

AUGUST 21, 2026



QUALITY ASSURANCE FOR CBM & STS ELECTRONICS WORK

Central Quality Assurance software development and first tests. Use of quality assurance software in simulations using different event generators.

Testing and measurements with an STS half-unit.

EMILIO SALDAÑA MARKUS★¹

Frederic Linz²

¹Department of Physics, College of Natural Sciences, The University of Texas at Austin

²CBM Experiment, GSI Helmholtzzentrum für Schwerionenforschung GmbH

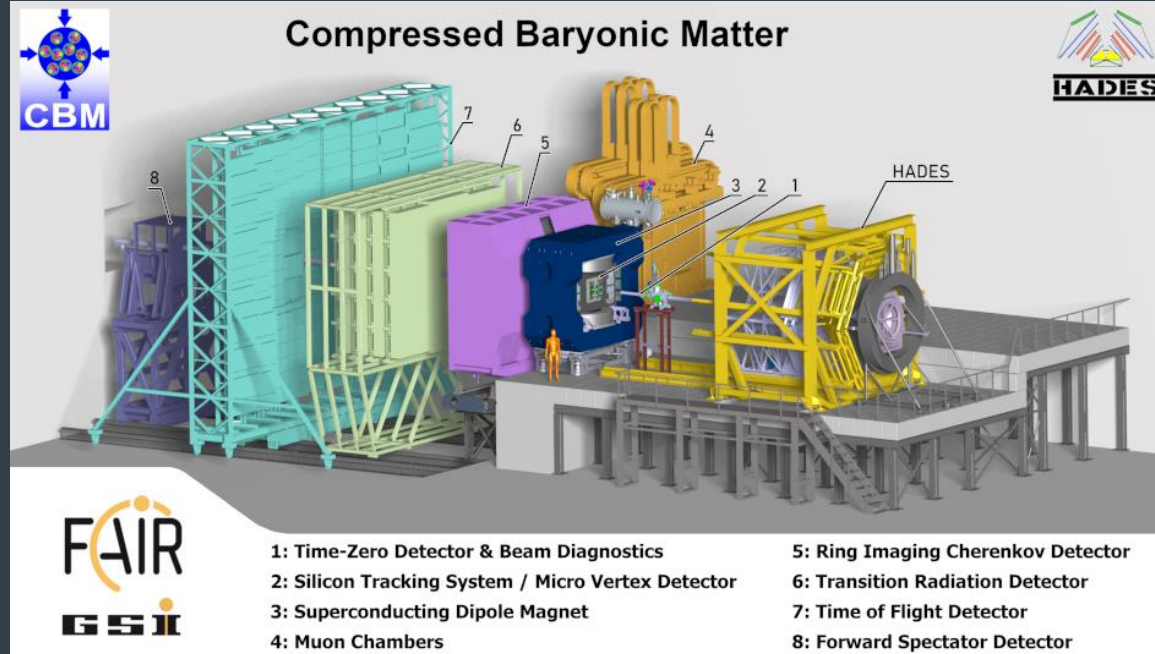
★esaldana@utexas.edu

Outline

- CBM Experiment
 - General view and role of simulations
 - CBMROOT
- Quality Assurance Software Development
 - Central QA software and workflow
 - Basic observable comparison for different event generators
 - Achievements and Next steps
- STS Work with Electronics
 - Silicon Tracking System / Half-Unit
 - Electronics Tests and Measurements

CBM Experiment

- Fixed-target heavy-ion experiment at FAIR.
- Exploration of the QCD phase diagram at high net-baryon density.
- Designed for intense beams coming from the SIS-100.
- Operation at very high interaction rates.



Simulations and Analysis



- Simulations allow us to see how our detectors, tracking, and analysis software behave.
- Quality Assurance allows us to identify differences caused by the generator, configuration, geometry, or software version.

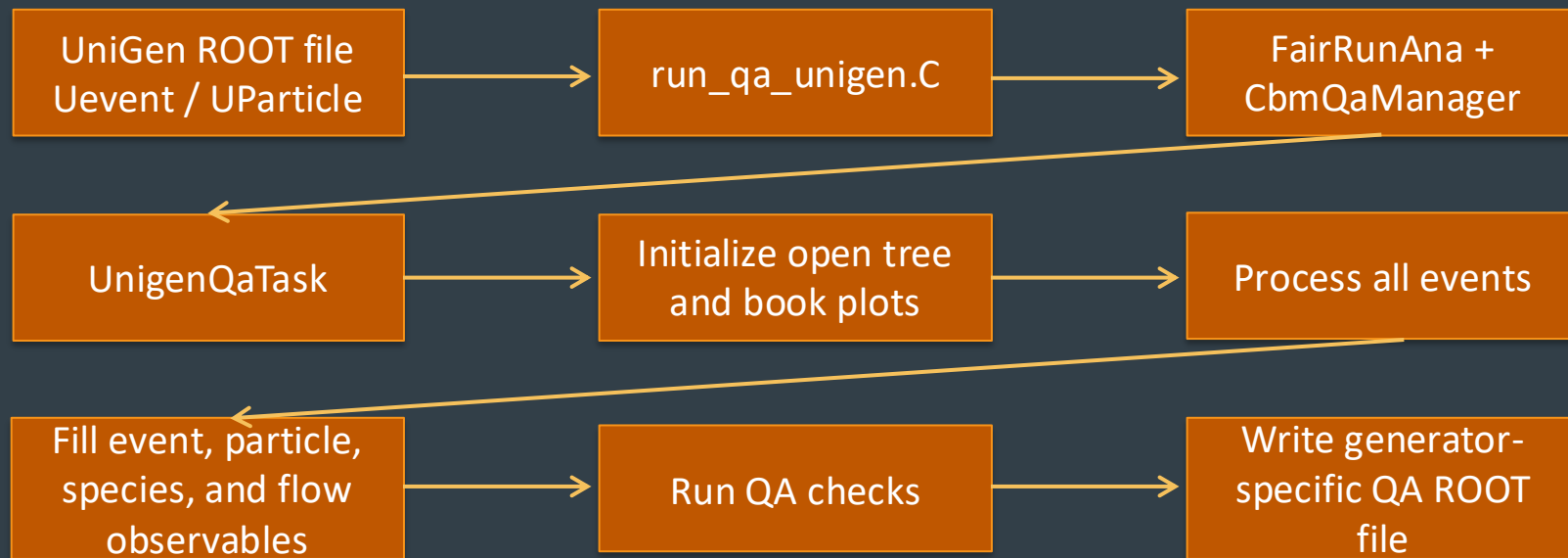
Central QA Software Objectives

- Process CBM simulation samples consistently.
- Calculate relevant observables.
- Compare generators, configurations, and CBMROOT versions.
- Produce standardized plots and outputs.
- Configurable QA, commonly developed and easy to execute.

