Bee and Wire Cell
Status and Larsoft Integration Plans

Brett Viren
(for the Wire-Cell effort)

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

DUNE Collaboration
21 May 2016

Outline

Bee

Wire Cell Prototype

Wire Cell Toolkit

Larsoft Integration

Bee in a nutshell

A web application for LArTPC event visualization and reconstruction

- Interactive 3D frontend through WebGL and JavaScript.
- Event file catalog through Django backend.
- Supports all major browsers including mobile devices.

Feature highlights:

- Overlay results of multiple algorithms.
- MC truth overlay and particle hierarchy browser.
- 2D projections, sliced animation, keyboard navigations.
- Multiple geometries (MicroBooNE, 35t, protoDUNE, ...).
- User file upload through drag&drop.

Seeing is Bee-lieving

http://www.phy.bnl.gov/wire-cell/bee/

Bee Development and Plans

Recent development highlights:

- Significantly **improved memory usage**, which improved **frame rate** with many ($\sim$ 1M) particles.
- Utilized *Web Worker* for non-blocking UI with required heavy **client-side computations** (eg, dead-region calculations).
- Added ray-casting for **object picking**.

Near-term plan:

- Modularize code through front-end build system.
  - NodeJS modules + browserify
- Move from ES5 to ES6 for enhanced language features.
- New UI design for the next-generation of Bee.
- Further improve memory usage.

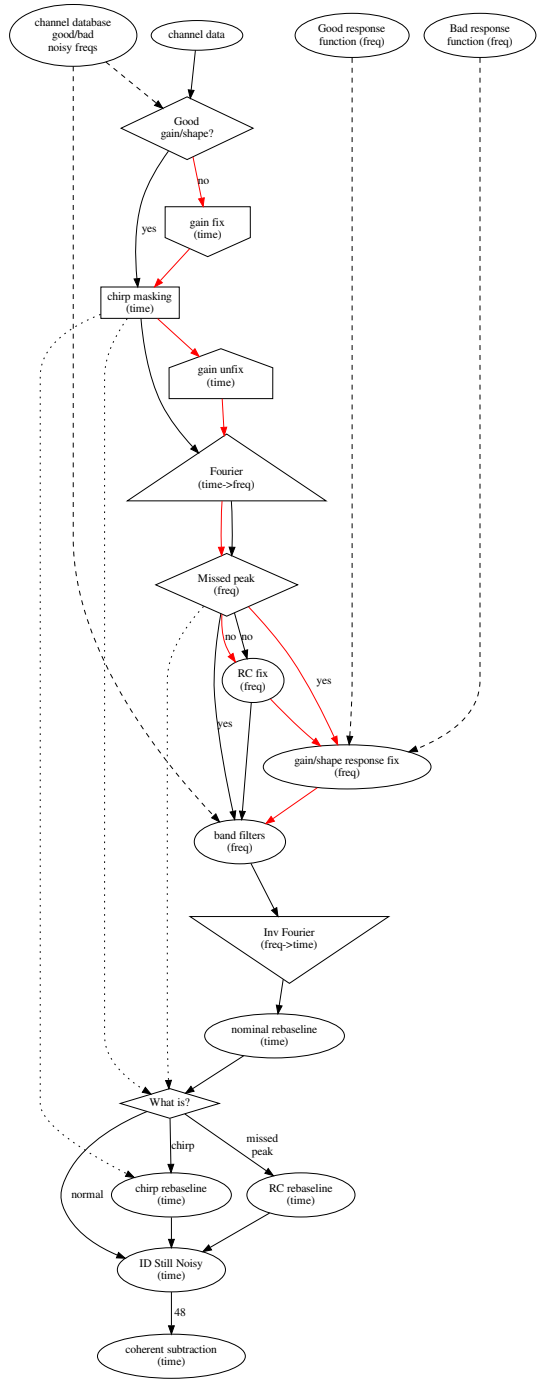
Outline

Prototype Noise Removal

- Currently focused on MicroBooNE.
- Port from prototype to toolkit and
- Larsoft integration in-progress.

1. Chirp detection.
2. Fourier transform.
3. Apply RC+RC correction (if signal is intact).
4. Fix incorrect gain/shape settings.
5. Apply narrow band filters on harmonic noise.
6. Inverse Fourier transform.
7. Rebaseline.
8. Coherent subtraction (48 channels).

Achieves excellent results!



Outline

Wire Cell Toolkit Overview

- Gestalt: **port the prototype** to a sustainable, integratable, support multi-{developer,processing,architecture} toolkit.
- A set of shared libraries offering a "white box" of functionality with **multiple levels of use**:

  low Simple functions taking basic data types.
  mid Concrete functor classes aggregating functions.
  high Pure, abstract interfaces, factory method construction, configuration mechanism.
  DFP Data-Flow Programming graph construction.
  CLI Single, command-line program (`wire-cell`) to aggregate DFP nodes, configure and run them.

