The Pythia-JNI Package: A Java Interface to Pythia

M. Ronan*
Lawrence Berkeley National Laboratory
University of California, Berkeley
DATED: JANUARY 1, 2002

The Pythia-JNI package is a stand-alone software system for setting up and running the Pythia Monte Carlo program from within Java. Full Pythia functionality is provided through a simple "straght-forward" interface to native FORTRAN code. The package includes examples for running Pythia within a Java framework for Linear Collider detector (LCD) simulations. These LCD examples use Pythia for event generation, execute the U.S. Fast Monte Carlo (FMC) detector simulation and perform user analysis in Java and/or FORTRAN.

1. Introduction

The Lund Monte Carlo program Pythia [1] is often used for event generation and hadronization in High Energy Physics (HEP) studies. The Pythia FORTRAN implementation will likely be used for quite some time. Meanwhile, the Java Language Environment [2] offers the possibility of developing physics analysis for the next generation of HEP facilities in a clean object-oriented methodology. A reliable Java interface to Pythia would allow new physics and detector studies to take advantage of the existing Pythia code.

The Pythia-JNI package [5] has been developed to provide a convenient general purpose Java interface to the Pythia Monte Carlo system. Basic physics generation and fragmentation processing is provided through a Java Native Interface (JNI) to underlying Pythia routines, such as pygive, pyinit, pyexec and pylist. In a Java framework package, the generated events are accessed through a JNI interface to the HEPEvt common block and passed to Java analysis modules. Provision has been made to run existing User FORTRAN code through a UserAnalysis class and corresponding User-JNI interface.

A Java software framework has been developed for Linear Collider detector (LCD) [3] simulations. In that framework, the Java Analysis Studio (JAS) [4] provides a powerful environment for reading in existing Monte Carlo event files and running LCD reconstruction and analysis applications. For unique or large statistics physics and fast detector studies, specialized stand-alone applications can generate and analysis events saving intermediate histograms and results for subsequent analysis. JAS can then be used to view histograms, analyze intermediate results and prepare final presentations. A number of simple stand-alone Java analysis examples for Linear Collider physics analysis and Fast Monte Carlo (FMC) detector simulations are discussed below.

2. Interfacing to Pythia

The Java Development Kit (JDK) [2] provides easy to use tools for constructing interfaces to legacy code such as Pythia and the HEPEvt FORTRAN common block. The developer describes the Java signature of underlying "native" methods with the required arguments and return variables. The JDK writes a header file specifying how the interface should be implemented. A simple ANSI-C implementation is used to invoke Pythia routines or to access HEPEvt event variables.

The **hep.generator.pythia** Java package contains a "straight-forward" interface to Pythia through simple interfacing methods. The **Pythia** Java class describes the signature of Pythia routines such as pyinit, pyexec, pylist, etc. Since the naming of Java packages and classes are unique, these same methods can be accessed simply as init, exec, list, etc, as illustrated below. A simple implementation **PythiaImp.c** provides the interface from the Java language methods to the

*ronan@lbl.gov; Presented by W. Yao

underlying FORTRAN routines. A standard GNU Makefile is used to compile the Java classes, make the native interface definitions and compile the C implementations. Links to Pythia and CERN object libraries are made in a shared object library that is loaded into the Java Virtual Machine (VM). Pythia can then be executed from its Java main method or included in applications.

```
Useage:
        To create the Pythia-JNI interface
                 Pythia pythia = new Pythia();

        To set up a "User-defined" process
                 pythia.give(parameters);
        Where parameters is a String of Pythia settings

        To initialize Pythia, one can use
                 pythia.init("CMS","e+","e-",Ecm);

        After Pythia event generation one can use the
        Pythia list facility with
                 pythia.list(1);
```

A **PythiaProcess** class has been provided to allow dynamic loading of predefined process settings into Pythia, such as **eetoZH**, **eetottbar**, etc. The LCD **PythiaGenerator** class, described in Sec. 3.3.2, uses a given process name to dynamically load the appropriate Java class which in turn sets the required Pythia configuration.

A **HEPEvt** class, derived from the JAS [4] package, provides a Java interface to the HEPEvt FORTRAN common block. An ANSI-C implementation **HEPEvtImp.c** is used to interface the Java methods to the HEPEvt data structure viewed from C. Java native methods such as getNEVHEP() and getNHEP() return the generated event number and number of particles, respectively. While methods such as getPHEP(i,j) return double precision values for the four momenta of each particle. The HEPEvt-JNI is loaded in the Pythia shared object library to allow access to the generated event quantities.

3. Java Framework

A simple Java framework for data processing has been developed as part of the JAS [4] environment. This base framework has been extended for Linear Collider detector (LCD) [3] simulations. In stand-alone Pythia-JNI packages [5], the main program uses the base framework in generating the Pythia events. The LCD simulation and analysis framework is then used for Fast Monte Carlo detector simulation and user analysis.

3.1. Event generation

The **hep.analysis** package provides a basic Java simulation and analysis framework. A **Job** class has methods for adding **EventSource**'s and **EventAnalyzer**'s to the processing stream. When the processors have been loaded the **Job** executes a specified number of events, writing out **StdHEP** events if requested, and then saves histograms and any other output at termination.

In simulations, a main program method creates an analysis job and then adds Pythia as the event source. Various LCD simulation and analysis modules are then added as described below.

