Software Progress Report

Ole Hansen

Jefferson Lab

Hall A Data Analysis Workshop
December 3, 2008

Outline

Outline

Outline

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: `http://hallaweb.jlab.org/root/`

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: `http://hallaweb.jlab.org/root/`

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: `http://hallaweb.jlab.org/root/`

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: `http://hallaweb.jlab.org/root/`

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: `http://hallaweb.jlab.org/root/`

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: `http://hallaweb.jlab.org/root/`

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: http://hallaweb.jlab.org/root/

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: http://hallaweb.jlab.org/root/

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: http://hallaweb.jlab.org/root/

New in Podd 1.5

- Support for ROOT 5.18, 5.20, Fedora 8, 9, gcc 4.3, etc.
- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Intelligent split run handling
- Improved scaler code (helicity sorting)
- Speed improvements (output)
- Large file support (> 2 GB)
- Bugfixes (split runs, formulas, output, database, scalers)

Download: `http://hallaweb.jlab.org/root/`

Code Example: Database Request

Printed by Ole Hansen

Dec 11, 07 12:46 **MWDC.cxx** Page 1/1

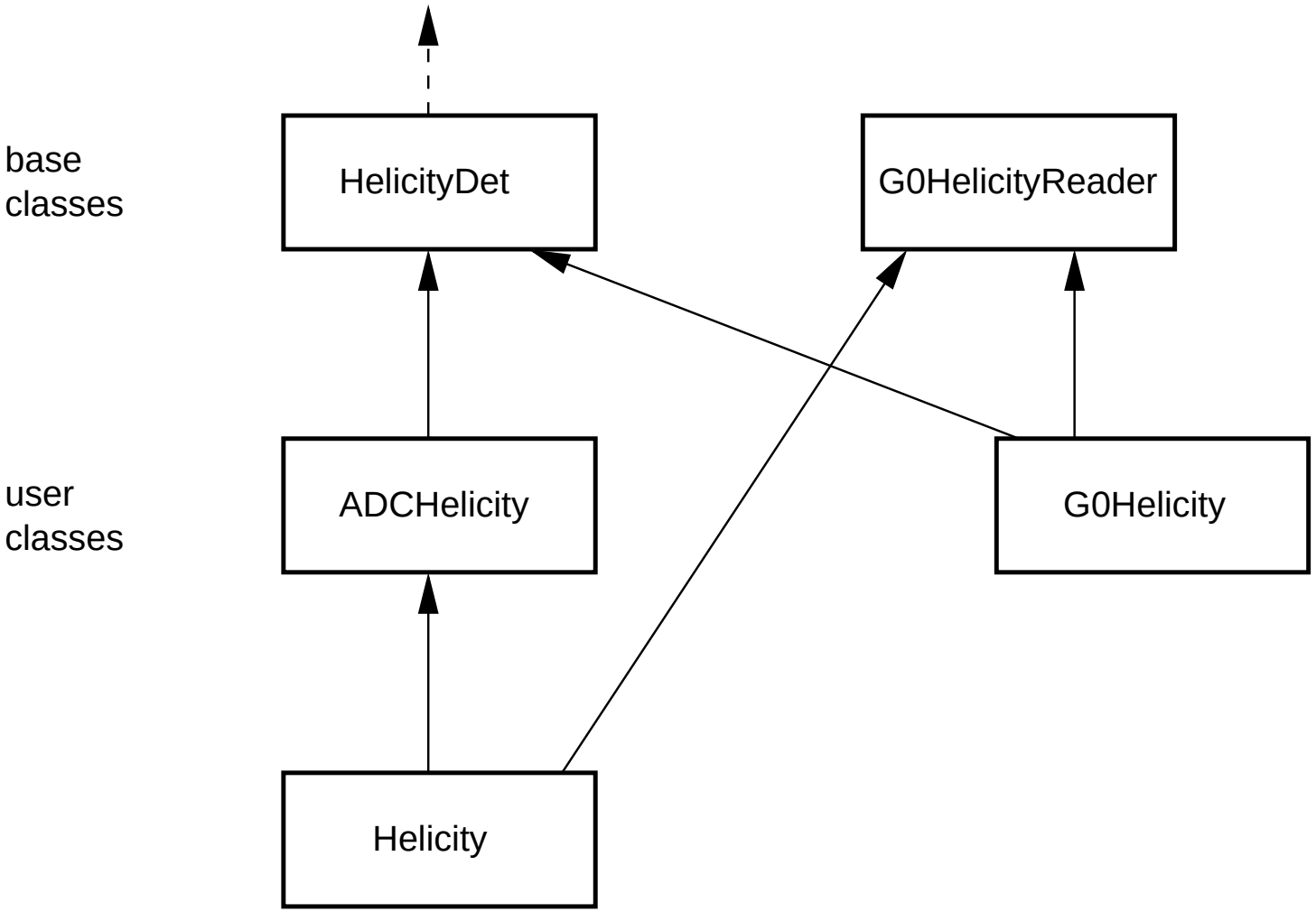
```
FILE* file = OpenFile( date );

vector<vector<Int_t> > *cmap = new vector<vector<Int_t> >;
string planeconfig;
Int_t time_cut = 1, pairs_only = 0, mc_data = 0;

DBRequest request[] = {
  { "planeconfig",  &planeconfig, kString },
  { "cratemap",     cmap,         kIntM,    6 },
  { "timecut",      &time_cut,    kInt,     0, 1 },
  { "pairsonly",    &pairs_only,  kInt,     0, 1 },
  { "MCdata",       &mc_data,     kInt,     0, 1 },
  { 0 }
};
Int_t err = LoadDB( file, date, request, fPrefix );
fclose(file);
```

