VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AKCELERACE ALGORITMU SLEDOVÁNÍ ČÁSTIC V EXPERIMENTU CBM

ACCELERATION OF PARTICLES TRACKING ON CBM EXPERIMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAEL ROTH**
AUTHOR

VEDOUCÍ PRÁCE **Ing. PETR MUSIL, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Roth Michael**
Program: Informační technologie
Název: **Akcelerace algoritmu sledování částic v experimentu CBM**
Acceleration of Particles Tracking on CBM Experiment
Kategorie: Zpracování signálů

Zadání:

1. Prostudujte zařízení a algoritmy využívané pro detekci a rekonstrukci trajektorie částic v experimentu CBM.
2. Prostudujte možnosti akcelerace algoritmů na hardwarových platformách.
3. Vyberte algoritmus pro sledování částic, prozkoumejte možnosti a vlastnosti jeho implementace a navrhněte metody jeho akcelerace na hardwarové platformě.
4. Implementujte vybraný postup, ověřte jeho vlastnosti na simulovaných nebo reálných datech.
5. Diskutujte dosažené výsledky a možnosti dalšího pokračování práce.

Literatura:

- Valentina Akishina, 4D Event Reconstruction in the CBM Experiment, Doctoral Thesis, Johann Wolfgang Goethe-Universität, 2017 Frankfurt am Main

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Musil Petr, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 29. července 2022
Datum schválení: 31. ledna 2022

Abstrakt

Cílem této práce je prostudovat metody detekce a rekonstrukce trajektorií částic v experimentu CBM a problematiku akcelerace těchto metod na hardwarových platformách. V práci byly srovnány výhody a nevýhody rozšířených metod a pro další studium byla vybrána metoda rekonstrukce na bázi celulárních automatů a Kalmanových filtrů. Práce se podrobně zabývá zejména vývojem simulačního modelu, vhodného pro generování testovacích dat pro usnadnění budoucí implementace vybraného sledovacího algoritmu. Byly vytvořeny dva odlišné simulátory částic, které budou v navazující práci použity pro výpočet predikčního kroku rozšířeného Kalmanova filtru a testování kvality implementované rekonstrukční metody.

Abstract

The focus of this work is to research various methods of particle track reconstruction in the CBM experiment, and the problem of hardware acceleration of these methods. The advantages and disadvantages of the extended methods were discussed and a reconstruction method based on cellular automata and Kalman filters was selected for further study. In particular, the thesis details the development of a simulation model suitable for generating test data to facilitate future implementation of the selected tracking algorithm. Two different particle simulators have been developed and will be used in the following work to calculate the prediction step of the extended Kalman filter and to test the quality of the implemented reconstruction method.

Klíčová slova

Experiment CBM, akcelerace na hardwarových platformách, Kalmanův Filtr, rozšířený Kalmanův filtr, celulární automaty, modelování a simulace, simulátor částic v magnetickém poli, částicová fyzika, kvark-gluonové plazma, kvantová chromodynamika, fyzika raného vesmíru, HEP experiment

Keywords

CBM experiment, hardware acceleration, Kalman Filter, Extended Kalman Filter, Cellular automata, modelling and simulation, particle simulators in magnetic field, High energy physics, quark-gluon plasma, quantum chromodynamics, physics of early universe, HEP experiment

Citace

ROTH, Michael. *Akcelerace algoritmu sledování částic v experimentu CBM*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Musil, Ph.D.

Akcelerace algoritmu sledování částic v experimentu CBM

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Musila, Ph.D. Další informace mi poskytli prof. Dr. Ivan Kisel a Mgr. Jan Jíša. Prohlašuji, že jsem uvedl všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michael Roth
29. července 2022

Poděkování

Děkuji panu doktoru Petru Musilovi za odborné vedení a ochotu pomoci ve všech fázích práce. Dále děkuji panu profesoru Ivanu Kiselovi za poskytnutí inspirace a pomoc při formulaci problému, ze kterého se nakonec vzniklo zadání práce. Nakonec děkuji panu Mgr. Janu Jíšovi, za poskytnutí nedocenitelné odborné pomoci a konzultace v oblasti fyziky.

