Analysing Muons using the ATLAS software

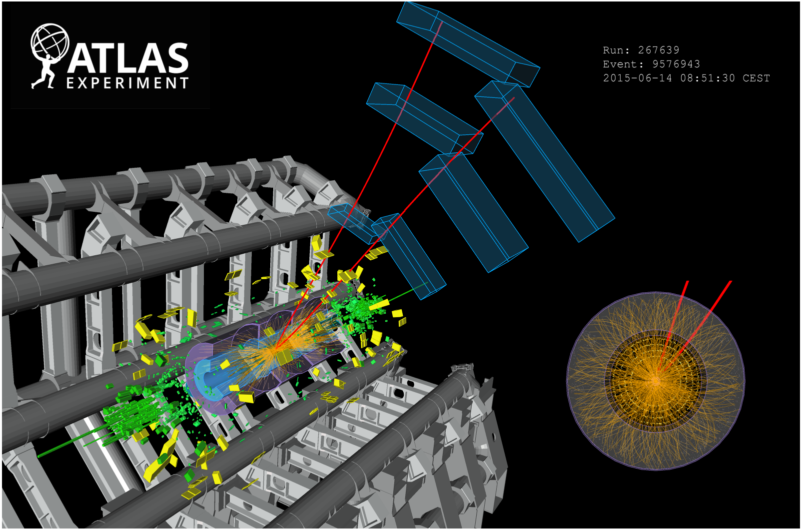
ATLAS software tutorial

Nicolas Köhler

on behalf of the Muon Combined Performance (MCP) group

Tuesday 22nd October, 2019

Muons in ATLAS

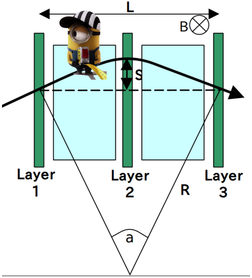


Muon reconstruction in ATLAS

Muons deposit only little energy in the calorimeters (minimal ionizing particle)

→ They cannot be stopped in ATLAS

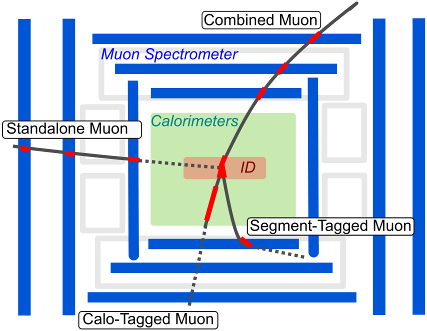
- Muon trajectory bent by the Solenoid & Toroid magnets
- Inner Detector (ID) & Muon Spectrometer (MS) record their hits



→ Momentum estimated from the sagitta of their path $p_T \propto \frac{L^2 \cdot B}{s}$

- Information from the calorimeter corrects for energy loss & allows for additional tagging
- I'll show you today...
  - ... the basics of the `xAOD::Muon` EDM in ATLAS
  - ... how to calibrate & select muons
  - ... how to retrieve (reconstruction) efficiencies

The four muon types



Reconstruction with the ID

- Combined muons : Fit the ID- & MS-track into one single track
- Calorimeter-Tagged muons: ID tracks with additional small energy deposits in the calorimeter (at $|\eta| \approx 0$)
- Segment-Tagged muons: ID tracks combined with single segments of the MS (at low energies)

The ATLAS-MS can reconstruct the muon independently of the ID

- Standalone muons: Used only at high $|\eta|$ (> 2.5) beyond the coverage of the ID

xAOD::Muon - your stairway to...

xAOD::Muon (← link) is the class handling all information concerned muons

Associated particles

```
/// @brief Returns a pointer (which can be NULL) to the  TrackParti
const TrackParticle* trackParticle( TrackParticleType type) const;
///This is determined in the following order:
///  1. CombinedTrackParticle
///  2. InnerDetectorTrackParticle
///  3. ExtrapolatedMuonSpectrometerTrackParticle
///  4. MSOnlyExtrapolatedMuonSpectrometerTrackParticle
///  5. MuonSpectrometerTrackParticle
const TrackParticle* primaryTrackParticle() const;
/// Retrieve the associated cluster with a bare pointer
const CaloCluster* cluster() const;
/// @brief Number of MuonSegments linked to by this Muon.
size_t nMuonSegments() const;
/// @brief Returns a pointer to the specified MuonSegment.
/// @param i Index of the MuonSegment requested. If i is n
const MuonSegment* muonSegment( size_t i ) const;
```

