1. QCD a CBM-ben

A kvantumszíndinamika elemi részecskéi a kvarkok és antikvarkok, amelyek a gluonokon keresztül hatnak kölcsön. Ezek mindegyike színtöltést hordoz, ahol a szín is kvantumszám. A gluonoknak 8 fajtája van, hogy az összes színátmenetet le lehessen velük írni. A gluonok önmagukkal is kölcsön tudnak hatni.

A CBM detektor elsődleges célja, hogy a barionikus anyag fázisátalakulásait vizsgálja. A fázisdiagram pontos feltérképezése elengedhetetlen ahhoz, hogy az arról alkotott elméleteket alátámasszuk.

A kvark-gluon plazma a barionikus anyag egyik fázisa. Ez az állapot jelen volt a Nagy Bummnál (kis sűrűség, extrém hőmérséklet) valamint minden valószínűség szerint a neutron csillagok magjában is megtalálható (kis hőmérséklet, extrém sűrűség). A kvark-gluon plazma földi megfigyelésének egyetlen lehetőségét a nagy energiás részecskegyorsítók biztosítják.

Ehhez szükséges gyorsítók és detektorok több helyen is léteznek a világon, RHIC, LHC, de a FAIR projekt sajátossága, hogy ezeknél magasabb barionsűrűséget akar elérni, valamint pont olyan enegián dolgozik majd, hogy képes legyen az elsőrendű fázisátlakulás vizsgálatára, valamint a kritikus pont környékének feltérképezésére.

Az *up* és *down* kvarkok tömege nagy energiás közelítésben nagyjából nulla, ezért szokták azt mondani, hogy a QCD-nek viszonylagos királis szimmetriája van. Ez alacsony hőmérsékleteken (és sűrűségeken sérülhet), ezt nevezzük királis szimmetriasértésnek. Egy-egy ütközés során nagyjából 10^{-22}s-ig tartó állapotra koncentrálunk, ekkor van jelen ugyanis kvark-gluon plazma. A jelenség tranzienssége miatt természetesen csak a melléktermékek, a részecskezápor vizsgálható. [fázisdiagramok: 2]

2. A CBM detektor

A detektor elrendezése balról jobbra haladva a következő (ábra):

- CBM szupravezető mágnes szilícium spektrométerrel
- a micro vertex detektor (MVD) az előbbi belsejében
- a szilícium követő rendszer (STS) is
- Cserenkov-detektor (RICH - ring imaging Cherenkov detector - világos kék)
- müon sprektrométer (fekete)
- ezt követi 4 réteg átmeneti sugárzás (TRD - transition radiation detector) detektor
- és egy time-of-flight (TOF) fal
- a fő detektorok után található még egy célfigyelő detektor (PSD)

A szilícium követő rendszer feladata az, hogy rekonstruálja majd a részecskék trajektóriáit. Csak töltött részecskék észlelésére képes, de képes mérni a töltés nagyságát és az impulzust is tud mérni. A TOF fal igen nagy felbontást tud elérni, nagyjából 60 ps-os felbontásra is lehetőség van.

A CBM projekt még egyenlőre csak terv szintjén létezik, a szimulációt folyamatosan fejlesztik. Jövőre, vagy legkésőbb 2019-re már tervben van egy miniCBM detektor építése az esetleg később felmerülő tervezési, kivitelezési problémák elkerülésére. A FAIR létesítmény építése idén nyáron kezdődött és az első részecskenyaláb 2022-ben várható. A miniCBM projekt a meglévő GSI gyorsítónál fog tevékenykedni az

addig fennmaradó időben, ahol megpróbálják a számítógépfarmot tökéletesíteni, hogy az adatokat minél gyorsabban feldolgozhassák.

