Simulation Study of Quasi-Elastic dd Collisions at the SPD

Amaresh Datta (amaresh@jinr.ru)

DLNP
Dubna, Russia

Mar 11, 2025

Primary Interest And Challenges

- Spin dependent effects in quasi-elastic polarized (vector and tensor) deuteron collisions - proposal from Yuriy Uzikov
- Process of interest $d + d \longrightarrow n + p + d$, with a spectator neutron and proton and target deuteron colliding in elastic process
- Detection signature : proton and deuteron tracks detected (no PID), small angle neutrons detected in ZDC and no other tracks or energy depositions
- Neutron momenta and mass can be calculated from detected p, d
- Typical event generators (Pythia, FTF) do not include such processes

Simulation Strategy - Attempt 1

- Use ROOT class TGenPhaseSpace to generate allowed distributions of final state $(n + p + d)$ starting from given invariant energy (of colliding $d + d$)
- Accept event according to matrix element/probability of the generated event
- Yuriy Uzikov provided the calculations of event probability in Glauber model - I am implementing them in the SpdRoot structure

Work on Generator Part

- In 'development' branch, there is now a new class 'SpdQuasiElpdGenerator'
- It uses TGenPhaseSpace
- Technical issues faced : there is a disconnect between Geant4 and FairRoot based databases : i.e. can not identify deuteron by mass in pdgdb in SpdRoot - solution, manually attach pdg PID (1000010020) to particle of relevant mass
- Sample usage of this generator in simu.C script :

```
SpdPrimaryGenerator* primGen = new SpdPrimaryGenerator();

SpdQuasiElpdGenerator *quelpdgen = new SpdQuasiElpdGenerator();
quelpdgen->SetEnergy(10.);
quelpdgen->SetSeed(seed);
primGen->AddGenerator(quelpdgen);
```

Integrating Event Probability : Step 1

- Calculations are done in Fortran code
- To integrate in SpdRoot, need to convert to C/C++ and modify code to create function to be called by other codes
- used a VERY helpful tool (decades old) developed by Bell Laboratory : f2c (www.netlib.org)
- It creates a .c code from .f fortran code that can be compiled with gcc and run

Integrating Event Probability : Step 2

- Can not integrate a stand-alone executable with event-by-event simulation
- Solution : create function that returns required quantity (matrix element in this case) $\longrightarrow$ compile to create a shared library/object (.so file) that can be loaded and functions called form anywhere in SpdRoot
- Requires extra libraries (F, C++ compiler related - provided by netlib.org) required for the compilation stage
- Final library (.so) is not portable as it depends on compiler
- Ideally to be done as part of SpdRoot - extra f2c libraries need to be added as external package in SpdRoot and added to makefile for compilation in future - for now, compilation done on computing local area (lxui, lxpub)

Problem With The Technique

- EXTREMELY inefficient! TGenPhaseSpce generates all possible events and the total possible phase space is huge compared to the events we are interested in
- Accepts only one in a few hundred million events generated
- Also (probably because of the ad hoc it is done) - call to functions from Yuriy's converted/translated code library is very slow
- Forced to switch to a new approach - do not depend on TGenPhaseSpace - manually generate events in a narrow phase space using probabilities from Yuriy's code