Obsah

Kapitola 1

Úvod

Teorie velkého třesku, nejpopulárnější kosmologická teorie o vzniku a prvních okamžicích existence našeho vesmíru tvrdí, že vesmír, jak ho známe vznikl mohutnou „explozí“ a po miliardy let postupně chladl a rozpínal se do podoby, kterou známe dnes. Přestože současné teorie odhadují čas od velkého třesku na přibližně 13,8 miliardy let, o tom, jak vypadal náš vesmír v prvních okamžicích od svého vzniku víme jen velmi málo a přesný průběh velkého třesku nám není znám. Předpokládáme, že velký třesk probíhal ve fázích, kdy se jeho podoba drasticky lišila od podoby, jakou má dnes. Mezi prvními z těchto fází (řádově mikrosekundy od velkého třesku) byla pravděpodobně fáze takzvaného **kvark-gluonového plazmatu**[1], kdy veškerá hmota v mladém vesmíru byla natolik hustá a horká, že se jednotlivé částice hmoty mohly volně a samostatně pohybovat (jev, který například u kvarků za normálních okolností nelze sledovat), aniž by je jakákoliv síla v mladém vesmíru dokázala spoutat. Předpokládáme, že stav hmoty s podobnými vlastnostmi dnes můžeme ve vesmíru sledovat pouze za extrémních podmínek, například v jádrech neutronových hvězd.

Kvantová chromodynamika[2] je disciplína teoretické fyziky, zabývající se silnou interakcí mezi **barevně nabitými částicemi**, tedy interakcí mezi kvarky, zprostředkovávanou částicí zvanou *gluon*, podobně jako je elektromagnetická interakce zprostředkovávána *fotony*. Barevně nabité částice jsou specifické tím, že je za „normálních“ teplot a tlaků nelze pozorovat samostatně: vždy se vyskytují po trojicích (hadrony, například *proton* a *neutron*) nebo po dvojicích (mezony, například *pion*). To je způsobeno jevem zvaným *barevné uvěznění*[3], který mimo jiné říká, že dodáme-li kvarkům dostatek energie na to, aby se dokázali rozdělit, je energeticky výhodnější vytvořit nový pár kvark-antikvark. V případě, že stlačíme hmotu do tak extrémních tlaků, že jednotlivé *nukleony* nemají dost prostoru k tomu, aby mohly samostatně existovat, přestávají být kvarky drženy pohromadě gluony a mohou se volně pohybovat. Takové fázi hmoty říkáme *kvark-gluonové plazma*.

Jeden ze způsobů, jak můžeme laboratorně pozorovat a studovat strukturu hmoty za extrémních podmínek je pomocí řízených kolizí těžkých iontů s vysokou energií, jako jsou například jádra zlata či olova. Taková jádra urychlujeme na rychlosti velmi blízké rychlosti světla a necháváme s obrovskou energií kolidovat s terči detektorů, čímž jádra přivedeme k přechodnému stavu s extrémně vysokou hustotou a teplotou, který fyzikové nazývají *fireball* (ohnivá koule). Aby bylo možné vytvořit vhodné podmínky pro zkoumání hmoty za tak extrémních podmínek, musíme použít urychlovače s velmi vysokou energií paprsku. Experiment CBM, připravovaný experiment při moderním výzkumném středisku FAIR v

[1] z anglického QGP – Quark-Gluon Plasma
[2] z anglického QCD – Quantum ChromoDynamics
[3] z anglického *colour confinement*

německém Darmstadtu, bude hrát důležitou roli při zkoumání fázového diagramu kvantové chromodynamiky. Schopnost sledovat takto extrémní stav hmoty je ale vykoupena obrovským množstvím experimentálních dat (cca. 1 TB/s), které je potřeba zpracovávat *on-line*.

Rychlé zpracování velkého objemu hrubých experimentálních dat je netriviální problém, jehož vhodné řešení je stěžejní pro úspěšnost fyzikálního experimentu. Ve svých pracích se tímto problémem zabývali například prof. Dr. Ivan Kisel a Dr. Valentina Akishina, kteří se zabývali vývojem algoritmu rekonstrukce trajektorií částic pro potřeby detektoru CBM a jeho možnostmi jeho akcelerace, v tomto případě pro platformu *Intel Xeon Phi*. Tato práce si klade za cíl prozkoumat alternativní způsoby akcelerace rekonstrukce trajektorií pro experiment CBM, zejména pak v kontextu programovatelných hradlových polí (FPGA). Kapitola 2 poskytuje velmi stručný úvod do výzkumného střediska FAIR a experimentu CBM, kapitola 3 se poté dopodrobna zabývá jednotlivými algoritmy, použitými pro rekonstrukci trajektorií a aproximaci vlastností jejich částic a jejich zhodnocením proveditelnosti jejich akcelerace na hardwarových platformách. Kapitola 4 se nakonec zabývá vývojem vhodného simulátoru pohybu částic uvnitř detektoru CBM a jeho použitím v rámci vlastního algoritmu sledování částic.