Central QA to be structured:

- Independent execution for each simulation step within CBMROOT:
 - UniGen (important to monitor input of CBM simulations)
 - Transport
 - Digitization
 - Reconstruction
- Step-by-step control of the simulation chain
- Structure within Transport/Digitization/Reconstruction:
 - General QA with basic/most-important plots that are useful for everyone
 - Very detailed QA meant for experts (such as detector/tracking experts only)
- Configurable
 - Realized by a YAML configurable QA binary; under finalization (provided by S. Zharko)
 - See Merge Request for reference: https://git.cbm.gsi.de/computing/cbmroot/-/merge_requests/2439

QA Workflow (for now, only UniGen)



Note: macro is temporary. Soon to be replaced by QA binary.

Current structure and observables

- Generator-specific root files with all predefined observables.
- Normalized plots of all generators for direct comparison.
- Observables produced:

Level	Observables
Event	Impact parameter, reaction-plane angle, multiplicity versus impact parameter
Inclusive particle	Species yields, p_t , y_{CM} , phase space
Primary particle	Multiplicity, species yields, y_{CM}
Species-specific	p_t , phase space, directed flow, elliptic flow
Flow	Azimuthal angle relative to RP, directed flow, elliptic flow.

Modification

- This is just the kick-off. Future development of the QA will happen along the way.
- UniGen QA fully defined in a dedicated class derived from CbmQaTask:
 - CbmQaUnigenTask.h
 - CbmQaUnigenTask.cxx
 - run_qa_unigen.C (obsolete soon, after QA binary is set in place)

Code structure: QA task file (.h)

Class inheritance and lifecycle:

```
namespace qa::unigen
{
    class UnigenTask : public CbmQaTask {
    public:
        UnigenTask(int verbose, bool isMCUsed);

        void Check() override;

        void SetFileName(const TString& name) { fsFileName = name; }
        void SetGeneratorName(const TString& name) { fsGeneratorName = name; }

    protected:
        InitStatus InitQa() override;
        void ExecQa() override;
        void CreateSummary() override;
        void DeInit() override;

    private:
        void InitHistograms();
        void ProcessEvent(const UEvent& event);
    };
} // namespace qa::unigen
```

Histogram organization:

```
struct EventHistograms {
    TH1D* impactParameter{nullptr};
    TH1D* reactionPlane{nullptr};
    TH2D* multiplicityVsImpactParameter{nullptr};
};

struct InclusiveParticleHistograms {
    TProfile* species{nullptr};
    TH1D* transverseMomentum{nullptr};
    TH1D* rapidity{nullptr};
    TH2D* phaseSpace{nullptr};
};

struct InclusiveFlowHistograms {
    TH1D* relativeAzimuth{nullptr};
    TProfile* v1VsRapidity{nullptr};
    TProfile* v2VsRapidity{nullptr};
    TProfile* v1VsPt{nullptr};
    TProfile* v2VsPt{nullptr};
};
```

Configurable particle species:

```
std::unordered_set<int> fPdgCodeBook = {
    211,   -211,   // pi+, pi-
    321,   -321,   // K+, K-
    2212,  -2212,  // proton, antiproton
    3122,  -3122,  // Lambda, anti-Lambda
};

std::unordered_map<int, ParticleHistograms> fParticleHistograms;
```

Note: particle list shall be moved to the YAML configuration file in the future. Then, user can define which particles to look at without recompiling.

Code structure: event processing file (.cxx)

Reading UniGen file

```
fpFile = new TFile(fsFileName, "READ");

fpTree = fpFile->Get<TTree>("events");
if (!fpTree) {
    LOG(error) << GetName() << ": no events tree found";
    return kERROR;
}

if (!fpTree->GetBranch("event")) {
    LOG(error) << GetName() << ": no event branch found";
    return kERROR;
}

fpTree->SetBranchAddress("event", &fpEvent);
InitHistograms();
```

Event processing:

```
void UnigenTask::ProcessEvent(const UEvent& event)
{
    fHistograms.impactParameter->Fill(event.GetB());
    fHistograms.reactionPlane->Fill(event.GetPhi());

    fHistograms.multiplicityVsImpactParameter->Fill(
        event.GetB(), event.GetNpa());

    for (int iParticle = 0; iParticle < event.GetNpa(); ++iParticle) {
        const auto* particle = event.GetParticle(iParticle);
        if (!particle) { continue; }

        ProcessParticle(*particle, event.GetPhi());

        if (IsPrimary(*particle)) {
            ProcessPrimaryParticle(*particle);
        }
    }
}
```

Complete-tree execution:

```
void UnigenTask::ExecQa()
{
    if (fbProcessed) { return; }
    fbProcessed = true;

    const Long64_t nEntries = fpTree->GetEntriesFast();

    for (Long64_t entry = 0; entry < nEntries; ++entry) {
        if (fpTree->GetEntry(entry) <= 0 || !fpEvent) {
            continue;
        }

        ProcessEvent(*fpEvent);
    }
}
```

Calculations and filling (for example, flow):

```
const double v1Weight = std::cos(deltaPhi);
const double v2Weight = std::cos(2. * deltaPhi);

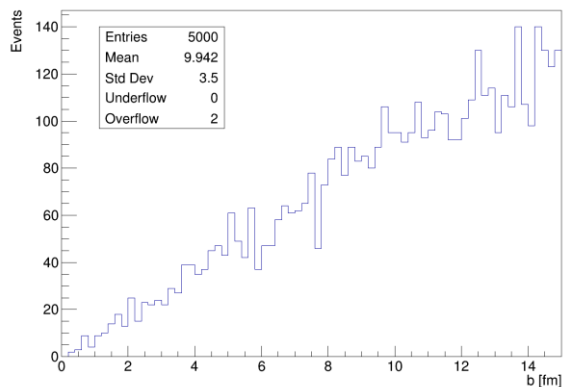
fInclusiveFlowHistograms.relativeAzimuth->Fill(deltaPhi);
fInclusiveFlowHistograms.v1VsPt->Fill(pt, v1Weight);
fInclusiveFlowHistograms.v2VsPt->Fill(pt, v2Weight);
```