A FAIR tudósai kifejlesztettek egy több tízezer soros szimulációt, ami a ROOT-on alapszik. Ezt ők cbmROOT-nak hívják, mivel teljes egészében a CBM-hez igazodik és ingyenesen elérhető bárki számára. Sok jól ismert nehézion szimulációs eljárást használnak, amik a CBM környezetre vannak szabva, úgy mint: UrQMD (Ultra Relativistic Quantum Molecular Dynamics), valamint PHSD (Parton Hadron String Dynamics). Ezek a szimulációs kódok széles körben használtak nem csak itt, hanem az egész nehézion fizika területén. [detektor: 1]

3. Mini projekt: Φ-mezon

A CBM detektor egy általános célú nehézion mérési eszköz lesz, hogy az erősen kölcsönható anyag fázisdiagramját vizsgálni lehessen. A rezonanciák nagyon fontosak, hogy a sűrű anyagot vizsgálni tudjuk az ütközés során. Az ilyen rezonanciák egyike ami fontos a CBM és a fázisdiagram vizsgálatának szempontjából pedig a Φ-mezon, aminek nagyon kicsi a hadronokra vett hatáskeresztmetszete így eléggé valószínűtlen, hogy kölcsönhat a nagy mennyiségű hadronnal, ami a reakció során keletkezik, vagyis jó indikátora a sűrű, kezdeti eseménynek. A Φ-mezon egy *strange* és egy $anti - strange$ kvarkot tartalmaz és a kulcsa lehet az s kvark partonikus anyagban lévő keletkezésére. A Φ-mezon K^+, K^- párokra bomlik nagyjából 50%-os eséllyel és egyebekre (pl. dileptonokra is). A közepes élettartama nagyjából $1.55 \cdot 10^{-22}$ s tehát még a TOF falat sem éri el, csak a bomlástermékei lesznek detektálva már jóval azelőtt is. A tömege 1.019 MeV ami a kaonok invariáns tömegével kifejezve egy rezonancia csúcsként látható az ütközés/szimuláció után kinyert adatok között. Ahol a kaonok invariáns tömege:

$$M_{KK} = \sqrt{(E_1 + E_2)^2 - (\underline{p}_1 + \underline{p}_2)^2}$$

Én a PHSD adatait vizsgáltam, amin lefuttattam a CBM szimulációt. Egy Au+Au centrális ütközést vizsgáltam 10 GeV-es bombázó energián. A CBM szimuláció kimenetét a cbmROOT-tal rekonstruáltam. Több mint 5 millió esemény szerepelt a kezdeti *.root* fájlban amit a szimulációhoz használtam.

A hisztogramokon az x-tengelyen a kaon párok invariáns tömege szerepel, az y-tengelyen pedig az adott energián a 'beütések' száma. Egy apró kiugrás látható a nagy kombinatorikus háttéren nagyjából 1.02 GeV környékén ami pontosan a Φ-mezonok bomlásából adódik.

Igen nehéz feladat lesz hatékonyan detektálni a Φ-mezonokat a CBM detektorrendszerrel. A részecskék nem csak rövid életűek, de egy hatalmas háttér is nehezíti az apró csúcs megtalálását. Ezért is kell hatalmas számú eseményt vizsgálni, hogy a csúcs a statisztikában már látható legyen. Ennek ellenére határozottan mondhatjuk, hogy a CBM detektor képes lesz a Φ-mezonok detektálására és ezáltal a strange kvark termelődésének megértésére az erősen kölcsönható anyagban.

A szimuláció hatékonysági mutatókat is biztosít. Mindezeket különböző részecske impulzusok esetén. A jelzett detektálás hatékonysági értékek nem túl magasak, de eléggé stabilak adott tartományokban az észleléshez.

4. A szimulációs program

Maga az ütközés a UrQMD és a PHSD programok segítségével játszódik le, a CBM szimuláció a detektor választ szimulálja, tehát az ezekből származó adatokat kapja meg kezdeti paraméternek. Ezek a modellek az ALICE, RHIC és LHC detektornak, valamint nem utolsó sorban a CBM detektornak lettek fejlesztve.

Az első lépés az, hogy le kell futtatni egy Monte Carlo szimulációt, hogy képeset legyünk a 'valódi' adatokat összepárosítani a keltett eseményekkel. A program ezen része arra lett tervezve, hogy kiszűrje a találatokat a detektor anyagban és olyan pontokat találjon, amelyek később trajektóriákká összeállíthatók.

A program a Geant3 (főként) és Geant4 programokat használja, hogy a részecskék anyagon való áthaladását szimulálja. Ez is a Monte Carlo szimuláció része.