Kinematics

```
/// The transverse momentum (\f$p_T\f$) of th
virtual double          pt() const;
/// The pseudorapidity (\f$\eta\f$) of the pa
virtual double          eta() const;
/// The azimuthal angle (\f$\phi\f$) of the p
virtual double          phi() const;
/// The invariant mass of the particle..
virtual double          m() const;
/// The total energy of the particle.
virtual double          e() const;
/// The true rapidity (y) of the particle.
virtual double          rapidity() const;
```

Muon quality

```
enum Quality {Tight, ///
              Medium, ///
              Loose, ///
              VeryLoose};
Quality quality() const;
```

Many more methods to access all information

Prerequisites to use muons in your analysis

- Which lines do you need to put where to compile your code?
- Needed dependencies in your `CMakeLists.txt` file:

  ```
  Event/xAOD/xAODMuon
  PhysicsAnalysis/Interfaces/MuonAnalysisInterfaces
  ```

- Add the include statements to the header of your analysis class

```
// General information about the event.
#include <xAODEvent/EventInfo.h>
// Access the Particle & vertex containers
#include <xAODMuon/MuonContainer.h>
#include <xAODTracking/VertexContainer.h>
#include <xAODTracking/TrackParticleContainer.h>
// Helper functions to calculate the
// closest approach to the primary vertex
#include <xAODTracking/TrackParticlexAODHelpers.h>
```

Retrieve the MuonContainer

- At the very beginning of each event you first need to retrieve the Containers

→ Add therefore in your event execution method

```
// Retrieve the muons from the event
const xAOD::MuonContainer* muons= nullptr;
ATH_CHECK(evtStore()->retrieve(muons, "Muons"));
// Retrieve the event info
const xAOD::EventInfo* ev_info = nullptr;
ATH_CHECK(evtStore()->retrieve(ev_info, "EventInfo"));
// We also need the primary vertex. Please consider the track-tutorial
// how to carry this out properly
const xAOD::Vertex* pv = nullptr;
```

First simple analysis

- Print out some information of the muon and its track

```
float primvertex_z = pv ? pv->z() : 0;
// Loop over the container and print some basic properties
for ( auto mu : *muons) {
    // retrieve the primary track
    const xAOD::TrackParticle* mu_trk = mu->primaryTrackParticle();
    float d0(-1), z0_sintheta(0), d0sig(0);
    // Calculate the impact parameters
    if (mu_trk){
        z0 = (mu_trk->z0() + mu_trk->vz() - primvertex_z);
        d0 = mu_trk->d0();
        d0sig = xAOD::TrackingHelpers::d0significance( mu_trk ,
                                                ev_info->beamPosSigmaX(),
                                                ev_info->beamPosSigmaY(),
                                                ev_info->beamPosSigmaXY());
    } else {continue;}
    ATH_MSG_INFO("Found muon with pt: "<< mu->pt()/1.e3
                            <<" eta: " << mu->eta()
                            <<" phi: "<<mu->phi()
                            <<" d0: "<<d0<<" rel d0sig: "<<d0sig /d0
                            <<" z0: "<<z0);
}
```

Calibrate the muon momenta

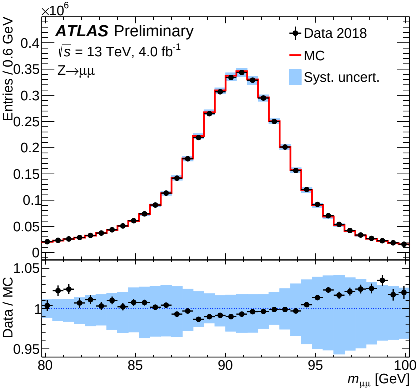
Δs_0: Energy loss inside ATLAS
Δs_1: Magnetic field integral & distortions

$$p_T \rightarrow \frac{\Delta s_0 + (1 + \Delta s_1) p_T}{\mathcal{G}\left(\mu=1, \sigma=\sqrt{\left(\frac{\Delta r_0}{p_T}\right)^2 + \Delta r_1{}^2 + (\Delta r_2\ p_T)^2}\right)}$$

