Fitting Tutorial

Introduction

The following set of exercises is meant to drive you through the AIDA interfaces. The suggested exercises are ordered by increasing complexity. Feel free to start from the complexity level you feel comfortable with. Our solutions to the exercises are also provided both in java and in the pnut scripting language. You can run them with JAS3 or JAIDA.
Please use the AIDA API documentation as your main reference.

Data File

Before you get started it is necessary to download the file JAS3DataSet.aida and to open it in JAS3.

Please take some time to inspect it; it contains the following objects:

- **dataPointSet line**: an IDataPointSet with points distributed according to a first order polynomial
- **dataPointSet parabola**: an IDataPointSet with points distributed according to a second order polynomial
- **tuple**: an ITuple with the following columns (and the corresponding distribution)
    - **gaussSum**: he sum of two gaussians with different mean
    - **sqroot**: a square root distribution
    - **doubleGauss**: the sum of two gaussians with the same mean
    - **signalAndBkg**: the sum of a first order polynomial background and a gaussian distributed signal
    - **lifetime**: the sum of an exponentially decaying signal and a gaussian distributed background
- **gauss Sum Hist**: an IHistogram1D corresponding to the projection of the *gaussSum* column above
- **sqroot Hist**: an IHistogram1D corresponding to the projection of the *sqroot* column above

Opening the Data File

Creating an AIDA Tree

With the aida standards it is possible to open the JAS3DataSet.aida file (assuming it is located in C:/Examples/JAS3 Tutorial/) in the following way:

```
IAnalysisFactory analysisFactory = IAnalysisFactory.create();
ITree tree = analysisFactory.createTreeFactory().create("C:\\Examlpes\\JAS3 Tutorial\\JAS3DataSet.aida");
```

Accessing the JAS3 Navigation Tree

Once a file has been open by hand it is possible to access its contents via the JAS3 navigation tree. In the code skeletons below show how to access the *ai daMasterTree* and *cd* into the already opened file.
This is a JAS3 specific way to access the main tree *aidaMasterTree* with which you can navigate the whole tree (the one displayed on the left of the JAS3 panel). The advantage is that you need to open the file only once, accessing it then via the aidaMasterTree. The disadvantage is that the code is no longer AIDA compliant.

Use the following skeletons as a starting point for your pnut script or java code if you want to use this approach.

Pnut

```
use("pnuts.lib")

IAnalysisFactory = class hep.aida.IAnalysisFactory;
af = IAnalysisFactory::create()

//Use the aidaMasterTree as the main ITree
aidaMasterTree.cd("/JAS3DataSet.aida");

//.... your code goes here
```

Java

```
import hep.aida.*;
import org.freehep.application.*;
import org.freehep.application.studio.*;

public class myExample {

    public static void main( String[] argv ) throws java.io.IOException {

        IAnalysisFactory af = IAnalysisFactory.create();

        ITree tree = (ITree) ((Studio) Application.getApplication()).getLookup().lookup(ITree.class);

        tree.cd("/JAS3DataSet.aida");

        //..... your code goes here
     }
}
```

Exercises

1. Perform a binned fit to the IDataPointSet **dataPointSet line** (java,pnut)
2. Create a scripted second order polynomial model and perform a binned fit to the **dataPointSet parabola** IDataPointSet (java,pnut)
3. Perform two separate binned fits to the histogram **gauss Sum Hist** by restricting the axis range to only one gaussian at a time. Ranges can be controlled via the IFitData interface (java,pnut) An alternative way is to perform two projections of the **tuple**'s column **gaussSum** onto IHistogram1Ds by filtering out the unwanted range, performing then the fit over the whole range of the histograms. Filtering on the tuples is done with IFilter and IEvaluator interfaces (java,pnut)
4. Extend example 3 by creating a scripted sum of two gaussians to fit the histogram **gauss Sum Hist** over the whole range using the two previous fits to extract the starting values of the parameters (java,pnut)
5. By filtering on the signal part of the **tuple**'s column **signalAndBkg** project the **lifetime** column onto an IHistogram1D and fit it with an exponential distribution (either scripted or using the built-in one) (java,pnut)
6. Project the **tuple**'s column **doubleGauss** onto a IHistogram1D and perform a binned fit with the scripted function created in example 4 setting as a constraints that the two gaussians have the same mean (java,pnut)
7. Extend exercise 6: use the fitted parameters from the binned fit to initialize an unbinned maximum likelihood fit to the **tuple**'s column **doubleGauss** (java,pnut)

It is possible to perform some of the above exercises (the easier ones) through the GUI, by graphically manipulating functions and controlling the fit.