Event Generator Comparison

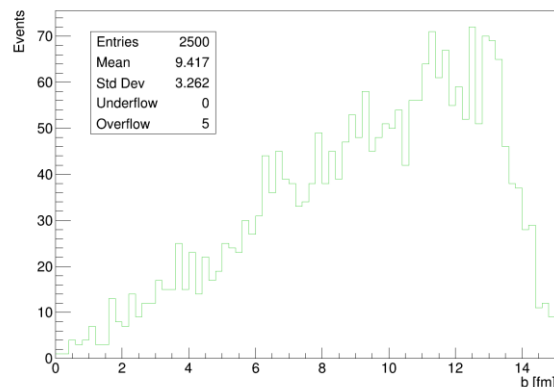
- Event generators: UrQMD, PhQMD, and DCM-QGSM-SMM.
- Tested with one file per generator (more relevant data by using several files of data).
- Files used (for reference):
 - UrQMD: /lustre/cbm/data/sim/gen/urqmd-40/AuAu/p12/mbias/unigen/
 - PHQMD: /lustre/cbm/prod/gen/phqmd52_winn/auau/pbeam12agev/mbias/all_clusters/
 - DCM-QGSM-SMM: /lustre/cbm/users/fkornas/gen/data/dcmqgsm/unigen/
- Different number of events per generator input file (5000, 1000, 2500).
- Events are: Au-Au, min bias.
- Normalization for some observables, not for others.
- More precise analysis would require several files of data.
 - Rare observables require higher statistics.
- Still, basic observables can already be investigated using a single input file.

Impact Parameter

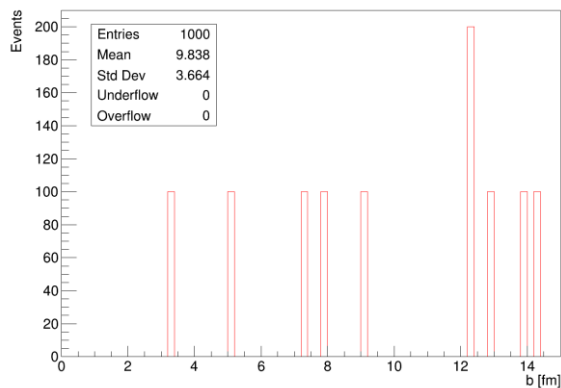
UrQMD: UniGen impact parameter



DCM-QGSM-SMM: UniGen impact parameter

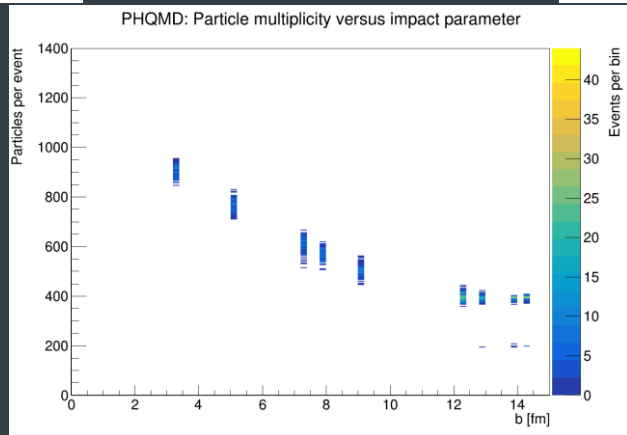
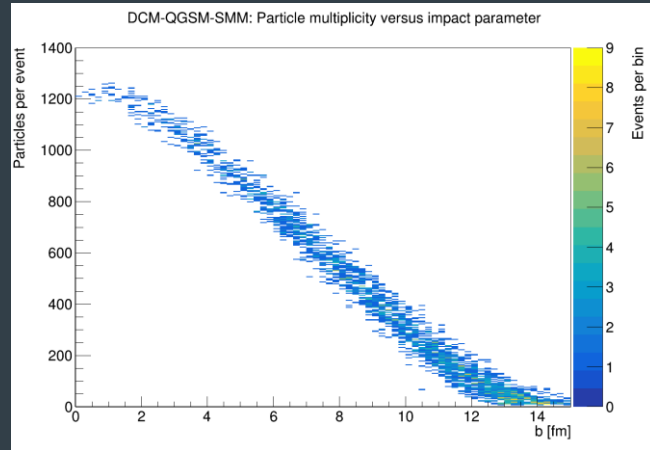
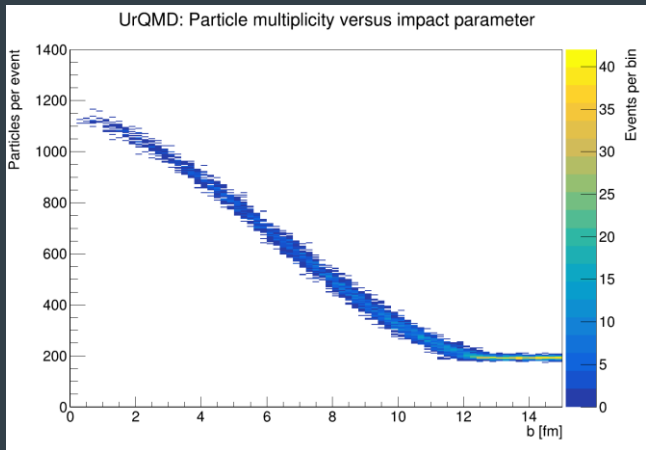


PHQMD: UniGen impact parameter



Note: PHQMD has special feature of always bunches of 100 events with same impact parameter.