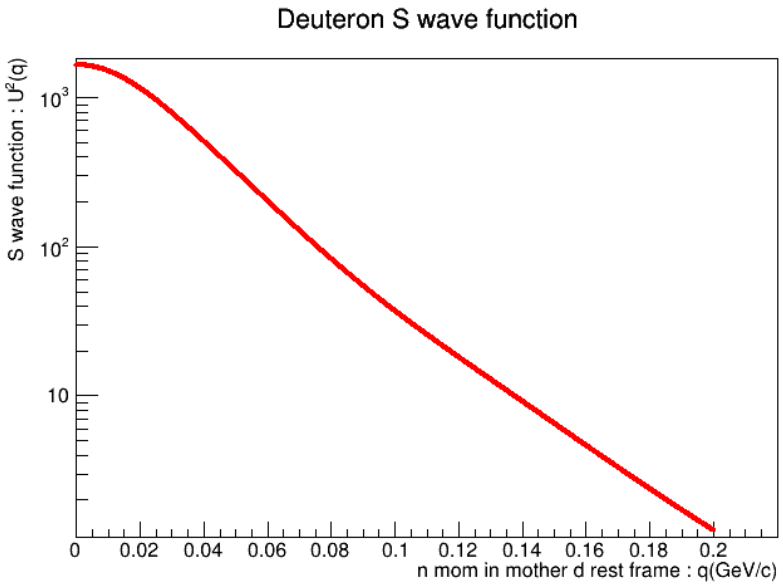
Simulation Strategy - Attempt 2

- Start from 4-momenta of two collding deuterons
- Generate spectator n in the rest frame of a deuteron ($p_z > 0$ for convenience) according to d S-wave function - boost it to lab frame (also the d+d COM frame)
- Accept event for small p_T^n (so they reach ZDC)
- Remaining p and d go through elastic scattering (4-mom conservation applied)
- Go to COM frame of $p + d$, calculate scattering probablity, accept event accordingly
- Boost them back to lab/dd-COM frame - now we have a full event with final state $n + p + d$

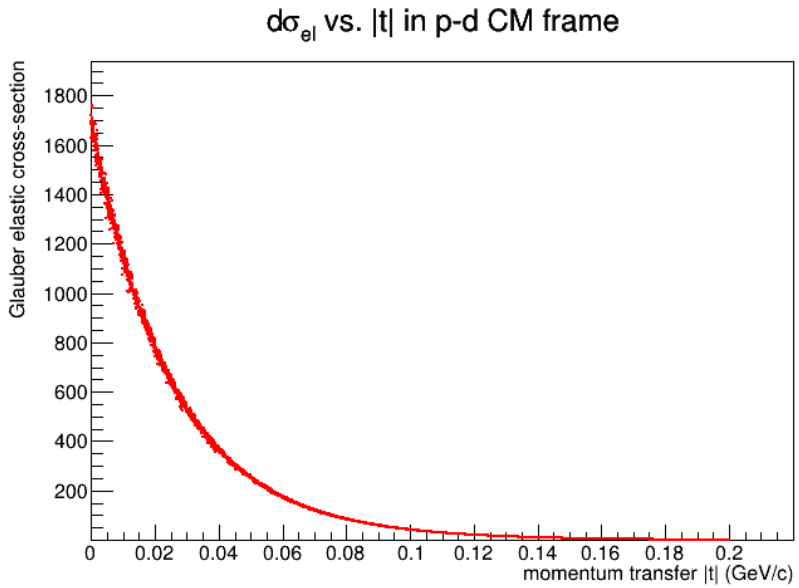
Simulation Strategy - Attempt 2

- This method is very efficient - standalone root macro can generate 1 Million events in 40 seconds
- To use these generated events in SpdRoot environment - using a dummy generator class that will read the results (4-momenta of final state n,p,d) from text file and pass it on to SpdPrimaryGenerator in simu script
- Reconstruction, analysis etc. will follow normally
- Sample features of generated events shown below : $\sqrt{s_{dd}} = 12.6$ GeV ($\sqrt{s_{NN}} = 6.3$ GeV)

