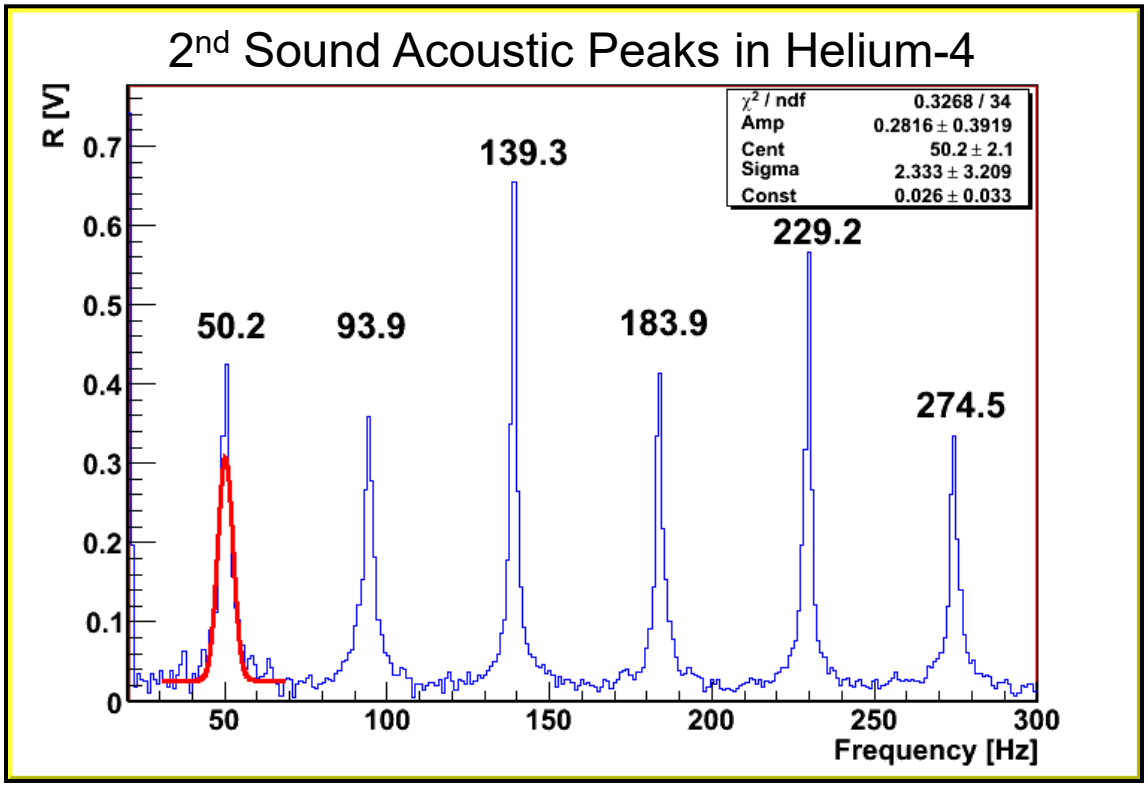
Introduction to ROOT

root.cern.ch



Want a big sledgehammer for free?

J. Long, from D. Hertzog,

27 January 2026

A completely self-installing system is available for you to download to many platforms

- https://root.cern.ch/install/all_releases/

Compiled binaries. For Windows, is complete*, just like any professional program installation.

Windows
Mac OS X
Linux
Solaris
…

Latest stable release: 6.36.06

Recent version on muon experiment Linux computer

Legacy Windows version (4.04.02) on Grainger server at Z:\APL Courses\PHYCSNew\root\bin

***May need to install dependencies (e.g.: MS Visual Studio): see https://root.cern.ch/install/dependencies/**

(Latest Windows version I could install/run: 5.34)

What can ROOT do?
Who makes ROOT?

- **General: much more than you need**
  - Data analysis environment
    - large sets
  - Graphics, histograms, fitting
  - Monte Carlo
  - 3D visualization
- **Users support ROOT, but center is at CERN**
  - Particle physics community main standard
  - Nuclear physics community, growing standard
  - Others using it for selected tasks
- **Documentation**
  - https://root.cern/doc/v636/

Once you install it, click to run

- **Start root by clicking on the icon**
- **Moving around in directories:**
  - **root[ ] .dir (lists the files in current dir)**
  - **root[ ] .cd xxx change to directory xxx**
  - **root[ ] .x macro.cpp (execute a "macro.cpp")**
  - **root[ ] .L stuff.cpp (load up the file stuff.cpp)**
    - **root[ ] stuff() ← call function in stuff.cpp to run**

You can pick up some simple tutorials we put together, now located on P403 server under "\Common\MyRoot:"

\\engr-file-03.engr.illinois.edu\PHYINST\APL Courses\PHYCS403\Common\MyRoot\Pre-2017 scripts, macros and data

Start with some introductory commands. Study the macros. They are not fancy. All commands can be typed one by one at the command prompt if you prefer.

