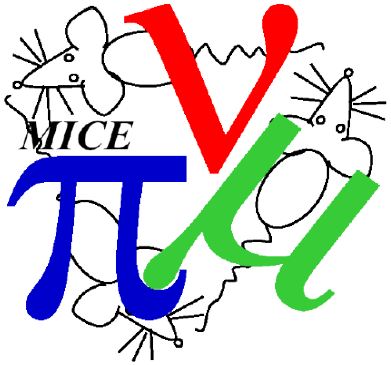
MAUS – Online Analysis



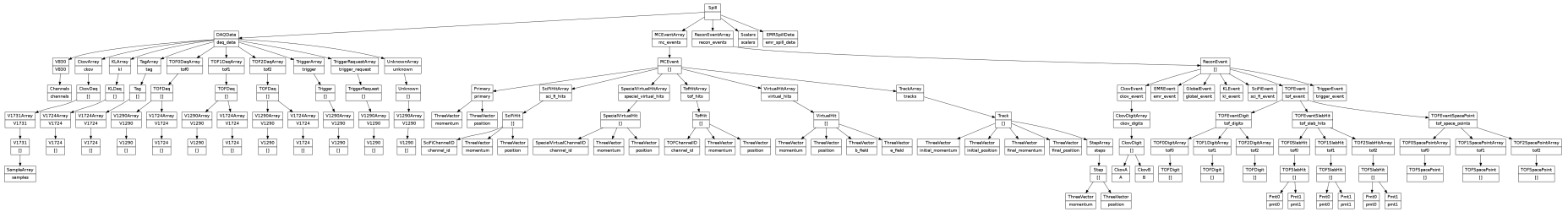
Chris Rogers,
ASTeC,
Rutherford Appleton Laboratory

Overview

- Few updates on progress in MAUS
- Summary of Monte Carlo requirements for Step IV
- Few thoughts on how we do “online analysis”

MAUS Data Structure

- MAUS is now writing to ROOT files natively
  - JSON is still maintained as an optional output format
  - Data structure is pretty complicated



- Documentation for usage and API:

  http://micewww.pp.rl.ac.uk/maus/MAUS_latest_version/maus_user_guide

- (or link from MAUS user page)

MAUS Geometry

- Starting to get some “official” geometry in MAUS
  - March run geometry based on survey data
  - May run geometry based on engineering data
    - Measured position of single station
  - Needs to be manually downloaded from CDB
    - $MAUS_ROOT_DIR/bin/utilities/download_geometry.py
    - Would like this to be automatic sometime (API in issue #839)
  - Documentation

    http://micewww.pp.rl.ac.uk/maus/MAUS_latest_version/maus_user_guide/node5.html
  - Some tweaking required to get the beam aligned correctly into the geometry etc

What we have currently:

- Well documented set up to pull in Monte Carlo configuration if it gets provided by the engineers
- Well documented set up to configure whatever fields and geometry people need
- Poorly documented module to generate multivariate gaussian beams with a reasonably flexible parameter set
- Well documented module to track and write out Monte Carlo data
- Poorly documented and possibly buggy tracker digitisation
- Elementary but probably reasonable TOF digitisation - but no user documentation

What I think we need in order to perform an end-to-end Monte Carlo of Step IV:

1. Step IV MICE "As-designed" (e.g. legacy/FILES/.../Step4.dat) set up needs to be checked and fixed - are the coil geometries correct? Can we get a reasonable beam through? What are the design absorber settings? Are they the same as what we have in Step4.dat
2. Infrastructure - need to have central store of datacards, field maps, geometry, etc
3. Functioning digitisation for the tracker
4. Trigger digitisation (what form should this take)?
5. KL, EMR, Ckov digitisation
6. Scalars digitisation
7. "As-built geometry" and CDB interface needs to be smoother (operational issue to do with getting geometries into CDB)

What I think we need to improve the "user experience":

1. Quick start guide + examples (with some automated testing)
    1. Track a beam through and generate beta function/emittance plots
    2. Track a beam through and generate reconstructed data -> residuals/emittance plot at the Tracker Reference Plane
    3. Anything else?
2. ~~Documentation maus_user_guide.pdf should be available to download/htmlarised from the wiki~~
3. MICEModules.pdf needs updating and integrating with maus_user_guide.pdf
4. Documentation - some stuff is in doxygen only, it needs to be in the user guide as well.
5. Optics calculation needs to be tidied
6. Python interface for extracting the field values
7. Automatic build of OpenGL visualisation
8. Output of built physical components in verbose mode

Online Analysis - Requirements

- How do we make the online analysis work?
- Task
  - Make graphs of statistical quantities for a collection of runs
  - e.g. plot beam mean vs magnet current to look for magnet misalignment
  - e.g. plot beam emittance vs diffuser thickness to check input beam parameters
  - e.g. perform some cuts/sampling on the input beam, plot beam emittance vs a few beamline parameters to find equilibrium emittance
- Reminder of online data flow
  - Read network socket
  - Reconstruct all data spill-by-spill
  - Hand data in memory to histogramming routines
  - Write histograms to disk
  - Hand histograms to web front end for display to shifters
- Currently we never write particle data to disk

Option 1

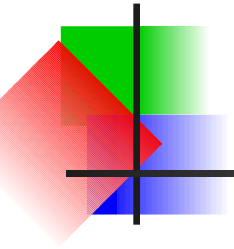
- We write out a reduced data set of global reconstruction hits for each run, which folks can analyse in a semi-offline way
  - Say global reconstruction “hit” consists of
    - Momentum = 3 doubles
    - Position = 3 doubles
    - Energy = 1 double
    - Time = 1 double
    - PID = 1 int+few doubles
    - 8*8 symmetric error matrix = 36 doubles
  - Each hit = ~50*8 bytes = 400 bytes per hit
  - Write out 100,000 tracks at, say 10-20 z-points ~ 200-400 MB per run (or lets say up to 400 MB per hour of running)
    - How many runs would we want to cache at any one time? Do we need a bigger disk?
    - How quickly can we post-process this data? (Probably okay?)
  - Provide a few scripts to make some “standard plots” based on this data
    - Needs a UI? See how it goes... maybe python is good enough
  - Experimenters bring their own scripts for fancy stuff

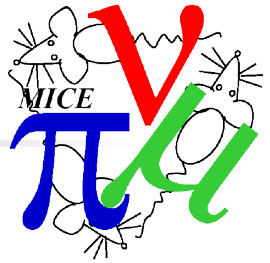
Option 2

- Add reducer(s) that calculates beam moments/emittances/etc on data in memory and write in a summary table
  - Module in MAUS framework comes after reconstruction
    - Extra bureaucracy
    - Potentially more reliable + usable
  - Don't ever need to write particle data to disk
  - But can't “change a cut” and then rerun
    - Once the data is out of memory, it's gone
- Probably makes Option 1 more attractive

Revised data flow (Assume Option 1)

- Reminder of online data flow
  - Read network socket
  - Reconstruct all data spill-by-spill
  - Hand data in memory to histogramming routines
  - Write histograms to disk
  - Write reduced data set to disk
    - Experimenters do their thing
  - Hand histograms to web front end for display to shifters
- Assume no online Monte Carlo required





FIN