Kapitola 2

Experiment CBM a výzkumné středisko FAIR

Tato kapitola je věnována stručnému popisu pracoviště FAIR[1] a jeho výpočetní infrastruktury, fyzikálního experimentu CBM[2] a jeho cílů. Kapitola se velmi stručně zabývá detektorem STS[3] a supravodivým dipólovým magnetem, ve kterém se STS nachází. Zvláštní pozornost je pak věnována výpočetní farmě při GSI **Green IT Cube**.

2.1 Pracoviště FAIR

Facility for Antiproton and Ion Research (FAIR) je mezinárodní výzkumné centrum, v současné době ve výstavbě v německém Darmstadtu při Helmholtzově centru pro výzkum těžkých iontů GSI[4]. Projekt je realizován partnery z Finska, Francie, Indie, Německa, Polska, Rumunska, Ruska, Slovinska, Švédska a Velké Británie. Česká republika se stala aspirujícím partnerem v roce 2018. [19, 12] FAIR se v budoucnosti stane jedním z největších a nejsložitějších urychlovačů na světě. Výsadou urychlovače bude jeho schopnost produkovat svazky částic všech prvků periodické tabulky (nebo jejich iontů), případně pak svazky antiprotonů, urychlené až na 99% rychlosti světla. [11]

Stávající infrastruktura GSI

Stávající zařízení GSI se v budoucnu stanou součástí infrastruktury FAIR a jeho dva urychlovače budou sloužit jako první urychlovací stupeň pro připravované experimenty. Lineární urychlovač UNILAC[5], dlouhý 120 metrů, urychluje částice až na 20% rychlosti světla. Odsud částice putují k prvním experimentům a prvnímu prstencovému urychlovači SIS18[6] o obvodu 120 metrů, který částice dále urychluje až na 90% rychlosti světla a dále vysílá ke stávajícím experimentům GSI a úložným prstencům. [11]

[1] FAIR – Facility for Antiproton and Ion Research (středisko pro výzkum antiprotonů a iontů)
[2] CBM – Compressed Baryonic Matter (stlačená baryonová hmota)
[3] STS – Silicon Tracking System (křemíkový sledovací systém)
[4] GSI – Gesellschaft für Schwerionenforschung (sdružení pro výzkum těžkých iontů)
[5] UNILAC – UNIversal Linear ACcelerator (univerzální lineární urychlovač)
[6] SIS – Schwerionensynchrotron (synchrotron pro těžké ionty)

Obrázek 2.1: Rozvržení objektu FAIR (vlevo). Stávající urychlovače GSI UNILAC a SIS18 (vyobrazené modře) budou kromě obsluhy stávajícího komplexu GSI (vyobrazeno šedě) sloužit jako injektor pro dva velké synchrotrony SIS100 a SIS300 (červeně). Částicové svazky budou rozváděny do různých experimentálních stanic nebo k terčovým stanovištím pro experimenty vyžadující svazky antiprotonů nebo radioaktivních izotopů. Převzato z [3]. Fotografie areálu GSI a staveniště FAIR (vpravo), převzatá z [21], pořízená v dubnu 2021.