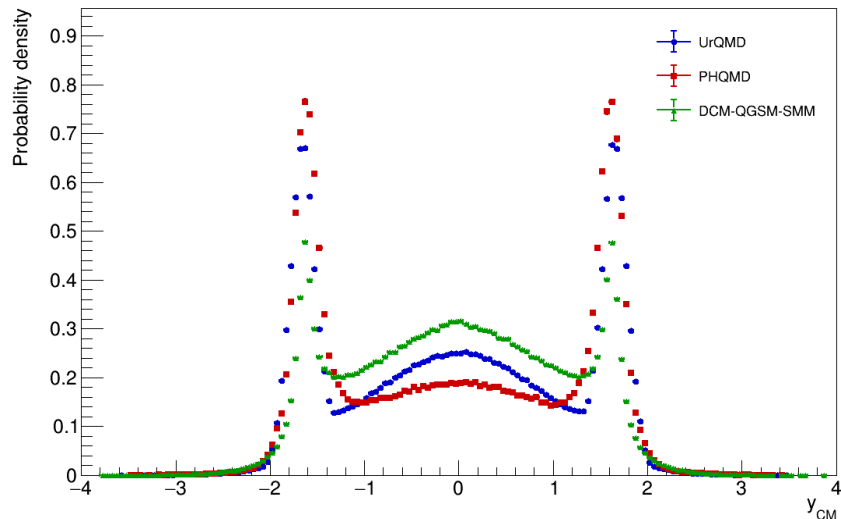
Multiplicity vs Impact Parameter



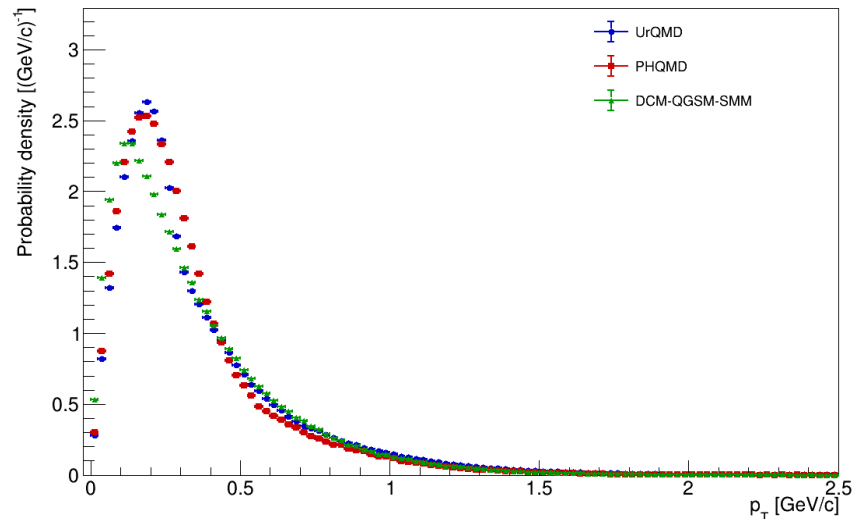
Large impact
parameter -> low
particle production,
and viceversa.

Rapidity and Transverse Momentum

Normalized inclusive rapidity comparison



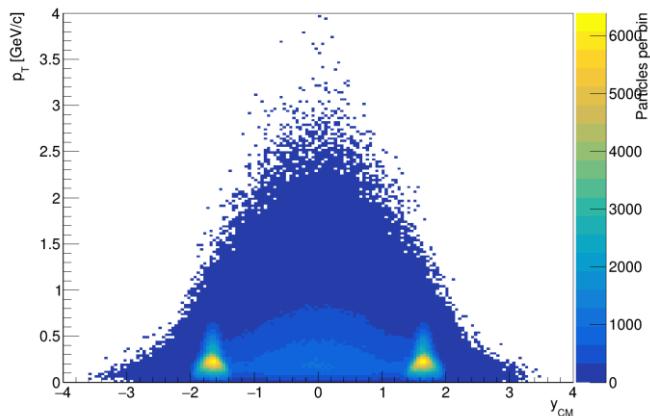
Normalized inclusive transverse-momentum comparison



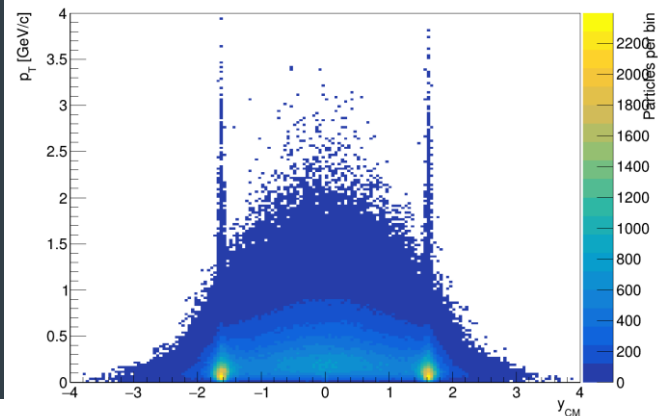
Peaks on target-like matter, participants at center, and projectile-like matter.
Large peaks due to spectators.