- **Open any text editor (emacs, WinEdt, notepad), open/drag the following files into the window:**
  - **basichist.cpp – simple histogram; very primitive**
  - **basicgraph.cpp – connects pairs of (x,y) points**
  - **basicfit.cpp – provides some data to use with fit panel GUI**
  - **basicfunc.cpp – draws a function**
  - **basicsimulator.cpp – fills hist with user func and stat fluctuations**

Some often used command lines

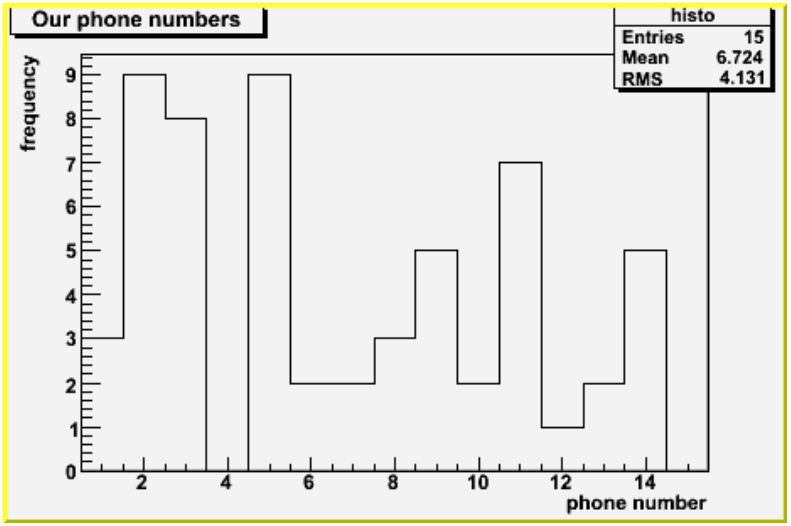
root [1] .ls ← What histograms do I have loaded up ?
TROOT* Rint The ROOT of EVERYTHING
OBJ: TH1F **histo** Our phone numbers : 0 at: 0B4ED3B0

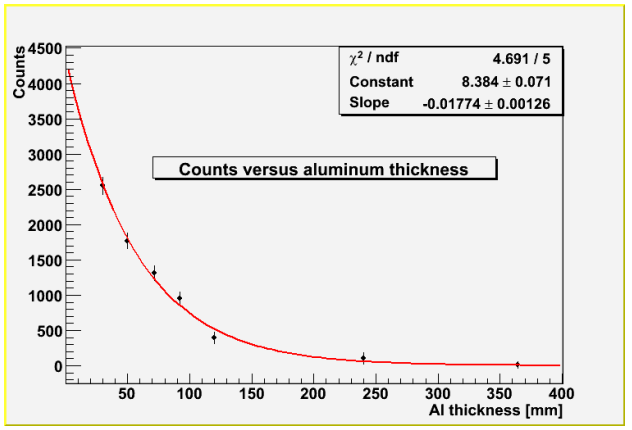
root [2] histo->Draw()
root [3] histo->DrawPanel() ← bring up drawing GUI
root [4] histo->FitPanel() ← bring up fitting GUI
root [5] histo->SetLineColor(4) ← make the trace blue
root [6] histo->Integral()
(const Stat_t)5.8000000000000000e+001
root [7] 179.3/(1.2e4*3.14) ← a handy inline calculator
(const double)4.7584925690021236e-003