Prstencové urychlovače SIS100 a SIS300

FAIR bude pracovat s částicovými svazky s vysokými intenzitami iontů (10^9 částic za sekundu v případě iontů zlata) nebo pozitronů (10^{13} částic za sekundu). [1] V současné době jsou ve výstavbě nové prstencové urychlovače SIS100 a SIS300, uloženy v podzemních tunelech v hloubce až 17 metrů s obvodem 1100 metrů. Částice budou do SIS100 vháněny z urychlovače SIS18 a dále urychleny až na 99% rychlosti světla. Takto urychlené částice jsou pak buďto použity přímo k experimentům, k přípravě tzv. sekundárních částic, nebo vehnány do tzv. úložných prstenců k pozdějšímu použití. [11]

Úložné prstence

K prstencovým urychlovačům je připojen komplexní systém úložných prstenců a dalších experimentálních stanic. Úložné prstence vytvářejí pomocí silného magnetického pole cyklickou trajektorii příchozích svazků, na rozdíl od prstencových urychlovačů ale svazkům již nedodávají žádnou energii, tedy je již dále neurychlují. Tento přístup je výhodný, protože příprava vysokoenergetických svazků je náročná a bez použití úložných prstenců jsou připravené svazky po průletu experimentálním stanovištěm ztraceny. Použitím úložných prstenců zajistíme, že vědecké týmy mohou na stejných částicích pracovat při každém průletu experimentálním stanovištěm a dále manipulovat s vlastnostmi svazku. Navíc použití úložných prstenců de facto dále zvyšuje intenzitu svazku bez nutnosti dalšího použití urychlovače.

Pro skladování a manipulaci se svazky se počítá s několika prstenci, včetně vysokoenergetického prstence HESR[7], určeného k práci s antiprotonovými svazky, sběrného prstence CR[8] a experimentálního prstence NESR[9]. Komplex úložných prstenců umožňuje chlazení svazků a zahrnuje terčová stanoviště pro rozptylové experimenty. [3]

Obrázek 2.2: Fotografie datového centra Green IT Cube, Administrativní budova GSI je viditelná vpravo. Obrázek byl převzat z webových stránek GSI [17].

Green IT Cube

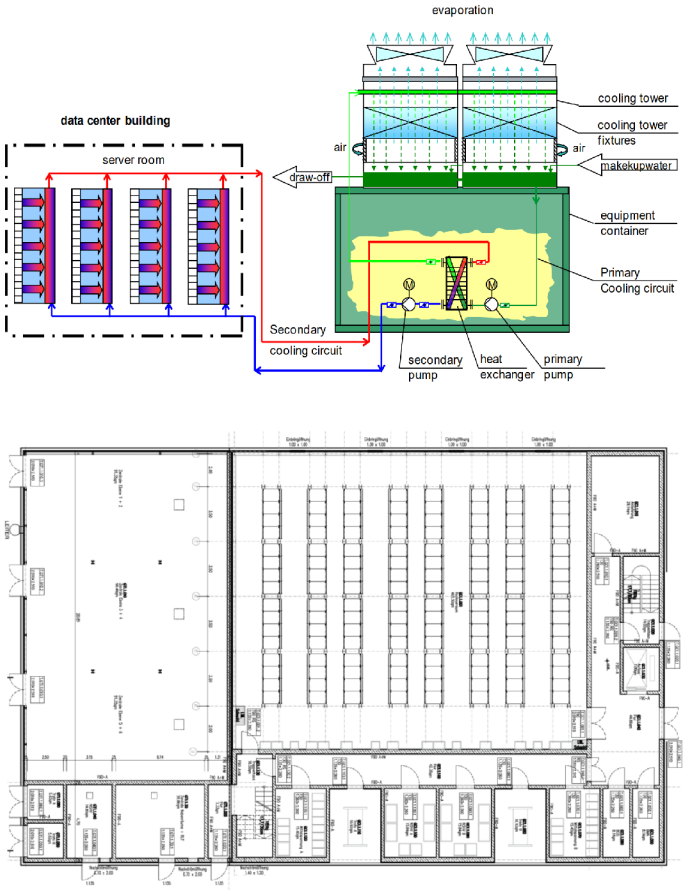
V roce 2008 se odhadovalo, že informační a komunikační technologie byly zodpovědné za 2% světové produkce skleníkových plynů, což je množství srovnatelné s emisemi letadlové dopravy. [23]. Firma Intel v roce 2012 odhadovala rozložení emisí své ICT[10] infrastruktury následovně: [8]

- 6% připadá na uživatelskou výpočetní techniku (kancelářské počítače, monitory, tiskárny apod.),
- 24% připadá na ICT infrastrukturu mimo datová centra,
- 70% připadá na datová centra.