Phase space

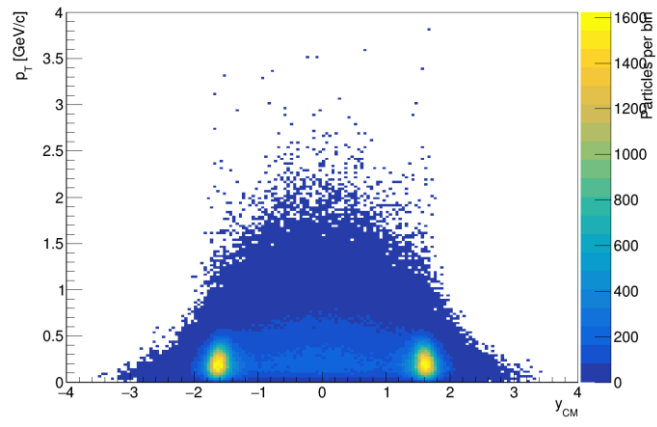
UrQMD: Inclusive particle phase space



DCM-QGSM-SMM: Inclusive particle phase space

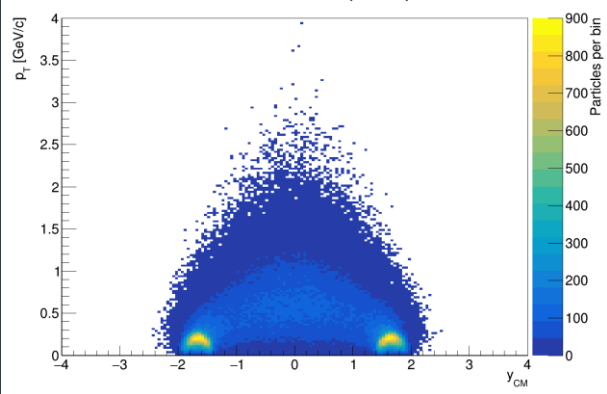


PHQMD: Inclusive particle phase space

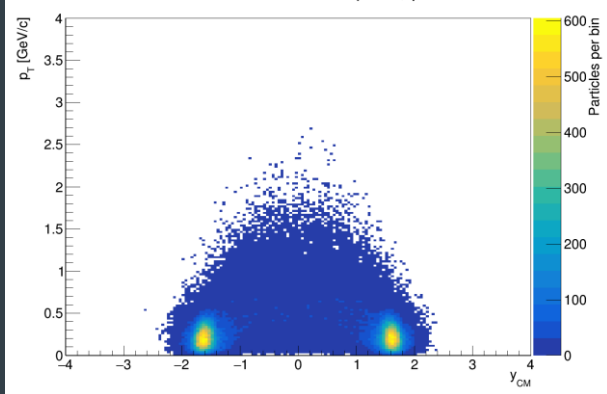


Species specific: protons vs lambdas

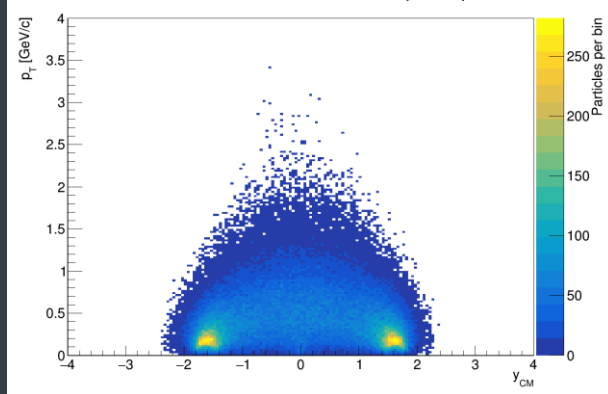
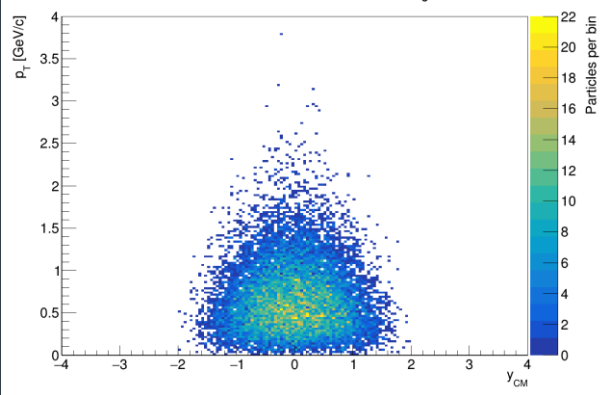
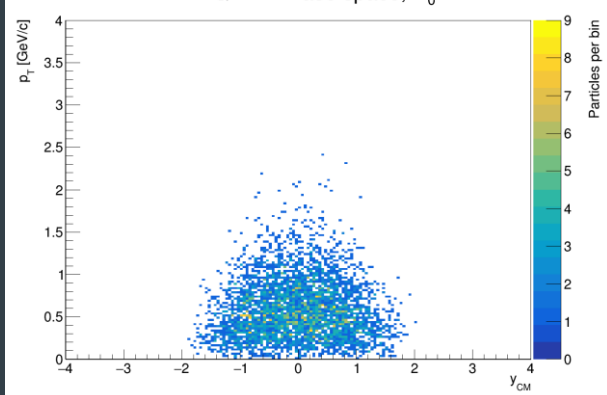
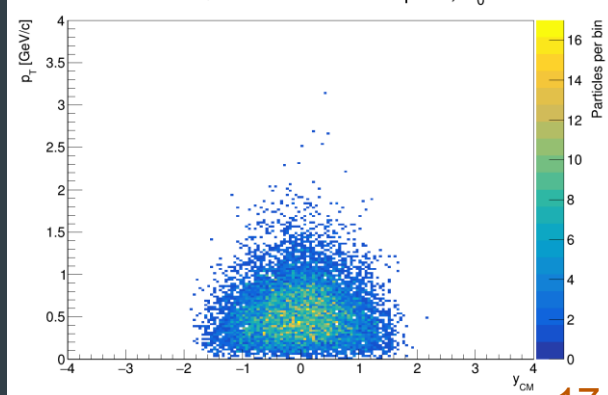
UrQMD: Phase space, p