Tuesday December 11, 2007 MWDC.cxx 1/1

New Helicity Class Hierarchy



Split Runs

Split Run Example

```
THaRun* r1 = new THaRun( "/daq/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/daq/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

Notes:

- Files must be in same directories, or in directories ending in `.../dataN/`
- Old methods for handling split runs still work
- Could be futher improved (single object for group of runs)

Split Runs

Split Run Example

```
THaRun* r1 = new THaRun( "/daq/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/daq/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

Notes:

- Files must be in same directories, or in directories ending in `.../dataN/`
- Old methods for handling split runs still work
- Could be futher improved (single object for group of runs)

Split Runs

Split Run Example

```
THaRun* r1 = new THaRun( "/daq/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/daq/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

Notes:

- Files must be in same directories, or in directories ending in `.../dataN/`
- Old methods for handling split runs still work
- Could be futher improved (single object for group of runs)

Split Runs

Split Run Example

```
THaRun* r1 = new THaRun( "/daq/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/daq/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

Notes:

- Files must be in same directories, or in directories ending in `.../dataN/`
- Old methods for handling split runs still work
- Could be futher improved (single object for group of runs)

Feature Wishlist for Podd 1.6

- Support for ROOT 5.22, Fedora 10, etc. (to do)
- "Event Type Handler" plug-ins (in progress)
- Consistent use of `LoadDB` to read database (to do)
- SQL backend for `LoadDB` (in progress → Tel Aviv group)
- Replace `TDatime` with `TTimeStamp` to handle timezones
- Documentation, User's Guide (maybe)

Of course, there is much more . . . but let's stay realistic

Feature Wishlist for Podd 1.6

- Support for ROOT 5.22, Fedora 10, etc. (to do)
- "Event Type Handler" plug-ins (in progress)
- Consistent use of `LoadDB` to read database (to do)
- SQL backend for `LoadDB` (in progress → Tel Aviv group)
- Replace `TDatime` with `TTimeStamp` to handle timezones
- Documentation, User's Guide (maybe)

Of course, there is much more . . . but let's stay realistic

Feature Wishlist for Podd 1.6

- Support for ROOT 5.22, Fedora 10, etc. (to do)
- "Event Type Handler" plug-ins (in progress)
- Consistent use of `LoadDB` to read database (to do)
- SQL backend for `LoadDB` (in progress → Tel Aviv group)
- Replace `TDatime` with `TTimeStamp` to handle timezones
- Documentation, User's Guide (maybe)

Of course, there is much more . . . but let's stay realistic

Feature Wishlist for Podd 1.6

- Support for ROOT 5.22, Fedora 10, etc. (to do)
- "Event Type Handler" plug-ins (in progress)
- Consistent use of `LoadDB` to read database (to do)
- SQL backend for `LoadDB` (in progress → Tel Aviv group)
- Replace `TDatime` with `TTimeStamp` to handle timezones
- Documentation, User's Guide (maybe)

Of course, there is much more . . . but let's stay realistic

Feature Wishlist for Podd 1.6

- Support for ROOT 5.22, Fedora 10, etc. (to do)
- "Event Type Handler" plug-ins (in progress)
- Consistent use of `LoadDB` to read database (to do)
- SQL backend for `LoadDB` (in progress → Tel Aviv group)
- Replace `TDatime` with `TTimeStamp` to handle timezones
- Documentation, User's Guide (maybe)