Podle výpočtů institutu Borderstep činila již v roce 2008 spotřeba elektrické energie datových center v Německu 10,1 TWh, což odpovídá 1,8% celkové německé spotřeby toho roku a celkovým nákladům přibližně 1,1 miliardy eur. Přitom výpočetní výkon (a s ním i nároky na výkon chlazení) datových center se neustále zvyšuje: Studie stejného institutu a autora o deset let později uvádí, že v roce 2017 činila spotřeba energie datových center v Německu 13,2 Terawatthodin.

Podle EYP Mission Critical Facilities Inc., New York [23], typicky ne víc než 50% energie potřebné k provozu datových center je využito na vlastní napájení IT vybavení. Uveďme

[7]HESR – High Energy Storage Ring (vysokoenergetický úložný prstenec)
[8]CR – Collector Ring (sběrný prstenec)
[9]NESR – New Experimental Storage Ring (nový experimentální úložný prstenec)
[10]ICT – Information and Communication Technologies (informační a komunikační technologie)



Obrázek 2.3: Diagram chladícího systému datového centra Green IT Cube (nahoře), půdorys přízemního patra budovy (dole). Obrázky byly přejaty z [23].

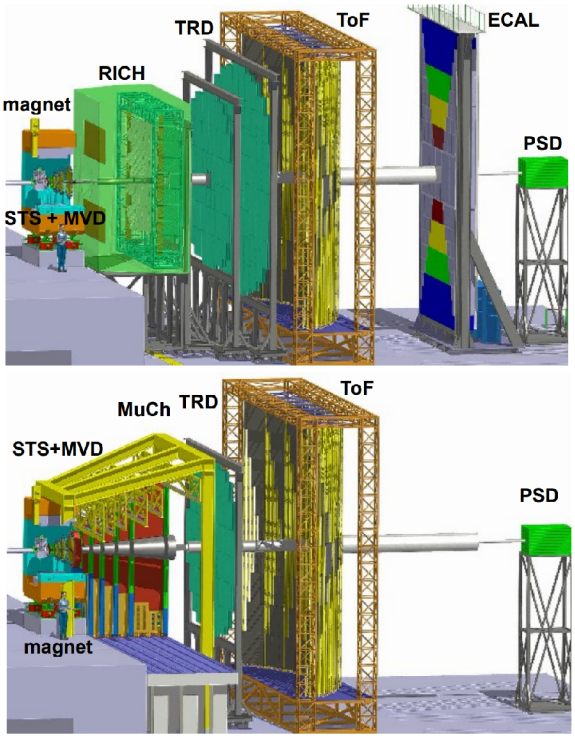
si metriku pro určení energetické efektivity Datového centra: **PUE**[11] je metrika, která zkoumá, jak velkou část celkové energetické spotřeby datového centra vyžaduje napájení výpočetní techniky:

$$\textbf{PUE} = \frac{\text{Celková spotřeba objektu}}{\text{Spotřeba výpočetní techniky}} \tag{2.1}$$

Přičemž PUE typického datového centra se může pohybovat kdekoli mezi $1,6$ a $3,5$, a teoretické PUE $= 1$ by mělo takové datové centrum, které využije veškerou svoji spotřebu energie pouze k napájení výpočetní techniky.

Součástí pracoviště GSI je nové datové centrum **Green IT Cube**, které si mimo jiné klade za cíl prozkoumat způsoby, jak snížit zátěž datových center na životní prostředí. Green IT Cube je postaveno tak, aby mohlo být osazeno 768 standardními devatenáctipalcovými rozvaděči v šesti patrech budovy. Zařízení disponuje kapacitou až 15 MW plně redundantního chladícího výkonu s PUE menším než 1,07. Takové efektivity bylo dosaženo pomocí speciálního dvouokruhového vodního chlazení, kdy sekundární okruh uvnitř datového centra chladí jednotlivé rozvaděče (nikoliv jednotlivá zařízení v nich, díky čemuž je možné rozvaděče osadit výrazně širším výběrem výpočetní techniky), výměník tepla v

[11] PUE – Power Usage Effectiveness (efektivita spotřeby energie)