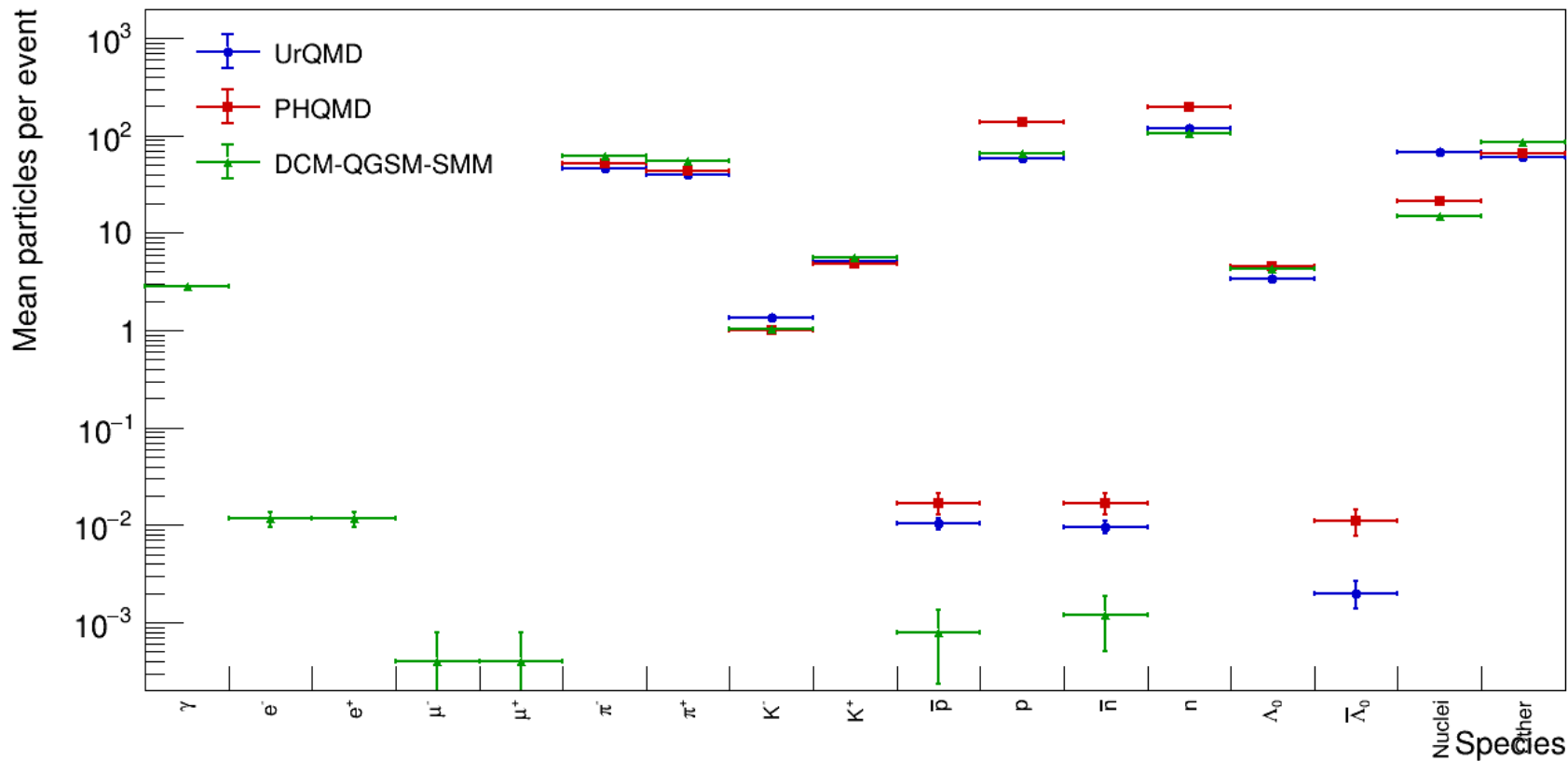
PHQMD: Phase space, p



DCM-QGSM-SMM: Phase space, p

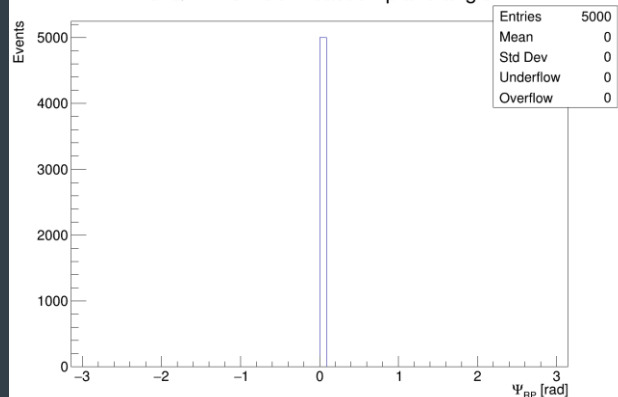
UrQMD: Phase space, Λ_0 PHQMD: Phase space, Λ_0 DCM-QGSM-SMM: Phase space, Λ_0 

Particle-species comparison

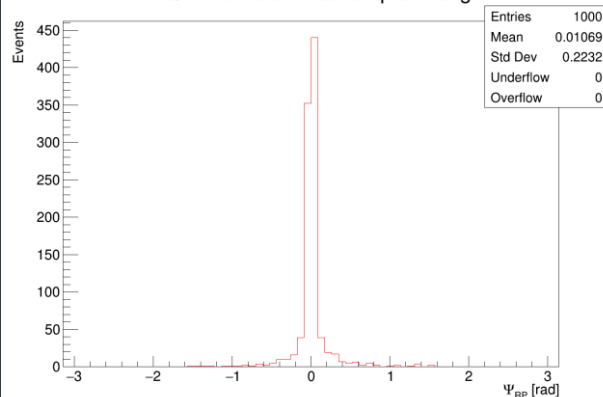


Reaction Plane Angle and Particle Azimuth relative to RP angle

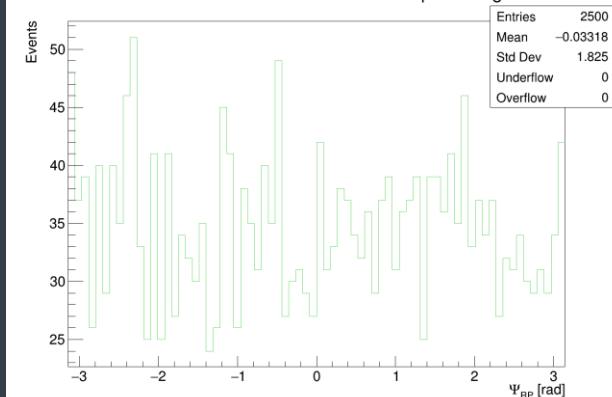
UrQMD: UniGen reaction-plane angle



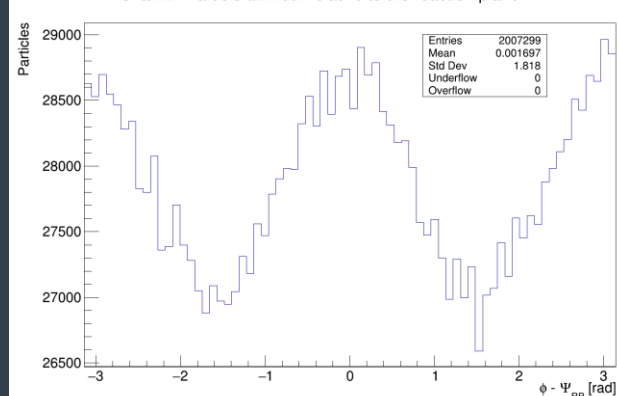
PHQMD: UniGen reaction-plane angle



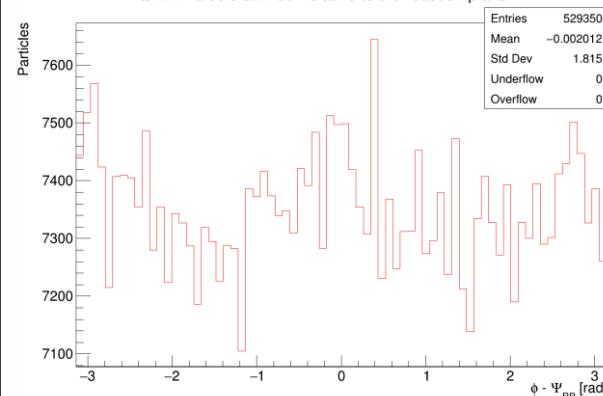
DCM-QGSM-SMM: UniGen reaction-plane angle



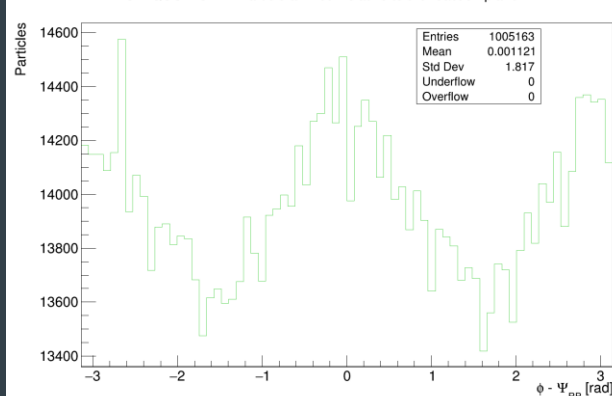
UrQMD: Particle azimuth relative to the reaction plane