Probability Distributions Used for Generating Events

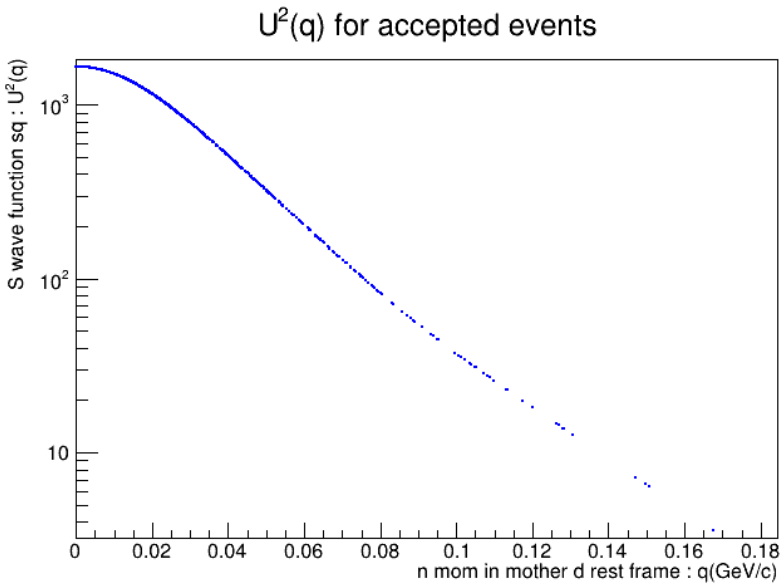


S-wave function squared as a function of p_n in mother rest frame

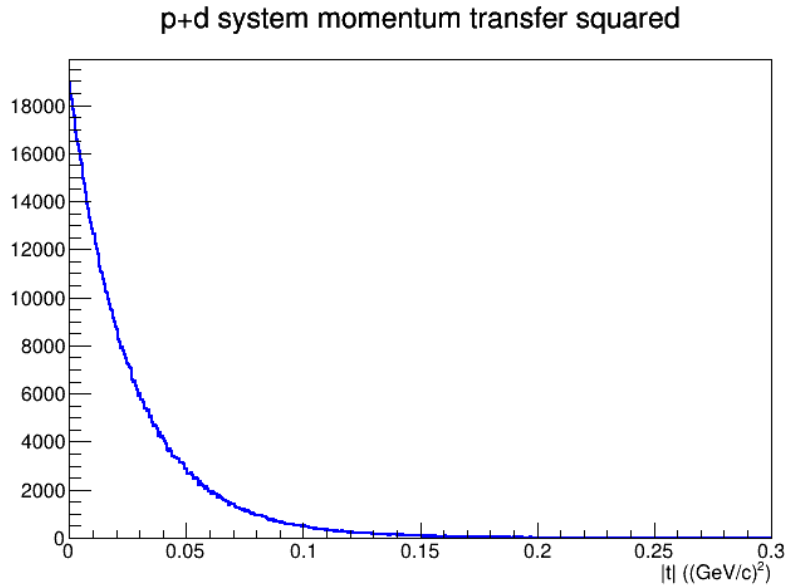


Elastic cross-section as a function of momentum transfer $|t|$ in the $p + d$ COM frame