Of course, there is much more . . . but let's stay realistic

Feature Wishlist for Podd 1.6

- Support for ROOT 5.22, Fedora 10, etc. (to do)
- "Event Type Handler" plug-ins (in progress)
- Consistent use of `LoadDB` to read database (to do)
- SQL backend for `LoadDB` (in progress → Tel Aviv group)
- Replace `TDatime` with `TTimeStamp` to handle timezones
- Documentation, User's Guide (maybe)

Of course, there is much more . . . but let's stay realistic

Feature Wishlist for Podd 1.6

- Support for ROOT 5.22, Fedora 10, etc. (to do)
- "Event Type Handler" plug-ins (in progress)
- Consistent use of `LoadDB` to read database (to do)
- SQL backend for `LoadDB` (in progress → Tel Aviv group)
- Replace `TDatime` with `TTimeStamp` to handle timezones
- Documentation, User's Guide (maybe)

Of course, there is much more . . . but let's stay realistic

Event Type Handler Plugins

- Problem: DAQ upgrades → new event types
- Solution: Plug-in modules for `THaAnalyzer`, replace `PhysicsAnalysis()`, `ScalerAnalysis()`, etc.
- Modules for standard event types (physics, scalers, EPICS) are provided in core library

Event Type Handler base class

```
class THaEventTypeHandler : public TObject {

public:
  THaEventTypeHandler();
  virtual ~THaEventTypeHandler();

  virtual Int_t  Analyze( const THaEvData& evdata, Int_t status ) = 0;
  ...
};
```

Event Type Handler Plugins

- Problem: DAQ upgrades → new event types
- Solution: Plug-in modules for `THaAnalyzer`, replace `PhysicsAnalysis()`, `ScalerAnalysis()`, etc.
- Modules for standard event types (physics, scalers, EPICS) are provided in core library

Event Type Handler base class

```
class THaEventTypeHandler : public TObject {

public:
  THaEventTypeHandler();
  virtual ~THaEventTypeHandler();

  virtual Int_t  Analyze( const THaEvData& evdata, Int_t status ) = 0;
  ...
};
```

Event Type Handler Plugins

- Problem: DAQ upgrades → new event types
- Solution: Plug-in modules for `THaAnalyzer`, replace `PhysicsAnalysis()`, `ScalerAnalysis()`, etc.
- Modules for standard event types (physics, scalers, EPICS) are provided in core library

Event Type Handler base class

```
class THaEventTypeHandler : public TObject {

public:
  THaEventTypeHandler();
  virtual ~THaEventTypeHandler();

  virtual Int_t  Analyze( const THaEvData& evdata, Int_t status ) = 0;
  ...
};
```

Event Type Handler Plugins

- Problem: DAQ upgrades → new event types
- Solution: Plug-in modules for `THaAnalyzer`, replace `PhysicsAnalysis()`, `ScalerAnalysis()`, etc.
- Modules for standard event types (physics, scalers, EPICS) are provided in core library

Event Type Handler base class

```
class THaEventTypeHandler : public TObject {

public:
  THaEventTypeHandler();
  virtual ~THaEventTypeHandler();

  virtual Int_t  Analyze( const THaEvData& evdata, Int_t status ) = 0;
  ...
};
```

BB Tracking: Progress since June 2008

- Some minor bugs/inefficiencies identified → to do
- Extensive calibration work (→ Sergey's talk)
- Summer student (CS major) developed advanced clustering algorithm (2D "de-cloning")
  - Faster than original code for very noisy E04-007 data
  - Status: preliminary

BB Tracking: Progress since June 2008

- Some minor bugs/inefficiencies identified → to do
- Extensive calibration work (→ Sergey's talk)
- Summer student (CS major) developed advanced clustering algorithm (2D "de-cloning")
  - Faster than original code for very noisy E04-007 data
  - Status: preliminary

BB Tracking: Progress since June 2008

- Some minor bugs/inefficiencies identified → to do
- Extensive calibration work (→ Sergey's talk)
- Summer student (CS major) developed advanced clustering algorithm (2D "de-cloning")
  - Faster than original code for very noisy E04-007 data
  - Status: preliminary

