EIC software

Alexander Kiselev

NPPS Group Meeting  August,23 2019
Contents of this talk

- Fast simulation tool: eic-smear
- Software frameworks
  - GEMC
  - fun4all
  - EicRoot
  - Argonne EIC software initiative
- PID consortium GEANT4 software (one slide)
- Near-term future trend(s)

Materials taken from presentations of T.Burton, M.Ungaro, E.Sichtermann, J.Repond, D.Lawrence, D.Romanov, Y.Furletova & others
eic-smear
by Tom Burton (BNL TF group)
Overview

- C++ code, runs in ROOT
- Build with `configure/Make` or `CMake`
- `libeicsmear.so` to load in ROOT

## Tree code:
Build ROOT tree containing events

- Large number of EIC Monte Carlo generators with standard ASCII format

### MC generator
- ASCII output

### Smearer:
Perform fast detector smearing

- PEPSI
- Milou
- LEPTO
- Djangoh
- DPMJet
- gmc_trans
“Smearer” defines some element of performance + acceptance

“Detector”

- Built-in standard smearers provided with eic-smear
- Users can define own smearers using inheritance

NOT a “physical detector”: represents the overall performance in measuring a quantity.

- Apply all smearers to an MC event
- Yield smeared event
- Optionally recalculate derived values e.g. $x, Q^2$
How to use it

• Write a ROOT script:

```cpp
Smear::Detector createDetector() {
    // Resolution in momentum, sigma(P).
    // sigma(P) = 0.4%P + 0.3%P^2.
    Smear::Device tracking("P", "0.004 * P + 0.003 * pow(P, 2)");
    // Add devices to a Detector.
    Smear::Detector detector;
    detector.AddDevice(tracking);
    return detector;
}
```

• Smear your ROOT tree:

```cpp
root[0] SmearTree(createDetector(), "mc.root", "smeared.root");
```

• “Standard” detector descriptions (like STAR or BeAST) exist

See K.Kauder: talk at the EIC software meeting 07/10/2019
GEMC
by Maurizio Ungaro (JLab)
GEant4 Monte Carlo Architecture

- Application independent geometry/digitization/fields: definitions stored in databases
- Realistic hits treatment: electronic time window, voltage versus time signals.
- Sensitive attributes assigned at run time: real calibration, survey tilts and displacements.
- Plugins for generator formats (LUND, BEAGLE, easy expansion)
- Plugins for output formats (TXT, CODA, JSON, easy expansion)
- Realistic signal treatment allows for background rate studies, including pile-up effects

- Application for detector simulations based on Geant4
- Macro language for detector design
- Various geometry definitions: GEMC, gdml, CAD
- Data card (XML) to steer application, all Geant4 macro commands supported by design
Geometry

Input: Native, CAD, GDML. Arbitrary hierarchy, can be mixed and matched. Materials, sensitivity assigned at run-time.

Experiments using the GEMC Framework: CLAS12 (Hall-B), EIC Beamline and detectors, HPS, Solid
Digitization, Output

- Single ADC/TDC over electronic time window.
- Voltage vs time signal.
- FADC output (4ns intervals or integral mode)
- Automatic true information
- All g4 steps in the output

Run options: tilts, displacement, calibration, inefficiencies

Geometry and Physics Modeling

Transport Calculation
- Energy loss
- Secondaries

Digitization
- Hit definition
- Sensitivity

Output
- Bank def's
- File format

Results

> BST

> True Step by Step infos (101, 0)
  - Edep (101, 1)
  - Pid (101, 2)
  - positions (101, 3)

> Dgtz Step by Step infos (102, 0)
  - ADCL (102, 1)
  - ADCR (102, 2)

> True Integrated infos (103, 0)
  - Edep (103, 1)
  - Pid (103, 2)
  - positions (103, 3)

> Dgtz Integrated infos (104, 0)
  - ADCL (104, 1)
  - ADCR (104, 2)

> Voltage as a function of time (105, 0)
  - Identifier (105, 1)
  - Time (105, 2)
  - Voltage (105, 3)

> Trigger Bank (106, 0)
  - Identifier (106, 1)
  - Time (106, 2)
  - Voltage (106, 3)
Graphical Interface

- Generator
- Event time window
- Background beams
- Camera views slices.
- Axis, Scale, Show field.