PHQMD: Particle azimuth relative to the reaction plane



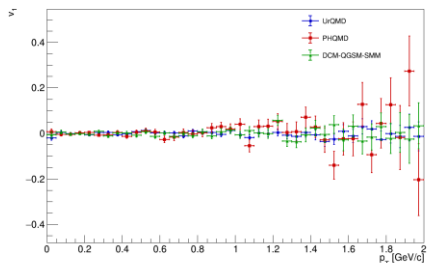
DCM-QGSM-SMM: Particle azimuth relative to the reaction plane



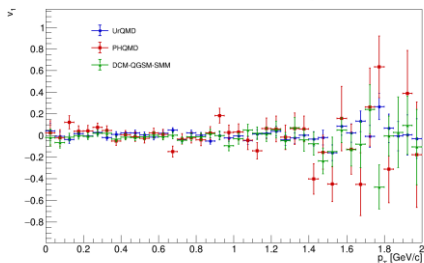


Directed flow (v_1)

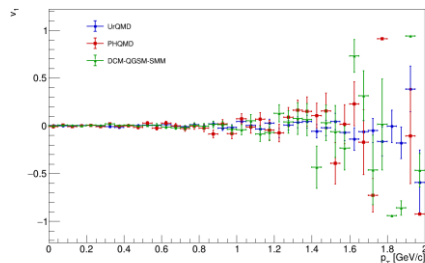
Directed-flow comparison, p



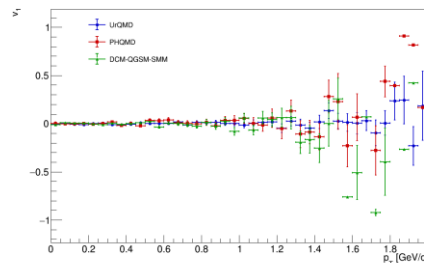
Directed-flow comparison, Λ_0



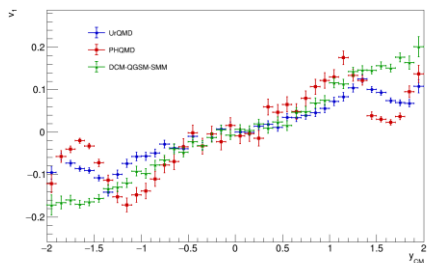
Directed-flow comparison, π^+



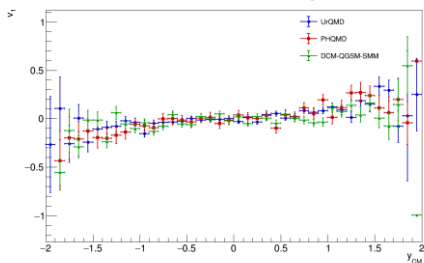
Directed-flow comparison, π^-



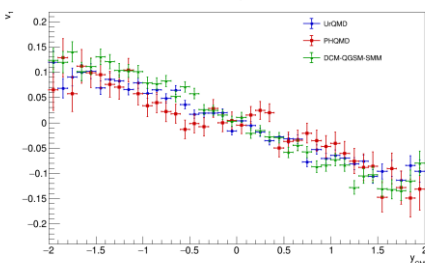
Directed-flow comparison, p



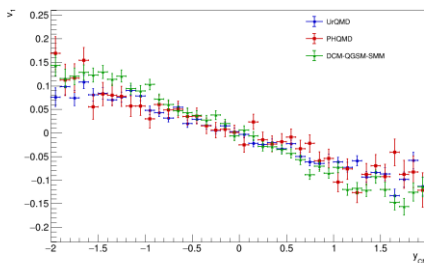
Directed-flow comparison, Λ_0



Directed-flow comparison, π^+

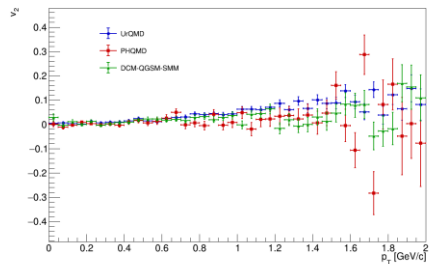


Directed-flow comparison, π^-

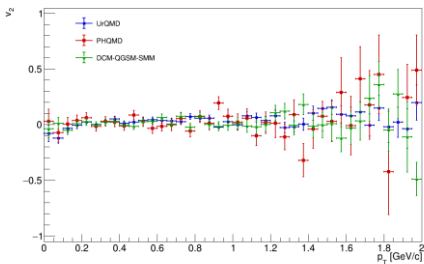


Elliptic flow (v_2)