Probability Distributions of Accepted Events

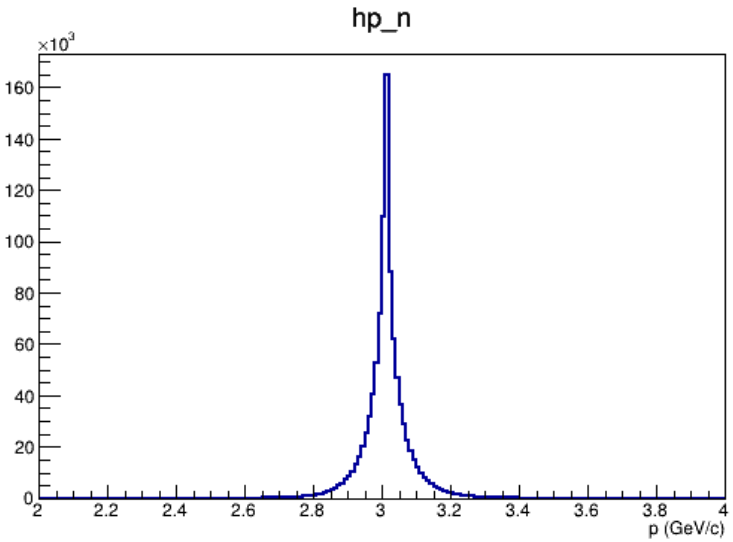


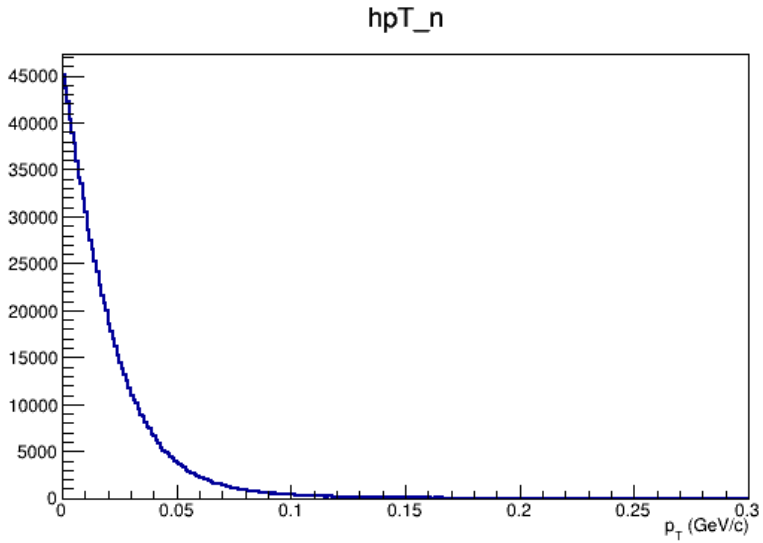
S-wave function squared as function of p^n in mother rest frame

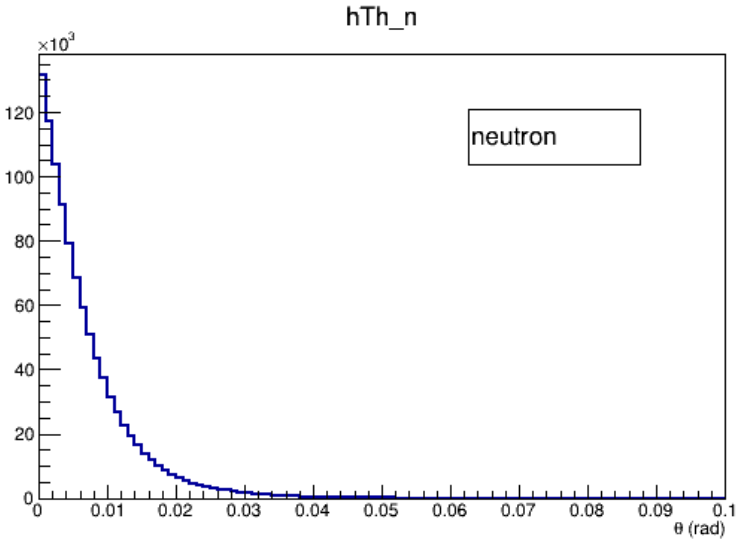


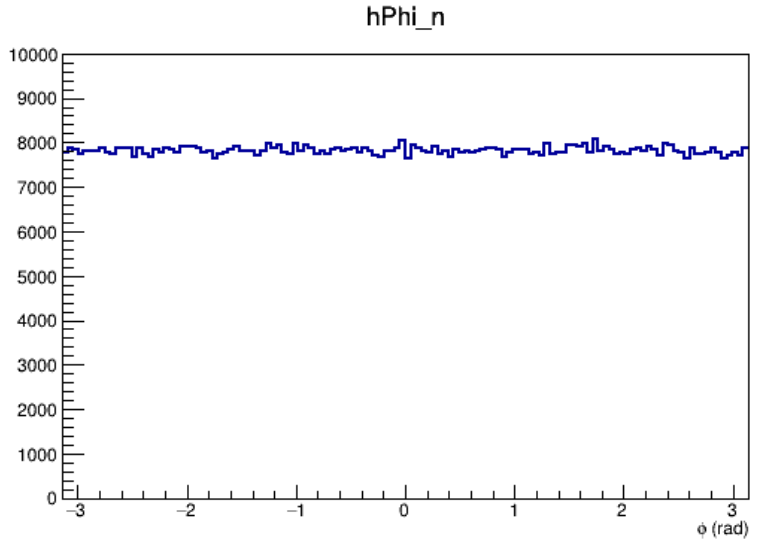
Distribution of momentum transfer $|t|$ in $p + d$ COM frame