- Volumes hierarchies and properties
- Output to GDML

- Geant4 OpenGL View for the whole detector.
- Can inspect and open a view on single volumes.

- Graphical analysis of steps in a hit.
- Can choose variable to display.
fun4all

by Chris Pinkenburg (BNL)

See talk at the EIC software meeting 07/10/2019
EicRoot

by AK (BNL)
EicRoot framework building blocks

- Interface to GEANT, ROOT, ...

PandaRoot
- “Ideal” track finder,
- Interface to GenFit
- ...

EicRoot
- MC generated evts import
- Fast smearing codes

FairBase
- Fast smearing codes

FopiRoot
- TPC R&D stuff, ...

CbmRoot
- RICH stuff

eic-smear
- IR design configuration

solenoid modeling

-> basically a yet another FairRoot software clone
End user view

- No executable (steering through ROOT macro scripts)

- ROOT files for analysis available after each step
- C++ class structure is well defined at each I/O stage

See AK: talk at the EIC software meeting 07/10/2019
Example case studies

Tracker momentum resolution

12 GeV pions: Hcal vs EmCal
slope ~1.20

Calorimeter design optimization

Neutron fluence

DIS electron reconstruction
Current modeling work

- Possible central tracker configurations (alternatives to a TPC)

- Temple University: $\mu$RWELL $\mu$TPC barrels

- LBNL: tapered all-silicon tracker

- $e^-, B = 1.5 \, T$
  $|\eta| < 1$
  $p = 20 \, GeV$

- $\frac{\delta p}{p} \sim 2\%$
Current modeling work

University of Birmingham: vertex tracker optimization

University of Birmingham: vertex tracker optimization

100 x 10 GeV

BNL: far forward acceptance

Florida Tech: tracking resolution in the RICH volume

Inner forward GEMs

Outer forward GEMs

Si tracker

TPC

RICH

Roman Pots

B0 spectrometer

first quad aperture & beam pipe

Events

0 50 100 150 200 250 300 350 400 450

Momentum [GeV/c]

Transverse pointing resolution [µm]

0 20 40 60 80 100 120 140 160

10x10 µm²

20x20 µm²

30x30 µm²

40x40 µm²
PID Consortium software
Mostly RICH & ToF applications

- All are custom GEANT4 codes
Argonne EIC software
Full simulation and reconstruction chain

- **Event generation**
  - Produce the simulation input events

- **Detector simulation**
  - Particle transport through detectors

- **Digitization**
  - Turn energy deposits in active media into detector hits

- **Reconstruction of**
  - Event vertex, charged tracks, Particle Flow Objects (PFO)

- **Perform analysis**
  - Collection of benchmark analyses
Argonne Software: Overview

Legacy chain

Adaptation of the SiD (ILC) simulation and reconstruction software chain

Major parts

- SLIC (wrapper around GEANT4)
- LCSIM (digitization and event reconstruction)
- slicPandora (PFA reconstruction)

Visualization with JAS4pp

Limitations

- Only SiD subdetectors (e.g. no RICH)
- Geometry description not centralized
- Geometry constrained to be symmetric
- Some parts difficult to maintain

Full chain

Available
Studies of $F_2$ reconstruction, timing...

