Installation of the Simulation Toolkit AnkeRoot

V.Hejny, November 20, 2002

1 Introduction

This set of classes is a add-on to PLUTO++[1] which allows to analyze the produced ROOT files with respect to the ANKE acceptance. Like in GEANT the particles are tracked in the magnetic field and hits in the individual detectors are registered. However, neither energy loss nor other physics processes are taken into account. **This program is meant for fast and efficient first simulations to check what basically is happening for a specific reaction.**

2 Requirements

The code will be maintained such, that the most actual version of ROOT and PLUTO will work. The minimum requirements are ROOT version 3.02/07 and PLUTO version 3.58p3, the recommended versions are 3.03/09 and 3.58p3, respectively. The PLUTO version is a patched 3.58, the one available at GSI still lacks of some patches. These packages are available on ankecvs.ikp.kfa-juelich.de[2].

3 Installation

For installation of ROOT, please refer to documentation available on the ROOT homepage[3]. If ROOT is set-up properly (including the environment variables ROOTSYS and LD_LIBRARY_PATH) PLUTO should compile out of the box. The environment variable PLUTOSYS must point to the installation directory of PLUTO, the path to libPluto.so must be added to LD_LIBRARY_PATH.

The AnkeRoot package itself can be fetched via cvs from ankecvs. For this you have to execute:

```
> cvs -d :pserver:anke@ankecvs.ikp.kfa-juelich.de:/data/cvs login
Logging in to :pserver:anke@ankecvs.ikp.kfa-juelich.de:2401/data/cvs
CVS password:
> cvs -d :pserver:anke@ankecvs.ikp.kfa-juelich.de:/data/cvs co AnkeRoot
```

You will end up with a subdirectory called AnkeRoot.

This is recommended way to fetch the software. However, if you do not have access to the host ankecvs[4], you can also fetch the package from the same place as ROOT and PLUTO (you have to keep in mind, that this package may be out-of-date). You need AnkeRoot_vx.xx.xx.tar.gz containing the basic software.

Independent from the way you fetched the software, you have to fetch a fieldmap from ankecvs (same place, again). You can download a general 1.6 T fieldmap, which will be scaled later on, and/or the fieldmaps currently used in Geant3.

[1] http://www-hades.gsi.de/computing/pluto/html/PlutoIndex.html
[2] http://ankecvs.ikp.kfa-juelich.de/ftp/Root
[3] http://root.cern.ch
[4] This may be possible from outside FZJ, because all hosts have to registered in order to pass through the firewall.

To compile the code a simple 'make' should be enough on any Linux computer. A shared library Anke.so will be built, which has to be loaded into ROOT afterwards. (Of course, libPluto.so must be loaded in advance.)

While the library 'Anke.so' will be made with a plain 'make', the example analysis class ExampleAnalysis is created with a 'make ExampleAnalysis.so'.

4 Magnetic field

Three methods (classes) describing the magnetic field at ANKE are provided:

AnkeFieldBox A simple box field (not optimized).

AnkeFieldMap A standard map for ANKE: 1.6T only, will be scaled to the actual value (the environment variable FIELDMAPFILE has to be set to the file name, default is AnkeFieldmap.dat).

AnkeFieldG3 An interface to the fieldmaps used in Geant3 (the environment variable G3MAPDIR has to set to the directory, which contains the fieldmaps, default is FIELDMAPS).

A pointer to the class object has to be passed to 'gAnkeSetup' using 'gAnkeSetup→SetFieldClass(...)'. If no specific magnetic field is selected, AnkeFieldMap will be used.

5 Basic tracking

For the following, assume you already have prepared the event file 'pluto_output.root' (beam momentum 2.4 GeV/c) using PLUTO. The global pointer 'gAnkeSetup' is initialized automatically when loading Anke.so.

1. Select the ANKE setting using SetField(momentum, field, angle) field or angle have to be 0. indicating that this value should be calculated. (Here: 2.4 GeV/c, 1.6 T)
   `root [] gAnkeSetup→SetField(2.4,1.6,0.0);`
2. Create a run and open the root file
   `root [] AnkeRun run("pluto_output.root");`
3. Process some events (arguments: start event, number of events)
   `root [] run.ProcessEvents(0, 100);`
4. Draw the result
   `root [] run.Draw();`
5. Close the file
   `root [] run.CloseFile();`

That's essentially all. If you want to speed things up, you can disable storing hits for later analysis with 'gAnkeSetup→StoreHits(0)'.

There some more options to set:

gAnkeSetup→StoreTrajectories(flag) Enable/disable trajectory storing. For plotting yoy have to keep it on, of course. For analysis of a large number of events (see below) you should really switch it of.

gAnkeSetup→SetFieldClass(AnkeField* f) If you want to use a different field, you have to instantiate it (see above) and pass the pointer

gAnkeSetup→SetTargetPosition(TVector3 v) This you have to use, when you don't use the famous K-mode. Default is (-8,0,-100)

gAnkeSetup→DecayParticles(flag) Let the particles decay in flight or not

gAnkeSetup→StepSize(size) Step size in cm; default is 5cm. The actual step size used for tracking is 5 times smaller.

6 Analysis

Looking for hits in ANKE makes only sense if one is able to analyze these data. Therefore an abstract base class 'AnkeAnalysis' is provided, which has to be used to derive own anaylsis classes. 'ExampleAnalysis' serves as an example. The class has to have basically two functions:

void Initialize() In this member function basic variables can be set. In ExampleAnalysis the four momentum of the initial state is calculated and the histogram is reset.

void Analyze(Int_t event_number) This is the main analysis function called after each event. A map of type 'pMapType' is used to hold information of each particle and track combination in one event. Here, pairs of pointers to PParticle and AnkeTrack, respectively, are stored. How this can be used is shown in ExampleAnalysis.

To enable an analysis, the following steps have to be done additionally to the ones above ('Store-Hits()' must not be disabled).

```
root [] ExampleAnalysis ana;
[...]
root [] run.SetAnalysisClass(&ana);
root [] ana.Initialize();
```

Keep in mind, that you call Initialize() **after** SetField() if you use the momentum , e.g. for calculation of the initial state.

In the example you can have a look at the histogram using ExampleAnalysis:: fHAlphaMiss→Draw() afterwards.

7 Direct use of PLUTO

You don't have to create a root file beforehand. Instead you can use directly the pluto output for further analysis:

```
root [] gAnkeSetup→SetField(2.43,1.00,0.0);
root [] AnkeRun run("d","d","alpha eta [pi+ pi- pi0 [g g]]");
```

```
root [] run.ProcessEvents(0,100);
root [] run.Draw();
```

The reaction string is parsed and the appropriate particles, channel and the reaction is set-up.

As a forth argument, you can pass a ROOT filename (without .root extension) to the AnkeRun constructor. In that case, the Pluto output will be written in standard Pluto format to file. No processing of the data through AnkeRoot is done!

8 Adding more detectors

New detectors can be added hard-coded or just via a macro. For testing and private setups the latter should be prefered. As an example the macro ExampleGeometry.C is provided. See also README.newdetectors in the package.

9 Further help

For further details and tricks ask me personally.