Kinematic Distributions of Generating Neutrons

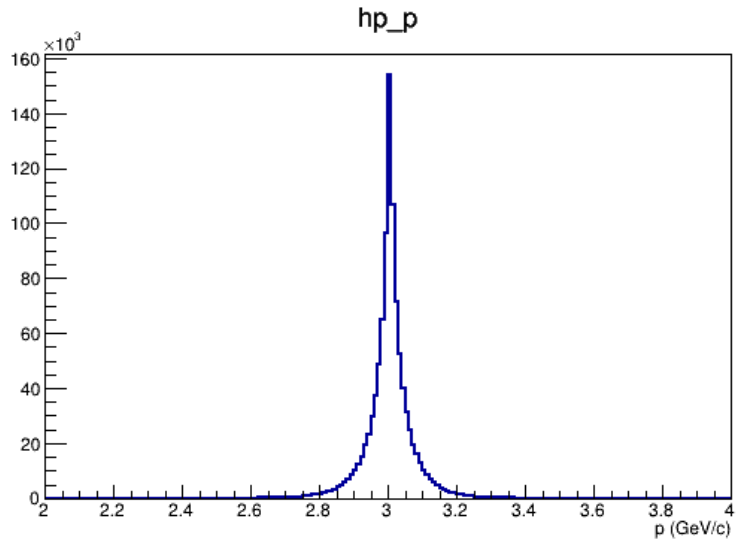


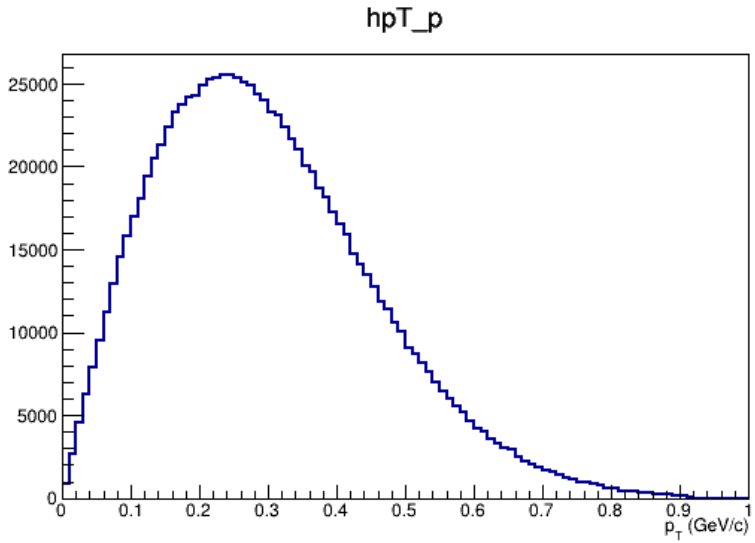


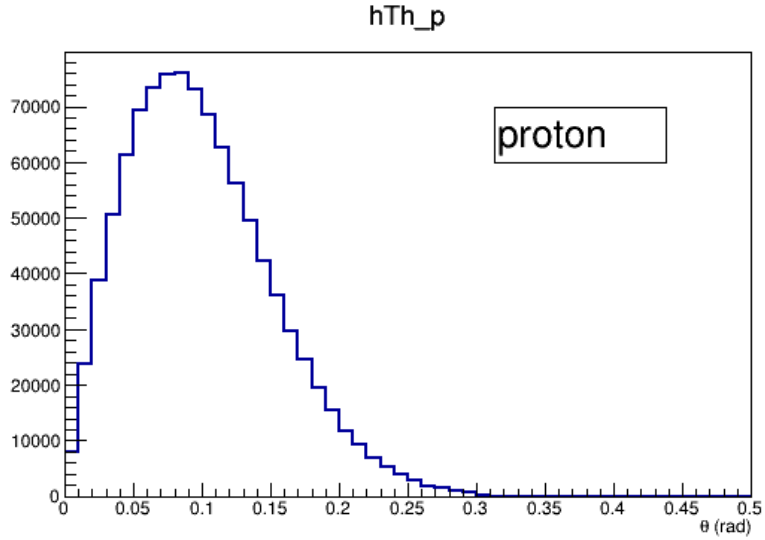


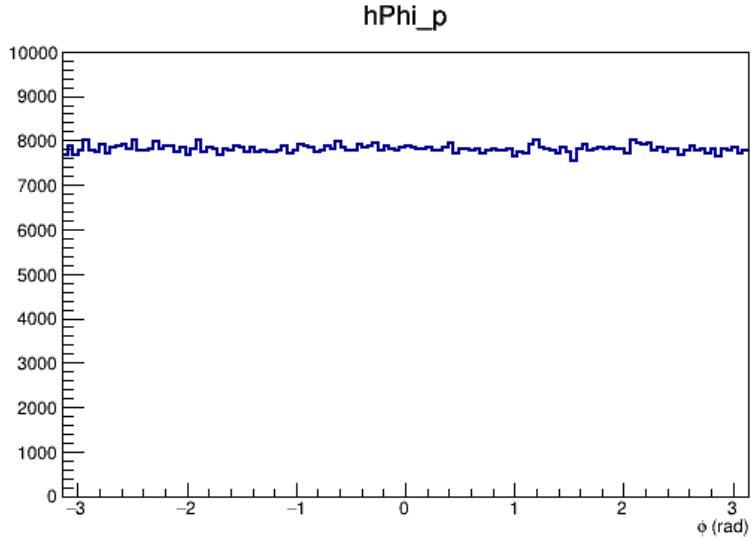


Kinematic Distributions of Generating Protons

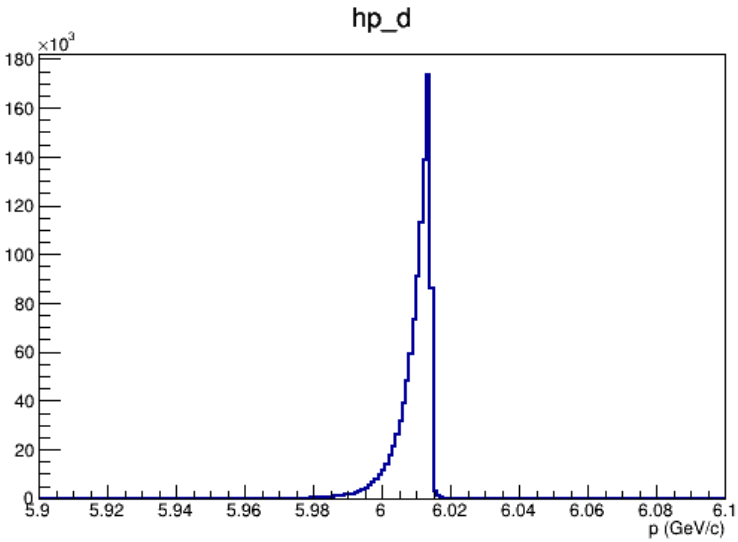


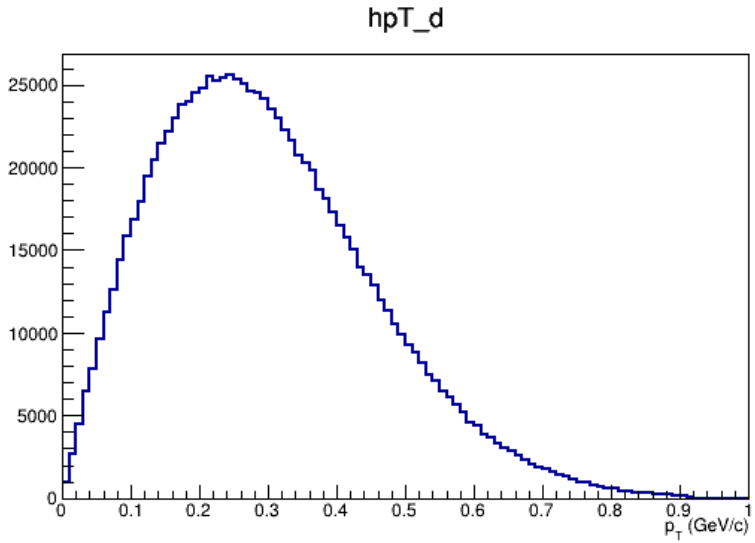