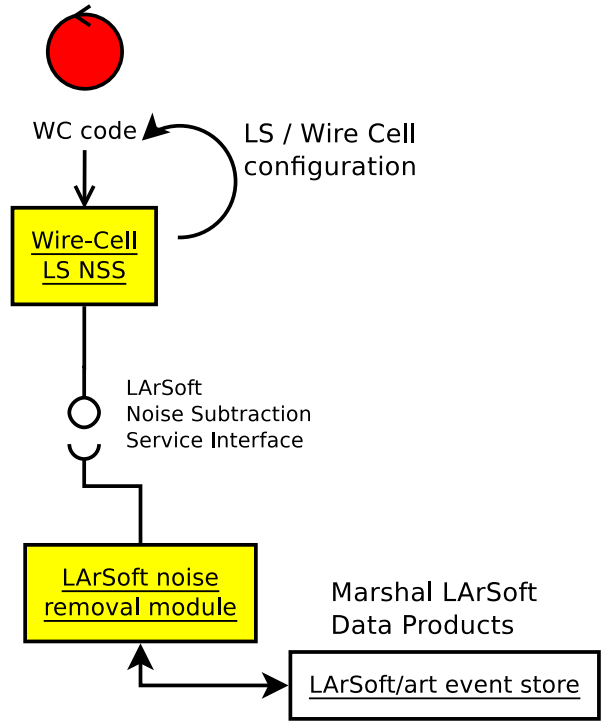
- Focus on development details at all levels and cleanly integrate into LArSoft at the right levels.

Outline

Initial Integration Target - Noise Subtraction

LArSoft/Wire-Cell **Noise Subtraction Service**:

- Links to Wire-Cell libraries.
- Adapts LS NSS interface to WC implementation.
- Provides LS→WC configuration translations.
- (more on NSS next)



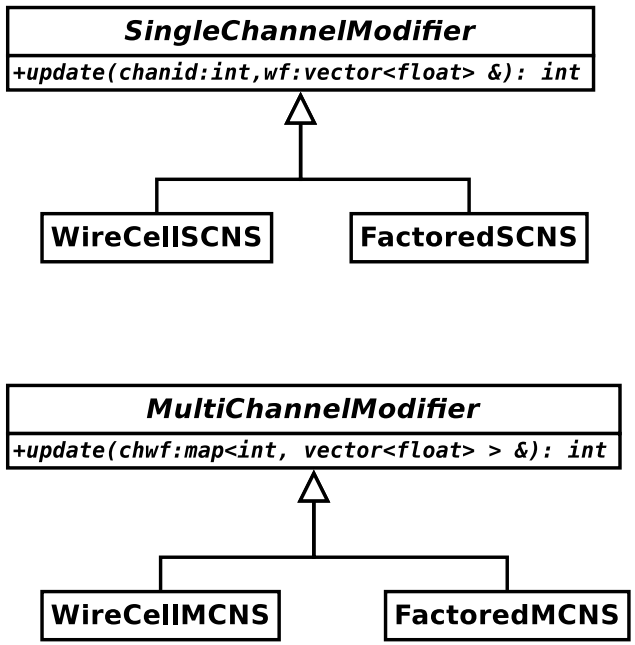
This will be the model for future LArSoft↔Wire-Cell integration.

- Define or adopt existing service interface and provide implementation.

→ high-CPU WC imaging procedures may require a different, more distributed model to mitigate high-RAM usage of LS (and WC).

Noise Subtraction Service Interfaces

- Feature request #11750 by David Adams
  - Driven by wanting to bundle current LS NS.
  - WC integration will follow suit.
  - Separate single- and multi-channel interface.
- A waveform is simply: `vector<float>`.
- No-copy, mutate-in-place semantics for efficiency and chaining of multiple NSS.
- Class names and exact factoring of implementations t.b.d.
  - "`WireCell`" = calls code in WCT,
  - "`Factored`" = factoring what is in LArSoft now.



Going further:

- Move common parts to cross-experiment `larsoft-*` packages.
- Develop finer-grained interfaces exposing noise subtraction internals.

→ Care needed as real-world subtraction procedures are highly interconnected by out-of-band metadata (see graph above)

LArSoft/Wire-Cell Integration To-Do

LS/WC integration source code:

- new package: `larsoft-wirecell` (follow PANDORA's lead)
- this is experiment-agnostic code (exp. specifics in config layer)
- source code to live in Redmine/git under LArSoft banner
  → Wire-Cell code itself continues in GitHub (prototype/toolkit).

Code builds:

- WCT has limited, external requirements:
  - Boost and Eigen needed by core libraries.
  - No ROOT dependency in core, only I/O and test layers.
- Easy "`configure`"-like build system. Supports Linux and Mac.
- Will need help to regularly produce UPS products at FNAL:

`wire-cell-*` the needed subset of Wire-Cell packages
`larsoft-wirecell` the integration package, itself.