Evolution chain

Evolved from the legacy chain

Geometry interface

DD4HEP

Features

- Fully maintainable
- Geometry obtained from single source
- Geometry can be parametrized
- Geometry not constrained to be symmetric
- New subsystems can be easily implemented

Still working on

- Realistic digitization
- Generic tracking
- PFA reconstruction
- Visualization
Nuclear Physics Detector Library (NPDet)

Collection of parametrized detectors which can be developed into full concepts
ProIO Key Concepts

- Language-neutral I/O for streaming events
- Thin, native containers for protobuf messages, simply adding the concept of an event
- protobuf + event structure = ProIO
- Serialized output can be accessed effectively in archival file, or in a stream
Grand unification, 
yet another try
by Dmitry Romanov, David Lawrence, Yulia Furletova & others (JLAB)
Key ingredients

- (Docker) containers
- Jupyter notebooks
- JANA2 software framework
- g4e GEANT-based EIC detector sandbox
Software distribution model(s)

Efforts required axis

Cloud
Containers
A PC farm
Conda
Workstation
Compilation
EJPM

Main focus at present
A side note: EIC Docker containers

-> introduced in Aug,2017; went public by EICUG meeting in Nov,2017

Clear benefits for EIC user community

- Allow EIC users to run the same software under standardized environment on any Linux, Mac OS or Windows machine, eventually including GRID sites, commercial cloud systems, and HPC resources
- Provide consistency between software generated at different facilities
- Make it easier for new users to start working on the physics program and detector design for the EIC, by minimizing the pain of “installation overhead”
Core functionality overview

Database with various MC samples

BNL & Jlab effort on Fast detector prototyping

ejana = EIC Jana
Community reference reconstruction

---

Generators Database

Fast simulation

- Eic smear
- Geant4 Fast mode

Full simulation

- g4e
  - JLEIC/eRHIC in Geant4

Reconstruction & analysis

- ejana – EIC JANA(2)

(1) MC events
(2) Digitized hits + magnetic field + material distribution
(3) Reconstructed events

-> user access (with graphics) either directly or through SSH or Web interface

See D. Romanov: talk at the EIC software meeting 07/10/2019
JupyterLab Web interface

Wiki: Jupyter Notebook is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format.

- Cloud based collaborative workspace
- The medium for studies, reports, analysis
- The bridge between modern Data Science and traditional Nuclear Physics methods
JupyterLab Web interface

- Self-documenting
- Appealing & modern ...
- ... yet not really mandatory to get access to the core (container) functionality
Community reference reconstruction

\[ e^{\text{JANA}} \] - stands for EIC JANA

- Basic reconstruction
- Physics analysis
- Users detector codebase integration

Reconstruction

- Tracking - Genfit
- Vertex finding – Rave
- Physical analysis:
  - ROOT C++ or
  - Python data science tools (Jupyter, Seaborn, Pandas, etc)

Any existing C++ (or even others) code can be:
- compiled as JANA plugin
- run parallelized in eJANA
- accessed by other plugins
Jana(2) software framework

- Provide mechanism for many physicists to contribute reconstruction codes to the “shared pool”
- Implement multi-threading efficiently & external to the contributed codes
- Provide common mechanisms for accessing job configuration, calibrations, etc.
Data on demand = Don’t do it unless you need it

Stock = Don’t do it twice

Conservation of CPU cycles!
Jana(2): multi-threading

- Each thread has a complete set of factories making it capable of completely reconstructing a single event.

- Factories only work with other factories in the same thread eliminating the need for expensive mutex locking within the factories.

- All events are seen by all Event Processors (multiple processors can exist in a program).
Jana(2): event reconstruction scheme

Framework has a layer that directs object requests to the factory that completes it.

Multiple algorithms (factories) may exist in the same program that produce the same type of data objects.

This allows the framework to easily redirect requests to alternate algorithms specified by the user at run time.

See D. Lawrence: talk at the EIC software meeting 05/21/2019
GEANT 4 EIC

• The codename **g4e**: Geant 4 EIC
• **Beta** stage
• $\sqrt{s}$ 100 GeV JLEIC design is implemented
• Imports CAD, accelerator group data
• Exports final Geometry in various formats
• Plain flattened analysis ready ROOT files

• Particle gun, Pythia6, Pythia8, Herwig, BeAGLE
• Work in progress: sensitive volumes, digitization, configuration

$\rightarrow$ a candidate for a shared EIC detector sandbox software

See Y.Furletova: talk at the EIC software meeting 07/10/2019