

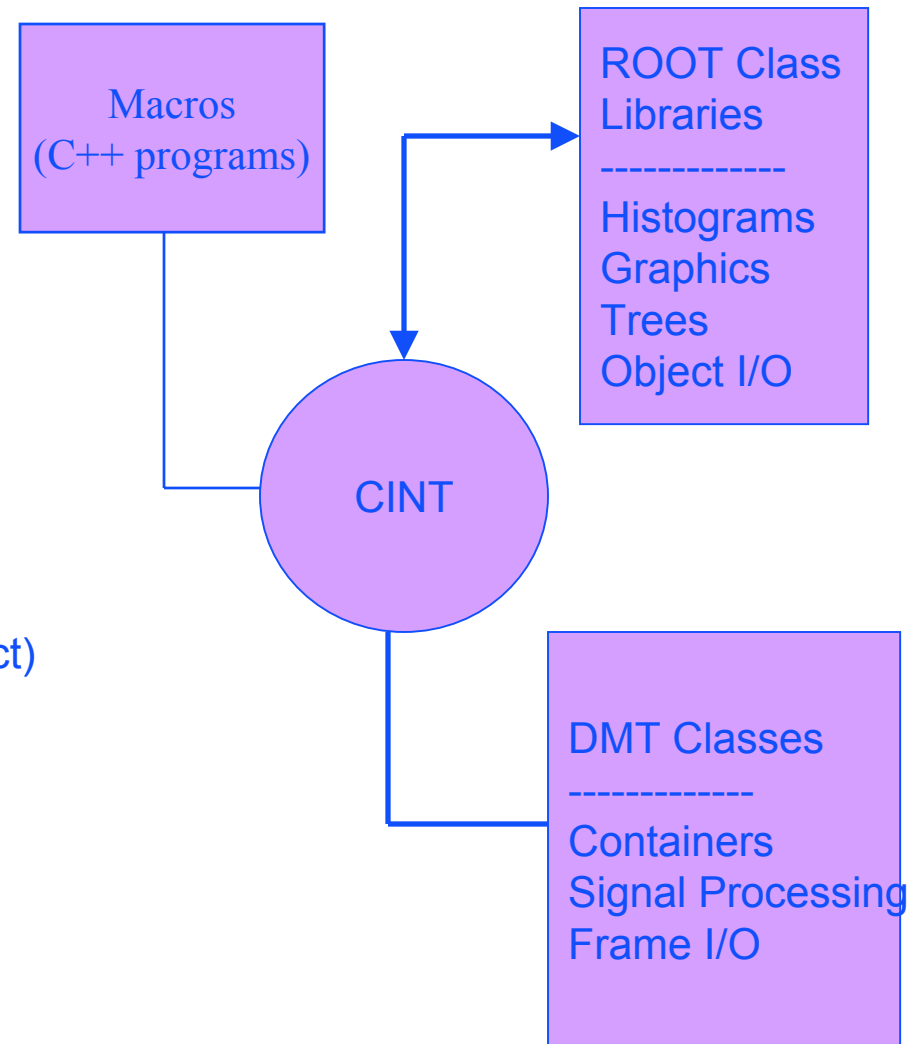


Interactive DMT with Root

John G. Zweizig
LIGO, Caltech

What is Root

- Scientific data analysis workstation
 - » Histogramming, fitting & plotting
 - » Data manipulation
 - » Graphics, GUI tools
- C++ Interpreter
 - » “Command” language is C++
 - » Macro facility -
 - » Class libraries provided with root
 - » Interface to user classes (dynamically loaded shared object)





Example Session: Hello World

```
[jzweizig@localhost ~]$ root
*****
*
*      W E L C O M E  t o  R O O T      *
*
*  version   4.00/06      7 July 2004  *
*
*  You are welcome to visit our web site *
*      http://root.cern.ch      *
*
*****
```

Run 'root'

Root says 'Hi'

```
FreeType Engine v2.1.3 used to render TrueType fonts.
Compiled for linux with thread support.
```

DMT startup macro takes over:
Loads DMT libraries.
And opens data accessor

```
CINT/ROOT C/C++ Interpreter version 5.15.138, May 23 2004
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
```

Root will execute any C command you enter.

```
LIGO Root initialization starting.
Loading libraries from $DMTHOME/rev_2.9.1/lib
Connecting default input accessor (In) to /online/LHO_Online
Root initialization complete
```

C++ statements can be entered as commands too.

```
root [0] printf("Hello world\n");
Hello world
root [1] cout << "Hello world" << endl;
Hello world
```

Expressions without ';' are evaluated and printed out.

```
root [2] 2+2
(const int)4
root [3] .q
```

Exit with ".q"

```
sandbox termination in progress...
All done. Ciao!
```



Documentation

- All DMT classes are documented under <http://www.ligo.caltech.edu/~jzweizig/dmt/DMTProject>
doc++ documentation for e.g. containers follow links
'DMT Library' -> 'Containers' -> 'Container classes'
- Within root (DMT and root classes), type
`.class <class-name>`
- For documentation of root classes, go to <http://root.cern.ch/>
and select 'Reference guide' link.



