+ and b in top decay, measured with respect to the top spin direction, and the angle between the ℓ^+ and b, comparing pandora results to lowest-order theory.

is hadronized. To facilitate the interface to PYTHIA, pandora returns in the `LEvent` class the color contractions of the final partons and the order in which partons are to be taken in pairs to compute QCD showers. This interface, `PANDORA_PYTHIA`, is avaiable from the pandora Web site.[1]

Acknowledgments

I am grateful to Tim Barklow and Masako Iwasaki for their many contributions to this project, and to David Gerdes and Ester Ruiz Morales for helpful criticism. This work was supported by the US Department of Energy under contract DE–AC03–76SF00515.

References

1. `www.slac.stanford.edu/~mpeskin/LC/pandora.html`.
2. S. Kawabata, Comp. Phys. Comm. **41**, 127 (1986), Comp. Phys. Comm. **88**, 309 (1995).
3. T. Ohl, Comp. Phys. Comm. **120**, 13 (1999), hep-ph/9806432.
4. S. Jadach, physics/9910004.
5. M. E. Peskin, LCC report LCC-0010 (1999).
6. T. Sjostrand, Comp. Phys. Comm. **82**, 74 (1994), hep-ph/9508391.
7. S. Jadach, Z. Was, R. Decker and J. H. Kuhn, Comp. Phys. Comm. **76**, 361 (1993).