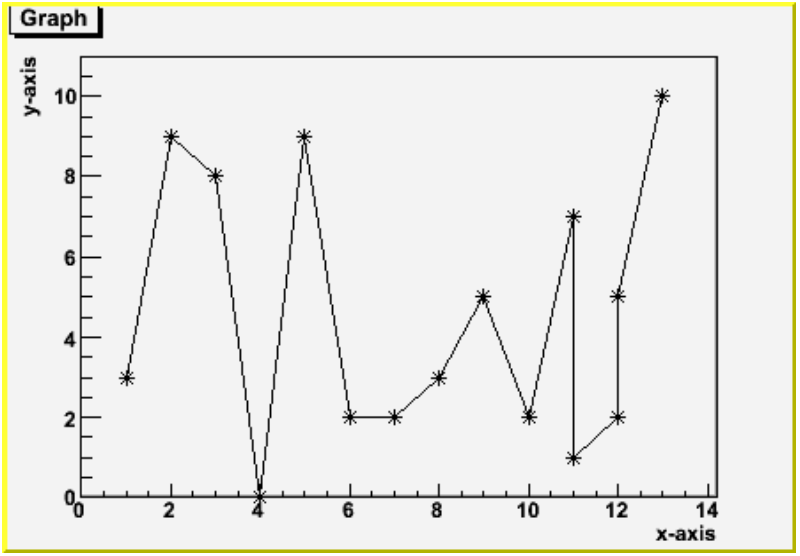
When you see the plot in the "Canvas", go to **View** menu and click on **Editor** and **Toolbar**

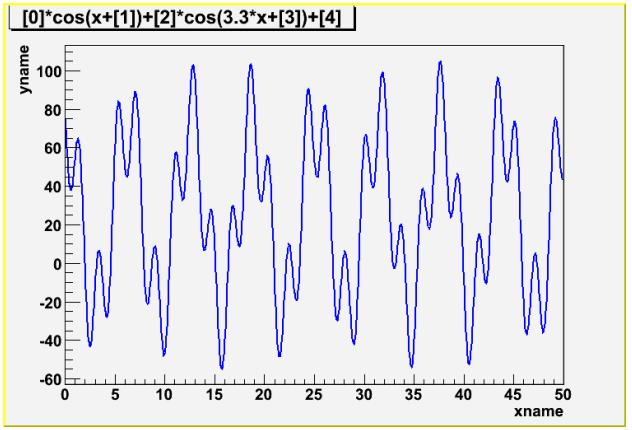
Note: exponential fit function is $y = e^{(a+bx)}$