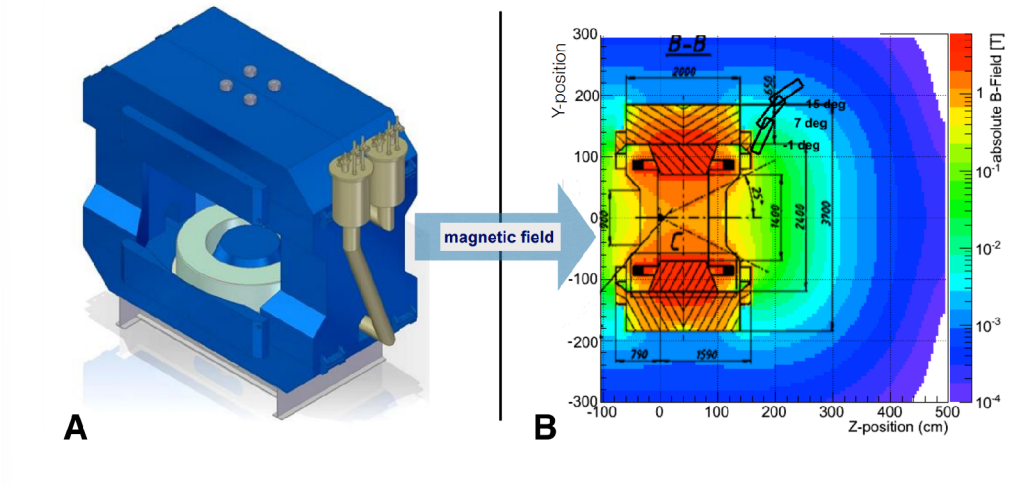
Obrázek 2.4: Rozvržení experimentu CBM v jeho dvou operačních nastaveních: nastavení pro detekci elektronů (obrázek nahoře) a pro detekci mionů (obrázek dole). Pořadí součástí detektoru v elektronové konfiguraci je následující: Micro Vertex Detector (MVD) a Silicon Tracking System (STS), oba usazené uvnitř silného supravodivého magnetu, Ring Imaging Cherenkov Detector (RICH), Transicion Radiation Detector (TRD, dva za sebou) Time of Flight (ToF) detektor, elektromagnetický kalorimetr (ECAL) a Projectile Spectator Detector (PSD) funguje jako hadronový kalorimetr. V mionové konfiguraci je odstraněn elektromagnetický kalorimetr a RICH detektor je nahrazen systémem mionových komnat (Muon Chambers system – MUCH) Převzato z [1].

přízemí budovy pak odpadní teplo převádí do primárního okruhu, které je pak částečně využito k vytápění kanceláří a jídelny budovy GSI (viz obrázek 2.3). Po uvedení do plného výkonu bude mít pracoviště FAIR k dispozici výpočetní výkon tří set tisíc procesorových jader, až 135 Petabajtů (35 PB pro stávající infrastrukturu GSI a 100 PB v rámci rozšíření pro FAIR) online úložiště na pevných discích a robota s 4280ti sloty pro magnetické pásky pro archivaci až 62 PB nekomprimovaných experimentálních dat. Pro experimenty FAIR se dále počítá s vysokou přenosovou rychlostí přes 1 Terabajt za sekundu pro přenos velkého objemu nezpracovaných experimentálních dat z pracovišť jednotlivých experimentů[18].

Při uvedení do provozu v roce 2014 obdrželo centrum Green IT Cube mnohá ocenění, včetně prvního místa v žebříčku superpočítačů Green500, které řadí nejefektivnější superpočítače podle GFLOPS/W.

2.2 Experiment CBM

CBM je jeden z vědeckých pilířů pracoviště FAIR. Jeho cílem bude studovat strukturu a stavovou rovnici baryonové hmoty o hustotách, které bychom mohli najít v jádrech ne-



Obrázek 2.5: Nákres supravodivého magnetu **(A)** a znázornění intenzity magnetického pole v magnetu **(B)**. V bodě (0,0) je znázorněn terč detektoru, graf intenzity je vykreslen podél roviny YZ, $x = 0$. Převzato z [1].

utronových hvězd. Experiment bude zkoumat srážky svazků protonů a těžkých iontů se statickým terčem při svazkových energiích od 2 do 45 AGeV (GeV na nukleon). [1] Experiment bude zaměřen mimo jiné na [14]:

- studium stavové rovnice jaderné hmoty o hustotách, které bychom mohli najít v neutronových hvězdách
- hledání fázové hranice mezi hadronovou fází a kvark-gluonovým plazmatem, případě oblasti fázové koexistence a koncového kritického bodu kvantové chromodynamiky.
- pátrání po jednoduchých a dvojitých hyperjádrech: těžkých objektech s nenulovým kvantovým číslem podivnosti