Δr_0: Energy loss fluctuations
Δr_1: Multiple scattering & local distortions
Δr_2: Intrinsic resolution & misalignment

- Muon momentum needs to be adjusted between data and simulation
- Correction of small residual discrepancies
- Calibration parameters from $(Z\ /\ J/\psi) \rightarrow \mu\mu$ decays
- ID & MS track separately calibrated

  Muon momentum scale known to $\sim 0.05\%$!
- Calibration procedure provided by `IMuonCalibrationAndSmearingTool`



The MuonCalibrationAndSmearingTool

- Recommendations how to calibrate muon momenta are summarized here
- Add these statements to the top of your header file

```
// Interface methods for MuonCalibration
#include <MuonAnalysisInterfaces/IMuonCalibrationAndSmearingTool.h>
// Takes care to create instances of the analysis tools
#include <AsgTools/AnaToolHandle.h>
```

- Add this line to your class declaration

```
asg::AnaToolHandle<CP::IMuonCalibrationAndSmearingTool> m_muonCalibTool;
```

- Finally setup the Tool inside your initialize section

```
// the MuonCalibrationPeriodTool is already preconfigured
// according to latest MCP recommendations
muonCalibTool.setTypeAndName(
    "CP::MuonCalibrationPeriodTool/MuonCalibTool");
// create the tool and retrieve it
ATH_CHECK(muonCalibTool.retrieve());
```

- Have a look at this TWiki page to learn about pile-up reweighting

The MuonCalibrationAndSmearingTool

- Since calibration changes the p_T of your muons, need to create a so-called `shallowCopyContainer` to store them (cf. `xAOD` EDM tutorial)

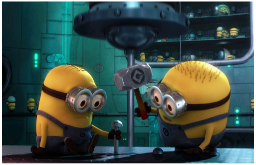
```
// The muons container is locked. I. e. you cannot change any property
// of the muons. For the calibration we need to have a writeable
// container let's create it first. I'm using a ShallowCopy here
std::pair<xAOD::MuonContainer*,
    xAOD::ShallowAuxContainer*> shallowcopy =
            xAOD::shallowCopyContainer(*muons);
// Use the EvtStore to take care of cleaning up the container
// if it is no longer needed. Define foreach container a unique_name
ATH_CHECK(evtStore()->record(shallowcopy.first, "CalibratedMuons"));
ATH_CHECK(evtStore()->record(shallowcopy.second,"CalibratedMuonsAux."));

xAOD::MuonContainer calibrated_muons = shallowcopy.first;
if (!xAOD::setOriginalObjectLink(*muons, *calibrated_muons)) {
    ATH_MSG_ERROR("Failed to set original object links for muons");
    // Bail out of the loop
}
```

The MuonCalibrationAndSmearingTool

- To calibrate the muon, call the `MuonCalibrationAndSmearingTool` and it changes the p_T of the muon you pass to it (cannot be the original `const` muon)

```
// Now we can calibrate the muons by calling the tool
for (auto mu : *calibrated_muons) {
    // Remember that auto mu -> xAOD::Muon* mu
    // The tool requires the reference to the object -> *mu
    if (m_muonCalibTool->applyCorrection(*mu) ==
                                        CP::CorrectionCode::Error){
        ATH_MSG_ERROR("Something went wrong");
        // Bail out
    }
}
```