Az első makró kimenetén tehát egy szimulációs fájl van, ami az STS és az MVD detektorok által detektált találatokat tartalmazza valamint a TOF fal és egyéb detektorok adatait is. Ezeket felhasználva lép a program a második fázisba, a rekonstrukció részhez. A rekonstrukciós kód először is klasztereket próbál találni az MVD detektorban, hogy megtalálja, hogy hol volt az ütközés/ütközések kiinduló pontja. Ha ezt megtalálta továbbhalad és megpróbálja lekövetni a részecske pályákat. A töltött részecskék körpályára állnak az erős mágneses tér hatására így a pontokra köríveket próbálnak illeszteni és a legjobb illesztéssel bírókat fogadják csak el (van egy százalékos határ, ami alatt hibás detektálásnak ítélik).

Nyilvánvalóan, a találatok és a pályákat többször próbálja meg a program helyesen megtalálni, azért, hogy elkerülje a hibákat. Kisebb az esélye így a hibás találatnak, vagy a hibásan illesztett trajektóriának. Ennék része a digitalizáció, ami lényegében azt jelenti, hogy a szimulációs program megpróbálja a detektor választ is számításba venni. Vegyük például az STS detektort. Ennek egy szálas, hálós elrendezése van, amikor egy részecske áthalad, akkor több szálban is detektáljuk, ezek metszéspontjában van a tényleges helye. De ha egyszerre két részecske ment át 'ugyan azon a ponton', akkor ezt nem láthatjuk, később a pályák illesztésénél probléma lehet. Ezért is van az, hogy ha az STS detektor több, mint 5%-a detektál, akkor a rendszer lényegében nem mér, nem szerez kiértékelhető adatokat.

A sikeres rekonstrukció után, ami a nyers adatokból létrehozta végső soron a trajektóriákat az egyetlen visszamaradó feladat a részecske felismeres és ezek pályákhoz való párosítása. Erre egy robusztus és hatékony program áll rendelkezésre, aminek a neve KFParticleFinder.

Ennek a programnak a kimenete egy .root fájl, ami rengeteg részecskét és hozzájuk tartozó adatot tartalmaz, detektálási hatékonyságról, háttérről, armenteros diagramokkal, bemenő és kimenő jelekkel.

Ahogy korábban említettem először a Monte Carlo szimulációt kell használni valamilyen bemeneti fájllal. Ez egy .root fájl vagy egy egyszerű ASCII fájl is lehet, a szimulációs kód képes mindkettő fogadására. Egy ilyen fájlban részecske ID-k és impulzusuk található. A kimenete a PHSD és a UrQMD szimulációknak általában egy .root fájl, de például a HIJING sima szöveges kimenetet produkál. A CBM szimulációnál különböző függvények teszik lehetővé mindkét adattípus feldolgozását.

Tehát a rekonstrukció után, valamint a részecske felismerés végeztével, ha bekapcsoltuk a vizualizációt képesek vagyunk vizualizálni az eseményeket. Ehhez az *eventDisplay.C* makrót kell futtatnunk. Ez a makró az egész CBM geometriát tartalmazza, tehát az egész detektort átláthatjuk vele. Megjeleníthető benne az összes trajektória és a részecskék.

5. Nehézion fizika itthon - Részecske klaszterezés - MST (/ BFS)

A Wigner Fizikai Kutatóközpontban dolgozó témavezetőmtől, Wolf Györgytől azt a feladatot kaptam, hogy az általa írt nehézion reakciós programhoz írjak egy klaszterező programot. Ez a szimuláció a korább-

ban említett, PHSD és UrQMD modellekhez hasonló, hazai fejlesztésű projekt. A hadron-mag és mag-mag reakciókat transzport-egyenletek segítségével vizsgálva, a BUU-modell (Boltzmann-Uehling-Uhlenbeck modell) felhasználásával egy időfüggő, részecskék kölcsönhatását figyelembe vevő modell segítségével szimulálja ez a program.