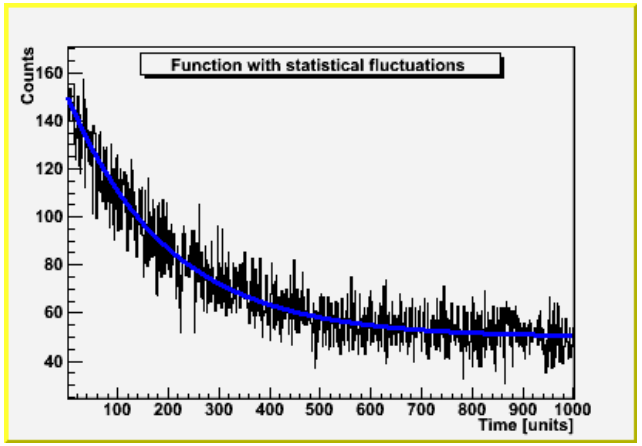
Snapshots of the output









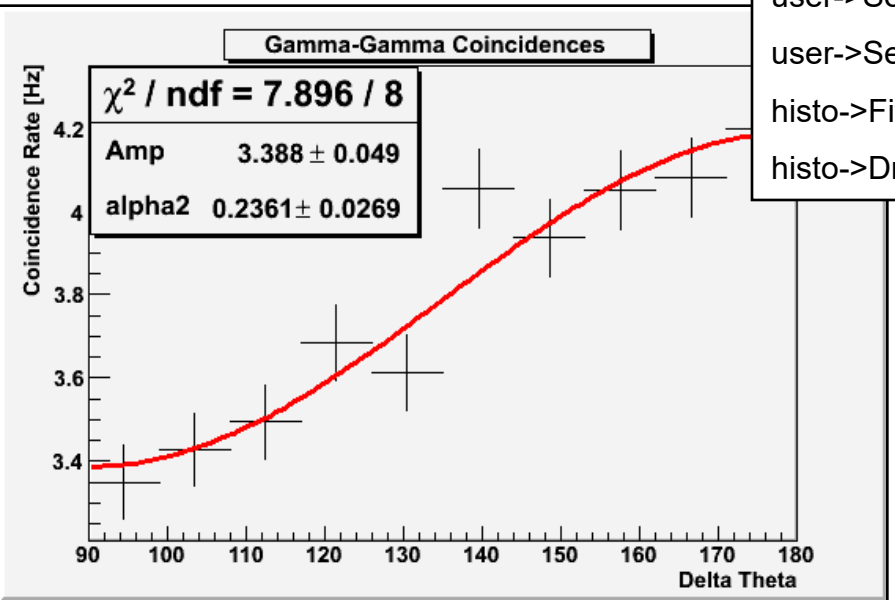


Specific from Phys 403

\Common\MyRoot\Pre-2017 scripts, macros and data\Phys403Expts

[root] .x histGammaGamma.cpp

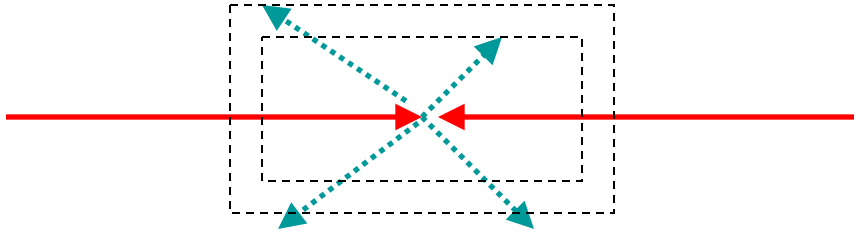
```
Stat_t yh[10]={3.35,3.428,3.495,3.683,3.613,4.054,3.935,4.05,4.08,4.197};
Stat_t yerr[10]={.088,.088,.089,.091,.090,.094,.093,.094,.094,.095};
TH1F* histo=new TH1F("histo","Gamma-Gamma Coincidences",10,90,180);
for (Int_t j = 0; j<10; j++)
{
        histo->SetBinContent(j+1,yh[j]);
        histo->SetBinError(j+1,yerr[j]);
}
histo->GetXaxis()->SetTitle("Delta Theta");
histo->GetYaxis()->SetTitle("Coincidence Rate [Hz]");
user=new  TF1("user","[0]*(1.+[1]*cos(x*3.14/180.)*cos(x*3.14/180))",90,180);
user->SetParNames("Amp","alpha2");
user->SetLineColor(2);  user->SetParameters(3.5,0.15);
histo->Fit("user","R");
histo->Draw();
```



(Extra slides for the muon experiment)

A word about how data is usually stored in nuclear and particle physics experiments

- Usually organized around "events" which are triggered



- Data is raw record of what happens in an "event"
  - Energy deposited in a detector
  - Hit position on wires
  - Time that a detector fires with respect to the trigger
- This information is used to make derived quantities and then to determine the history of what happened
  - Track of a particle
  - Type of a particle
  - Topology, or sequence of what happened
- Repeat for lots of events to arrive at physics histograms from the raw quantities or from the derived quantities

For our muon experiment

- **For each "trigger" (which is a guess that we just might have a muon stopping in our device), we record …**
  - **24 Raw pulse Areas (0 – 256 pC) for 24 photomultipliers, connected on east and west ends of 12 plastic scintillators**
  - **8 (brief) Time differences [0 – 100 ns] between the Trigger time and the time that something else happened. This is mostly a placeholder.**
  - **14 (longer) Time differences [0 – 8 μs] between the Trigger and the time that the scintillators may have "re-fired" owing to a decay positron or electron**
- **This is a like a vector recorded for each event with 46 entries. The data is a sequence of these vectors.**

When the data is Analyzed … "processed"

- **The Analyzer program write raw data to an NTUPLE, which ROOT can easily plot.**

- **The Analyzer can compute derived quantities and add them to the NTUPLE for each event**
  - **TDCSUM ← a handy way to get the times from any of the detectors that might have seen a decay**
  - **NLAYER ← where do we think the muon stopped?**
  - **UP ← the time of decay for positrons that hit scintillators above the muon stop**
  - **DOWN ← same but for scintillators below**
  - **Etc. ← you can (will) come up with others**
- **You can re-analyze raw data if you create new or better derived quantities. You don't have to rerun the experiment!**

Typical muon lifetime plots

- `h1iTDCSUM->GetXaxis()->SetLimits(0,8)`
  - Sets the 4000 channels to 0 – 8 us
- `h1iTDCSUM->Rebin(10)`
  - Rebins adjacent 10 channels into 1 bin
- `h1iTDCSUM->Fit("user","R")`
  - Fit to function defined by "user" over range "R"

```
root[]user=new TF1("user","[0]*exp(x/[1])+[2]",0.2,7.5)
```

Go to desktop for updates here…