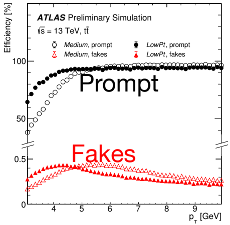


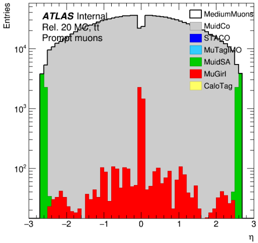
Pick the good muons in the event

- Most of the muons not from primary process or poorly measured

$\rightarrow$ Apply muon selection based on hits & fit quality

- MCP provides 5 dedicated working points (c.f. Twiki):
  `Loose`, `Medium`, `Tight`, `LowPt`, `HighPt`





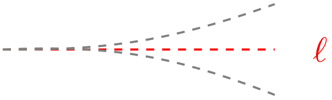
- The `IMuonSelectionTool` is the recommended tool for muon selection
- Requirements on $z_0 \sin\theta < 0.5$ mm & $\sigma(d_0)/d_0 < 5$ (TTVA cuts) reject muons from secondary processes

A short word about isolation

- Electrons & muons from a primary *W*, *Z* or SUSY decay (real) expected to fly through ATLAS without a particle appearing in the close vicinity
- Leptons from a *b* or π-decay originate from a QCD-jet (fake) expected to have many accompanied particles close-by

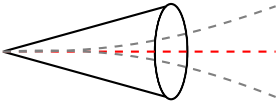
A short word about isolation

- Electrons & muons from a primary *W*, *Z* or SUSY decay (real) expected to fly through ATLAS without a particle appearing in the close vicinity
- Leptons from a *b* or π-decay originate from a QCD-jet (fake) expected to have many accompanied particles close-by



A short word about isolation

- Electrons & muons from a primary W, Z or SUSY decay (real) expected to fly through ATLAS without a particle appearing in the close vicinity
- Leptons from a b or π-decay originate from a QCD-jet (fake) expected to have many accompanied particles close-by



$\rightarrow$ The concept of isolation helps to efficiently suppress secondary leptons by defining $p_{\mathrm{T}}^{\mathrm{(Var)Cone}\mathcal{X}}$ & $E_{\mathrm{T}}^{\mathrm{Cone}\mathcal{X}}$

1. The sum of the p_{T} of all ID-tracks close by tracks within $\Delta R < \min\left(\frac{p_{\mathrm{T}}(\ell)}{10\ \mathrm{GeV}}, \frac{\mathcal{X}}{100}\right)$
2. The sum of the E_{T} of all calorimeter within $\Delta R < \frac{\mathcal{X}}{100}$

- The `IsolationSelectionTool` provides many working points based on the ratio of the isolation variables to the lepton-p_{T}

Selecting muons and checking isolation

- Needed includes to use the both tools

```
#include <MuonAnalysisInterfaces/IMuonSelectionTool.h>
#include <IsolationSelection/IIsolationSelectionTool.h>
```

- Declare the member variables in your analysis class

```
asg::AnaToolHandle<CP::IMuonSelectionTool> m_muonSelTool;
asg::AnaToolHandle<CP::IIsolationSelectionTool> m_iso_tool;
```

- Create the instance of the MuonSelectionTool

```
muonSelTool.setTypeAndName(
    "CP::MuonSelectionTool/MuonSelectionTool");
ATH_CHECK(muonSelTool.retrieve());
```

- And finally also the IsolationSelectionTool

```
m_iso_tool.setTypeAndName(
    "CP::IsolationSelectionTool/IsolationSelectionTool");
// since this is a common tool also for electrons/photons
// set also other "Working Points"
ATH_CHECK(m_iso_tool.setPropery("ElectronWP", "FCLoose"));
ATH_CHECK(m_iso_tool.setPropery("MuonWP", "FCTight"));
ATH_CHECK(m_iso_tool.retrieve());
```

Apply reconstruction and isolation criteria...