Ennek kimenetén többek között szerepelhetnek bizonyos részecskék és azok momentum- és térbeli eloszlása. Detektortól függően máshogy lehet ezeket mérni. Ha olyan detektorunk van, ami csak töltött részecskéket mér, és a töltés nagyságát nem, akkor figyelembe kell vennünk, ha például térbeli (vagy impulzustérbeli) közelség miatt csak egy beütést kapunk. Így az én programom pontosan arra képes, hogy euklideszi-térben (vagy impulzustérben) klasztereket keres. Így a beütésszámra pontosabb jóslatot lehet majd adni tényleges detektor környezetben.

Továbbá a CBM kísérletben kutatni fogjak az egyszeres és kétszeres ritka magokat, amihez szintén szükséges a koaleszcencia szimulációja.

Nem tökéletesítettem még a programot, ha későbbiekben erre igény van természetesen fejlesztem. Egyenlőre hely- és impulzus-koordinátákat olvas be, majd ezután próbálja meg klaszterezni a részecskéket. A klaszterezéshez nem a legjobban ismert klaszterező algoritmust használtam hanem az úgynevezett minimális feszítő fa (vagy MST (Minimal Spanning Tree) a későbbiekben) algoritmust. Ez egy gráfban a lehető legrövidebb utat találja meg. Két pont akkor van összekötve a gráfban, ha egy adott minimum távolságnál közelebb vannak. Természetesen ez a minimális távolság is a bemenetről állítható. Egy részecske egy klaszter része, ha legalább az egyik részecskéhez a klaszterben kellően közel van.

Ennek az algoritmusnak talán az a legnagyobb előnye, hogy nem kell előre feltételezni, hogy hány klaszter van és azt sem, hogy azok vajon hol helyezkedhetnek el. Elméletben az algoritmus hatékonysága $O(\log m \cdot n)$ vagy $O(\log n \cdot n + m)$, ahol n a pontok száma a gráfban, míg m az élek száma. A hatékonyság a használt adatstruktúráktól függ. Ez természetesen Prim algoritmusára igaz, vannak ennél hatékonyabb megoldások is, de számomra ez tűnt a legkényelmesebb, legmegvalósíthatóbb választásnak. Továbbá egy francia kutatócsoport Nantes-ban hasonló nehézion fizikai szimulációjában is ezt az algoritmust javasolják.

Az egyik elméleti nehézség a megvalósítás során az volt, hogy az algoritmus képes legyen több klasztert formálni. Hiszen miután nem tud továbbhaladni egy klaszterben, azaz nem tud több pontot hozzáadni, ki kell venni az adathalmazból a klaszterezett pontokat. Ezután lehet csak választani egy random pontot újra, és lefuttatni az eddigi algoritmust a már redukált gráfon.

A kimeneten tehát egy olyan fájlt kapunk, amiből meghatározható igen könnyen a maximális klaszterméret és az átlagos klaszterméret is. Az fentebbi ábrákat is ezek alapján készítettem egy egyszerű programmal. Ezek mellett még külön fájlokban megkapjuk a klaszterek elemeit is a program korábbi verziójában.

A kód sebessége nagyban függ a definiált maximum távolságtól. Ha az előbbi algoritmust egy igen nagy szám, akkor egy nagy klasztert találunk, viszonylag gyorsan, hiszen az MST keresés hiba nélkül lefut. Azonban minél nagyobb távolságokra is éleket rakunk a kreált gráfba annál több számítást kell végeznünk és a sebesség nagyban leromlik. Lényegében mondható az, hogy a klaszterezés belátható időn belül lefut tetszőlegesen nagy adathalmazra, hacsak a rendszer ki nem fogy a memóriából. 12 ezer sornyi adat klaszterezése esetén ez már megtörténhet. A program több dologra is képes mint amire jelenleg használva van. Az éleket tudja definiálni és menteni is, ez az MST algoritmus sajátossága, de nekem nincs rá szükségem ebben az alkalmazásban. A távolság számítást módosítanom kell majd, hiszen jelenleg csak euklideszi normám van, ami nem a legmegfelelőbb, de ez csak egy belső függvény átírást igényel majd. A kommentált kódrészletek a kimeneti fájlok emberi olvashatóságáért feleltek volna részben, valamint a további funkcionalitást kapcsoltam ki, de az adatok feldolgozása során kényelmesebb volt így eljárnom.