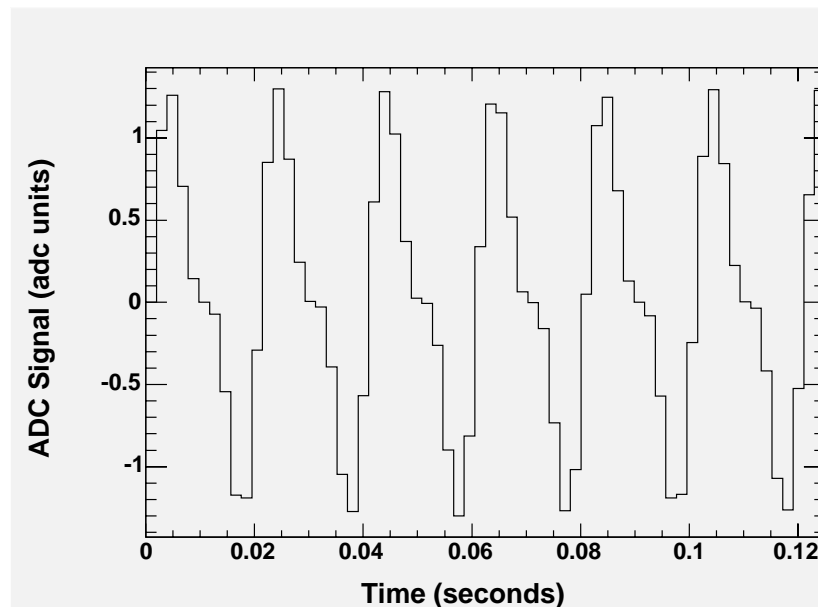
Setup Root & DMT

- On LHO GC machines
 - » Log on to e.g. `fortress.ligo-wa.caltech.edu`
 - » `touch .dmt-setup`
 - » Log off / log on.
 - » `root-setup`
- On DMT machines
 - » Log on to e.g. `sand.ligo-wa.caltech.edu`
 - » Add to your `.cshrc` file
 - `setenv DMTHOME /export/home/dmt`
 - `source $DMTHOME/pro/bin/setup`
 - » Log off / log on
 - » `root-setup`
- `Root-setup` prints out root and DMT versions, checks compatibility, links in root initialization macro

Data Containers

- **TSeries: time series**
 - » Contains arbitrary type data: short, int, float, double, complex
 - » Data in copy-on-write vectors
 - » Time metadata: t_0 , Δt
 - » Manipulation methods: `getData()`, `extract()`
 - » Arithmetic methods: `add()`, `sub()`, `mpy()`, `div()`, `dot()`, `cdot()`
 - » Arithmetic operators: `+`, `-`, `*`, `/`, `+=`, `-=`, `*=`, `/=`
- **FSeries: frequency series**
 - » Frequency & time metadata.
 - » Similar methods to `TSeries`.
- **FSpectrum: PSD**

```
TSeries ts(Time(0), Interval(1./512), 512, Sine(100));
TSeries ts2(Time(0), Interval(1./512), 512, Sine(50));
ts *= 0.5;
ts += ts2;
TSeries ts3(ts.extract(Time(0), 0.125));
TPlot(&ts3);
```





Data accessor (Dacc) class

- Function

- » Read multiple channels from a common time stride.
- » Channels may be time series or frequency series

- API

- » Specify input frame files:
 - `addFile(const char* frames), addList(const char* listfile);`
- » Specify channel names:
 - `addChannel(const char* chan, TSeries** pts=0, int decim8=2);`
 - `addFSeries(const char* chan, FSeries** pfs=0);`
- » Fill all channels with one stride of data
 - `int fillData(Interval dt);`
- » Find TSeries with channel data
 - `const TSeries* refData(const char* chan);`
- » Skip data to specified time
 - `seek(const Time& t);`