- Select Medium muons with $p_T > 5$ GeV and $|\eta| < 2.7$

```
// create VIEW_ELEMENTS container for Medium muons
xAOD::MuonContainer medium_muons(SG::VIEW_ELEMENTS);
// loop on the calibrated muons and select only the Medium ones
// with pt > 5GeV and |eta|<2.7
for (auto mu : *calibrated_muons) {
    if (std::fabs(mu->eta())>2.7 || mu->pt()<5000) continue;
    if (m_muonSelTool->getQuality(*mu) <= xAOD::MuonMedium) {
        medium_muons.push_back(mu);
    }
}
```

- Check that muons also pass isolation criteria

```
// only check this for the Medium muons
for (auto mu : medium_muons) {
    if (m_iso_tool->accept(mu)) {
        ATH_MSG_INFO("Found isolated muon");
    }
}
```

Muon efficiencies - fine-tune your Monte Carlo

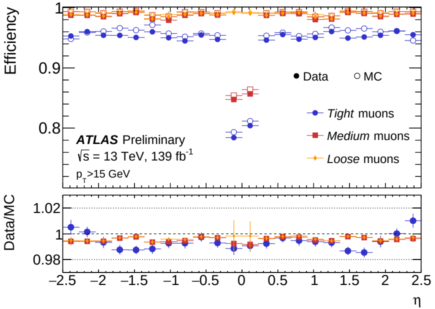
- Simulation might not exactly reflect the detector behavior during data taking

$\longrightarrow$ Tiny deviations in muon modeling $\rightarrow$ tiny differences in efficiencies

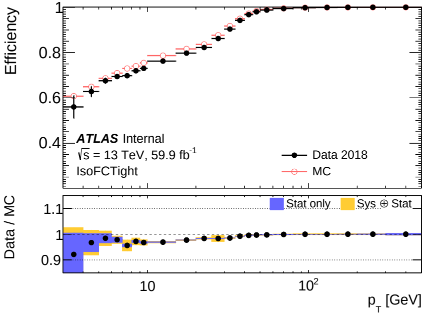
$\longrightarrow$ The efficiency of the muon reconstruction, TTVA & isolation selection measured in Data & Monte Carlo using $(Z\,/\,J/\psi) \rightarrow \mu\mu$-decays

- Efficiency Scale Factors (SFs) applied on simulation to match Data

Reconstruction & Identification



Isolation



IMuonEfficiencyScaleFactors

- Latest prescriptions of the SF measurements summarized here
- `IMuonEfficiencyScaleFactors` provides framework to apply *reconstruction*, *isolation* & *TTVA* SFs
- Needed include:

```
#include <MuonAnalysisInterfaces/IMuonEfficiencyScaleFactors.h>
```

- Declare member variables, one for each measurement:

```
asg::AnaToolHandle<CP::IMuonEfficiencyScaleFactors.h> m_reco_medium_sf;
asg::AnaToolHandle<CP::IMuonEfficiencyScaleFactors.h> m_reco_ttva_sf;
asg::AnaToolHandle<CP::IMuonEfficiencyScaleFactors.h> m_reco_iso_sf;
```

- Initialize the tools (e.g. for reconstruction SFs):

```
m_reco_medium_sf.setTypeAndName(
    "CP::MuonEfficiencyScaleFactors/Medium_SF");
// the "WorkingPoint" property already encodes whether
// the tool is for reconstruction/isolation/ttva...
ATH_CHECK(m_reco_medium_sf.setProperty("WorkingPoint", "Medium"));
ATH_CHECK(m_reco_medium_sf.retrieve());
```

Calculate the muon scale-factor

```
// retrieve the muon reconstruction efficiency scale factor per event
// only take the 'good' selected muons to calculate it
float tot_sf = 1;
for (auto mu : good_muons) {
    float mu_sf = 1;
    if (m_reco_medium_sf->getEfficiencyScaleFactor(*mu, mu_sf, ev_info)
        != CP::CorrectionCode::Ok) {
        // printout warning, that something went wrong with
        // retrieving the scale factor
    }
    tot_sf *= mu_sf;
}
```

Concluding remarks

- MCP Twiki contains all relevant information about handling of muons
- Make yourself comfortable with the `xAOD::Muon` EDM
- There is still much to learn about:
  - Dealing with systematic variations
  - Muon trigger selection & efficiencies
  - ...

Request developer access!

$\longrightarrow$ At some point, you will need an analysis *framework* (such as XAMPP, ...)

- Join MCP (or Muon Software) if you want to work in an awesome CP group

Thanks & enjoy your time in ATLAS!