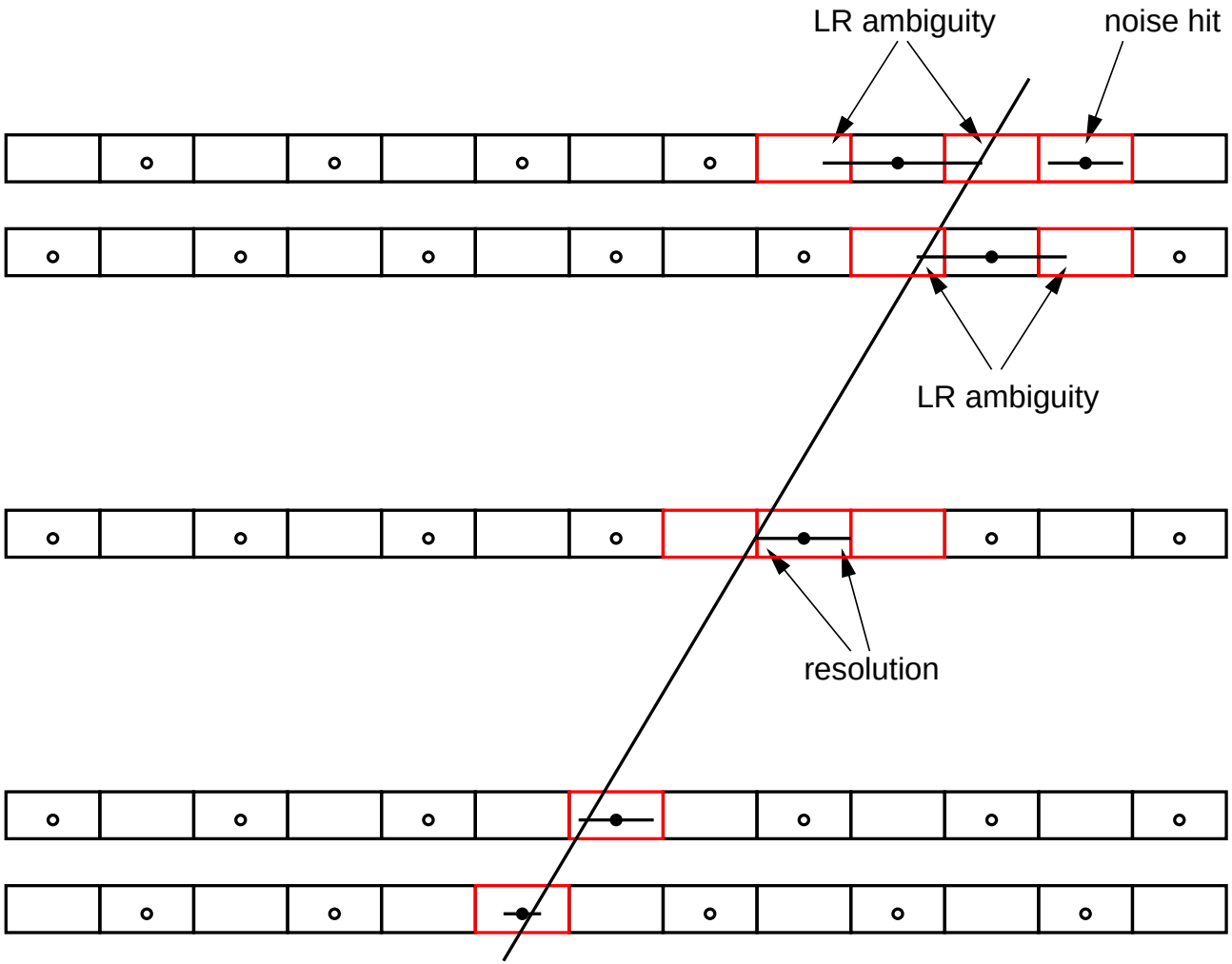
BB Tracking: Progress since June 2008

- Some minor bugs/inefficiencies identified → to do
- Extensive calibration work (→ Sergey's talk)
- Summer student (CS major) developed advanced clustering algorithm (2D "de-cloning")
  - Faster than original code for very noisy E04-007 data
  - Status: preliminary

BB Tracking: Progress since June 2008

- Some minor bugs/inefficiencies identified → to do
- Extensive calibration work (→ Sergey's talk)
- Summer student (CS major) developed advanced clustering algorithm (2D "de-cloning")
  - Faster than original code for very noisy E04-007 data
  - Status: preliminary

Pattern Cloning (multiple patterns for single track)



To Do List

- Finish optimized clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Add ability to pre-cut using shower or scintillator (started)
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")
- Documentation!

To Do List

- Finish optimized clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Add ability to pre-cut using shower or scintillator (started)
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")
- Documentation!

To Do List

- Finish optimized clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Add ability to pre-cut using shower or scintillator (started)
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")
- Documentation!

To Do List

- Finish optimized clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Add ability to pre-cut using shower or scintillator (started)
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")
- Documentation!

To Do List

- Finish optimized clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Add ability to pre-cut using shower or scintillator (started)
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")
- Documentation!

To Do List

- Finish optimized clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Add ability to pre-cut using shower or scintillator (started)
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")
- Documentation!

To Do List

- Finish optimized clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Add ability to pre-cut using shower or scintillator (started)
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")
- Documentation!