3.2. LCD Software (The hep.lcd Class Library)

The **hep.lcd** class library [3] provides an extensible Java framework for running sophisticated analyses such as fast parameterized Monte Carlo (FastMC) simulation and full event reconstruction. The LCD simulation system allows analysis of generator-level four vectors, FastMC quantities or space point data from the full tracking package. Use of the Java Language Environment allows

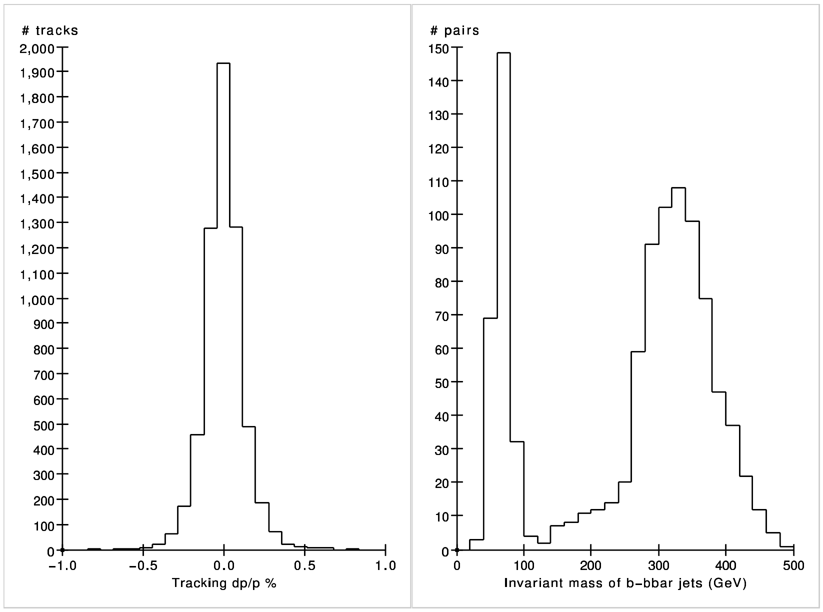

Figure 1: Comparison of simulated tracks with parent charged Monte Carlo particles (a), and invariant mass of b-bar jets in 500 GeV e^+e^- collisions.

rapid development of analysis modules and reconstruction algorithms with excellent throughput in processing. A suite of physics analysis tools, such as histograming, event display and jet finding, enable users to build on proven code. The overall design emphasizes flexibility and extensibility, typically providing multiple algorithms in following object-oriented design rules. The framework can be used as a standalone or run inside Java Analysis Studio. Within the modular LCD design, **Driver** objects receive the **LCDEvent** data from the Monte Carlo generator and execute a list of **Processor**'s.

The **PythiaGenerator** class in the **hep.lcd.generator.pythia** package provides a wrapper to the Pythia Java interface described in Sec. 2. It executes Pythia methods to generate each event and then reads out the **HEPEvt** common block in forming a valid **LCDEvent** object.

The **hep.lcd.mc.MCFast** processor performs a simple parameterized Fast Monte detector simulation with straight forward acceptance cuts on final state particles. Charged particle momentum resolution tables calculated for tracks at various momenta and angles are used for smearing the generated particle momenta. Simple parameterized energy resolutions are used for photons and neutral hadrons.

Java analysis modules can simply be added to the LCD framework as described below. A User-JNI is available to fill ntuples for use in FORTRAN analyses or to be written out for subsequent analysis.

4. Java Analysis Examples

In the event generation and analysis examples, a main **GenPythia** class configures Pythia for $e^+e^- \to b\overline{b}$ event generation at 500 GeV. It creates a **hep.analysis.Job** adding Pythia as the event source. A simple analysis **Driver** executes a **ShapeAnalysis** process to determine the event Thrust axis and then executes the Fast Monte Carlo simulation. The analysis **Processor**'s described below are then executed.

The **PythiaEventAnalysis** class gets the tracks and clusters from the simulated event, writes out the quanties for the first few events and makes some simple histograms. In the plot shown in Figure 1a, simulated track momenta are compared to the generated charged particle momenta. The plot shows an average charged particle momentum resolution of about 0.1 % for the Large TPC detector.

The **PythiaJetAnalysis** class uses tracks and clusters to find jets in simulated $e^+e^- \to b\overline{b}$ events. The reconstructed jets are compared to the partons and leptons from Pythia event generation.

The jet-jet invariant mass for b-bbar jets are shown in Figure 1b. The plot shows a broad signal somewhat below the CM energy due to initial state radiation and a peak due to radiative return to the Z resonance.

Acknowledgments

I'd like to thank my colleagues developing the LCD Java software system. Marco Battaglia and Weiming Yao provided useful comments on early versions of the Pythia-JNI software.

References

[1] Pythia 6.152, "The Lund Monte Carlo", T. Sjostrand, Computer Physics Commun. 82 (1994) 74, http://www.thep.lu.se/~torbjorn/Pythia.html.
[2] JDK1.1.8 and higher, Sun Microsystems, Inc., http://java.sun.com.
[3] M. Ronan et al, "Java Analysis Studio and the hep.lcd Class Library", International Workshop on Linear Colliders, Sitges, Barcelona, Spain, 1999; G. Bower et al., "Java-based LCD Reconstruction and Analysis Tools", International Linear Collider Workshop, Fermilab, 2000; and N. Graf et al., "LCD Software Status", in these proceedings.
[4] A.S. Johnson, *Java Analysis Studio*, http://jas.freehep.org/.
[5] M. Ronan, http://obsidian.lbl.gov/~ronan/docs/Pythia-JNI.