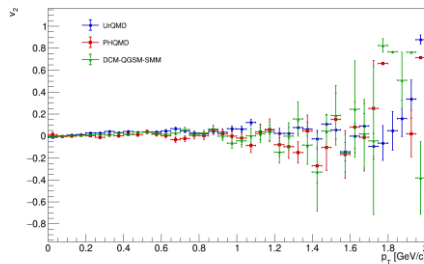
Elliptic-flow comparison, p



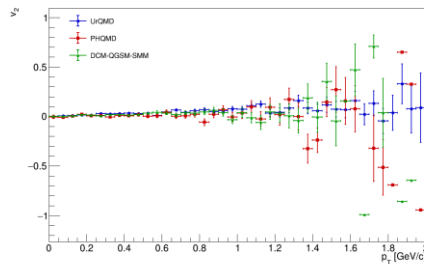
Elliptic-flow comparison, Λ_0



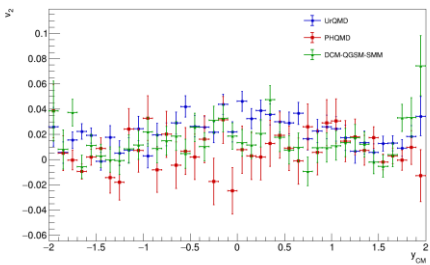
Elliptic-flow comparison, π^+



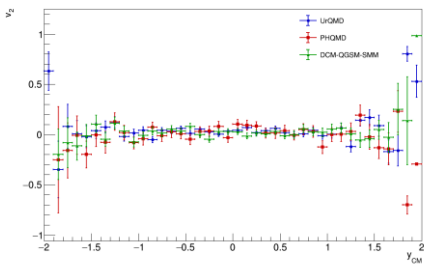
Elliptic-flow comparison, π^-



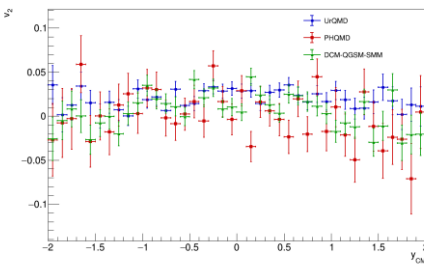
Elliptic-flow comparison, p



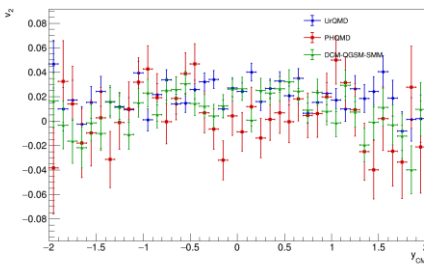
Elliptic-flow comparison, Λ_0



Elliptic-flow comparison, π^+



Elliptic-flow comparison, π^-

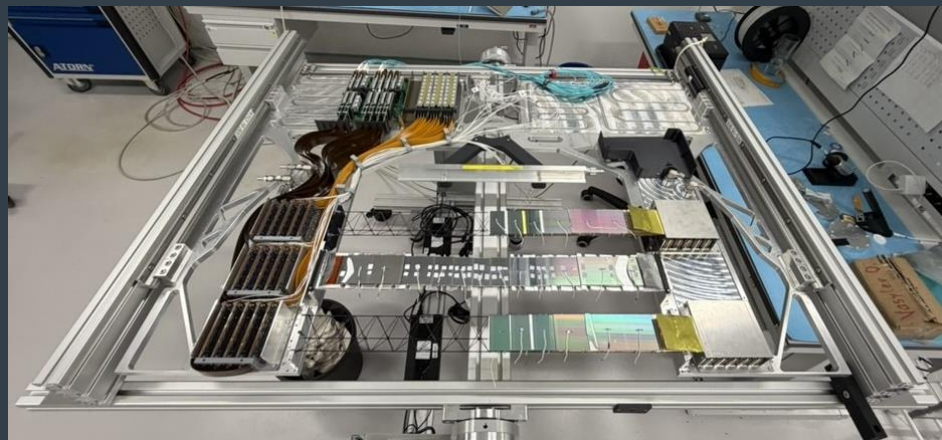
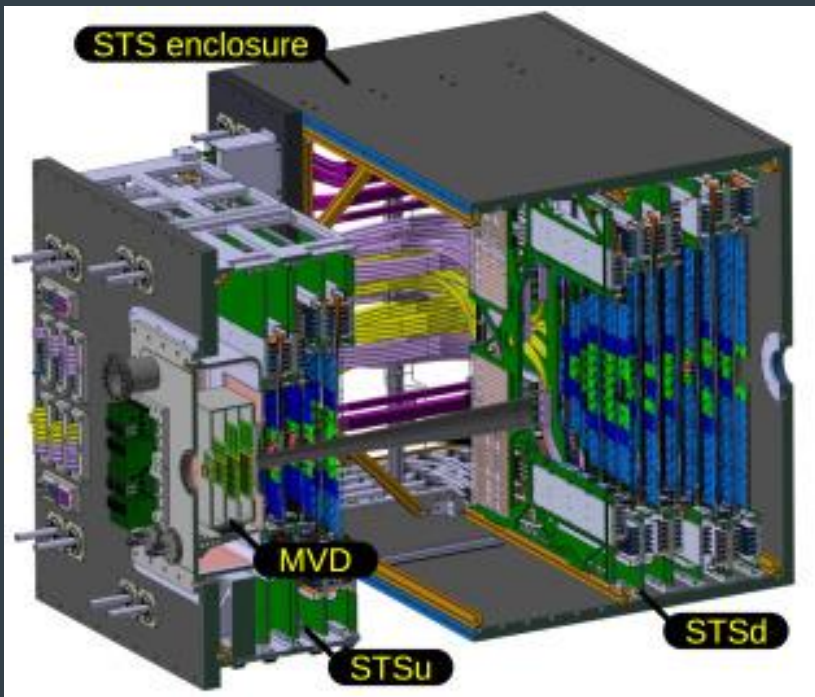


Status and Next Steps

- Code is working!
- And... data makes sense
- Lots more analysis to make...
- Development of Transport QA software
- I will remain involved from far away

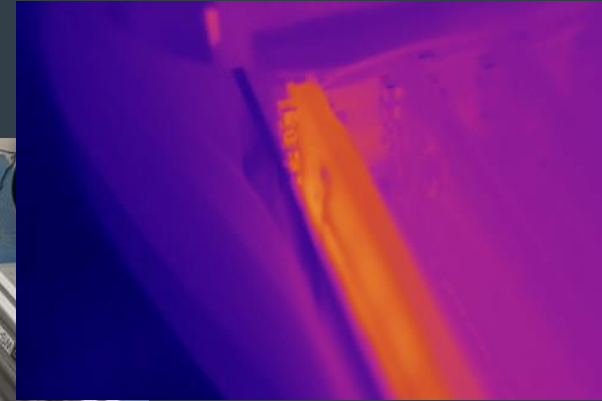
STS Work with Electronics

Half-unit...





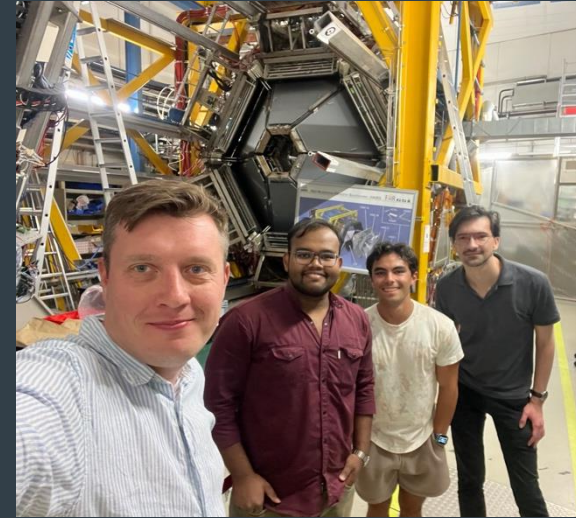
My work





Special Thanks...

Sadly, no picture with Frederic, but biggest thanks to him.



Thank you!

Any questions...?