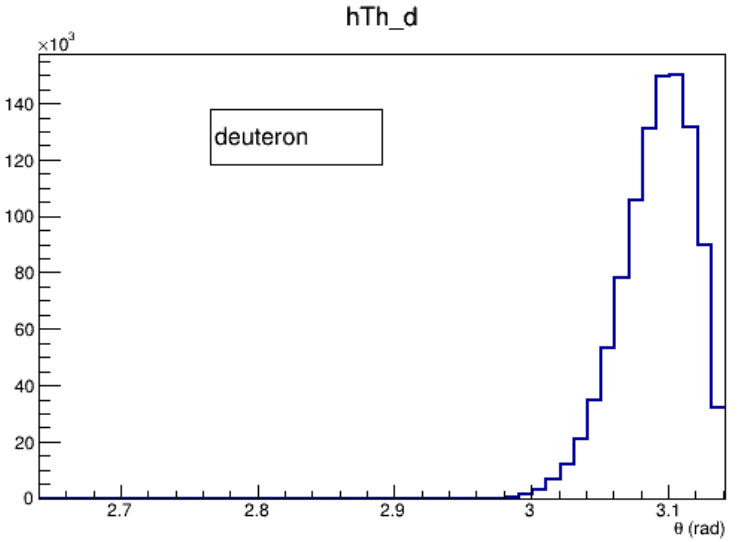


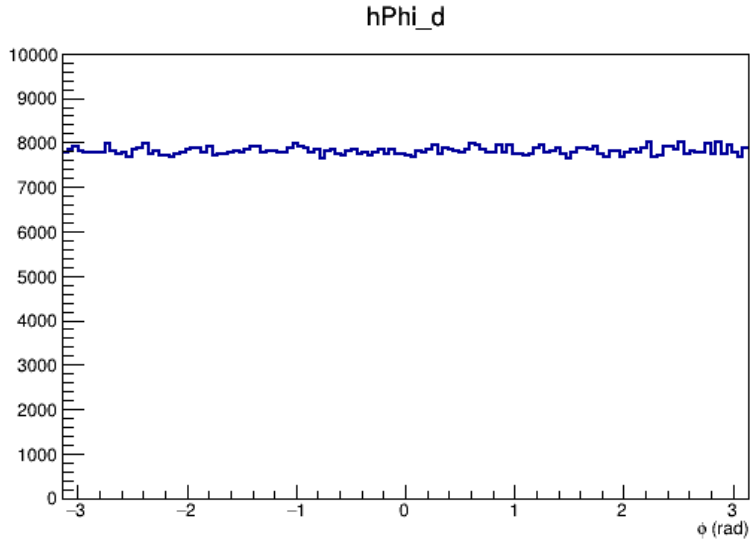


Kinematic Distributions of Generating Deuterons









Summary and To Do List

- We have an efficient method to generate events in the narrow range of phase space we are interested in
- In the process of integrating it with SpdRoot reconstruction system
- Once we have reconstruction efficiency, we have to (carefully) estimate what fraction is our (very specific) generated events out of $d + d$ total cross-section
- That will give us estimate of statistical precision of signal
- Have to think how to estimate background - may be Pythia8 or FTF to generate d+d inelastic
- Do the entire chain for a few different collision energies

Thank You