Root Macros

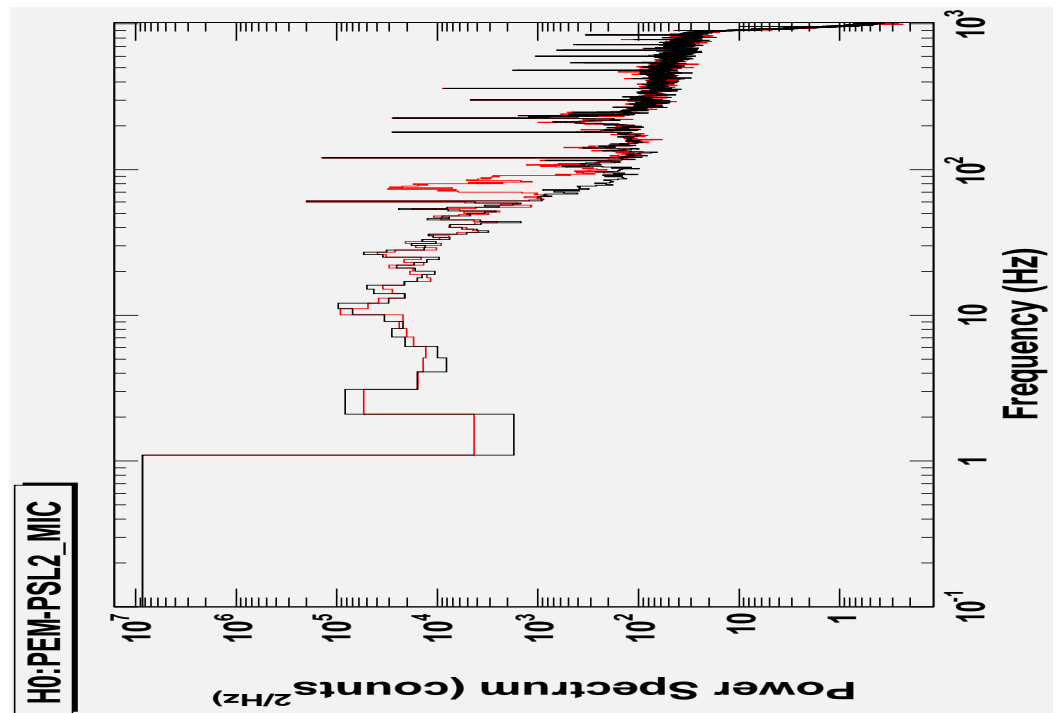
- Root macros are c++ code segments
 - » Unnamed macros enclosed in “{...}”
 - » Named macros have c++ function syntax, i.e., “*type fname(args) {...}*”
- Named functions may be loaded into CINT with “.L *macro.cc*” or with `gROOT->Load(“macro.cc”)` and executed with e.g. `fname()`
- Macros may be executed with “.x *macro.cc(args)*”
- DMT plotting macros
 - » `TPlot(const TSeries* ts);`
 - » `TPlot(const char* name);`
 - » `THist(const TSeries* ts);`
 - » `XYPlot(const TSeries&x, const TSeries& y);`
 - » `Spectrum(const FSpectrum& fs);`
 - » `ABSpectrum(const FSpectrum& a, const FSpectrum& b);`
 - » `pTrend(const char* chan, const char* fdir, const Time& t, Interval dt);`
- Loaded by DMT initialization macro



Example: Plotting Spectra

Compare 2 Spectra

```
{  
In.addFile(  
    "/samrds/S2/L1/LHO/H-RDS_R_L17320/*.gwf");  
In.seek(Time(732000704));  
TSeries* ts(0);  
In.addChannel("H0:PEM-PSL2_MIC", &ts);  
FSpectrum fs;  
for (int i=0; i<20; i++) {  
    In.fillData(1.0);  
    fs += FSpectrum(*ts);  
}  
In.fillData(20.0);  
FSpectrum fs2;  
for (int i=0; i<20; i++) {  
    In.fillData(1.0);  
    fs2 += FSpectrum(*ts);  
}  
ABSpectrum(fs, fs2);  
}
```





Signal Processing

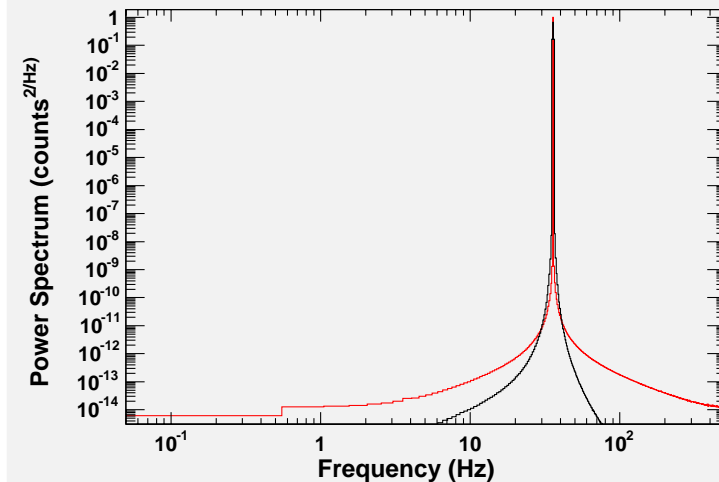
- Pipe: Time domain filter API
 - » Perform transformation on a TSeries producing a TSeries
 - » Base class for time domain filters, windows, decimators, et.
 - » Status methods:
 - `getStartTime()`, `getCurrentTime()`, `dataCheck(const TSeries&)`, `inUse()`
 - » Control methods
 - `reset()`
 - » Data processing methods
 - `apply(const TSeries&)`, `operator()(const TSeries&)`
- FDFilter: Frequency domain filter
 - » So far, only applies to FSpectrum



Signal Processing: Examples

- **Windowing:**

```
.L ABSpectrum.cc  
Hanning w;  
TSeries ts(Time(0),  
           Interval(1./1024), 1024,  
           Sine(35.5));  
FSpectrum fs(ts);  
FSpectrum fsw(w(ts));  
ABSpectrum(fs, fsw);
```



- **Filter design:**

```
FilterDesign fd("notch(60,100)",  
              1024.0);  
Pipe* p = fd.release();  
TSeries ts(Time(0),  
           Interval(1./1024), 2048,  
           Sine(60.0));  
TSeries tsf(p->apply(ts));  
TPlot(&tsf);
```