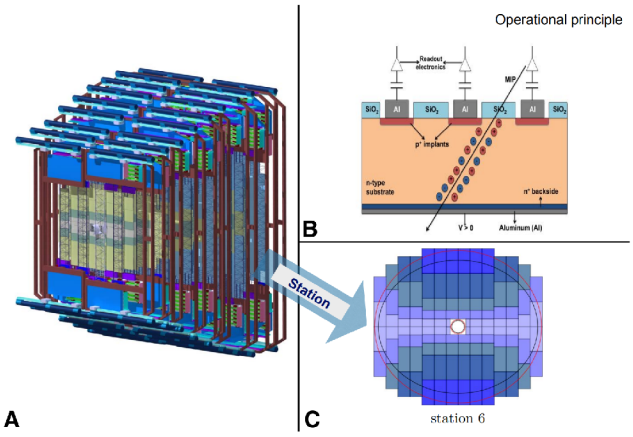
Experiment CBM je velmi rozsáhlý a složitý detektor, stávající se z mnoha podsystémů, my se však v rámci této práce budeme zabývat pouze jeho hlavním detektorem **STS**.

Supravodivý magnet

Účel silného supravodivého magnetu je *ohýbat* nabité částice, u kterých pak podle jejich zakřivení dokážeme určit jejich náboj a energii. K tomu, aby bylo možné uvnitř magnetu uložit detektor STS bylo potřeba, aby měl magnet rozměry minimálně $1,3 \times 1,3m^2$

Silicon Tracking System

Silicon Tracking System (STS) je primární detektor experimentu CBM. Cílem detektoru STS je vhodný odhad vlastností jednotlivých trajektorií částic, kterých může detektorem proletět během jedné „události“ při vzorkovací frekvenci 10 MHz až 700. Detektor je uspořádán do osmi „hladin“: stanovišť, kdy první je od terče detektoru vzdálen 30 centimetrů, mezi každou hladinou detektoru je pak vždy deset centimetrů prostor. Detektorové hladiny jsou uspořádány tak, aby pokrývaly polární úhly od 2,5°do 25°. Pro polární úhly menší než



Obrázek 2.6: 3D pohled na detektor STS a rozvržení jeho stanovišť **(A)**, diagram křemíkového proužkového detektoru **(B)** a schéma rozvržení šesté hladiny detektoru STS (světlejší barva proužku značí vytíženější oblasti detektoru) **(C)**. Převzato z [1]

2,5°prochází detektorem silný paprsek hadronů, u kterého je vzhledem k vysokým energiím a množství částic detekce zbytečná.

STS má citlivé elektrody na každé straně detektoru, kde každá strana se stará o odečítání jedné ze dvou souřadnic, přitom proužky nesvírají pravý úhel, nýbrž úhel 15°, čímž se sice zmenší rozlišení detektoru podél jedné osy (y), zvýší se ale rozlišení druhé osy (x). Detektor je takto zařízen z toho důvodu, že nabité částice jsou ovlivněny magnetickým polem silného supravodivého magnetu, ve kterém se detektor nachází, v důsledku čehož mají částice uvnitř magnetu mnohem větší rozptyl ve směru osy x, než ve směru osy y.

Kapitola 3

Rekonstrukce trajektorií částic

Následující podkapitola byla převzata z disertační práce Dr. Akishiny [1]. Díky neustálému vývoji v oborech **urychlovačové fyziky**[1] a výpočetní techniky je dnes možné provozovat HEP experimenty při energiích a četnostech kolizí, které dříve nebyly možné. Tento fakt má ale také za důsledek, že zpracování stále se zvětšujících objemů experimentálních dat je čím dál komplikovanější úkol, na který již běžně dostupná výpočetní technika není vhodná.

Výrazně vyšší kolizní energie částic v HEP experimentech způsobují komplikovanější vzorce událostí a výrazně větší množství sledovaných částic v detektoru. Zároveň se zvyšuje také frekvence těchto událostí, což má za následek omezení času, který máme k dispozici na vyhodnocení každé takové události, tedy k rekonstrukci trajektorií jednotlivých částic a srážek mezi nimi.

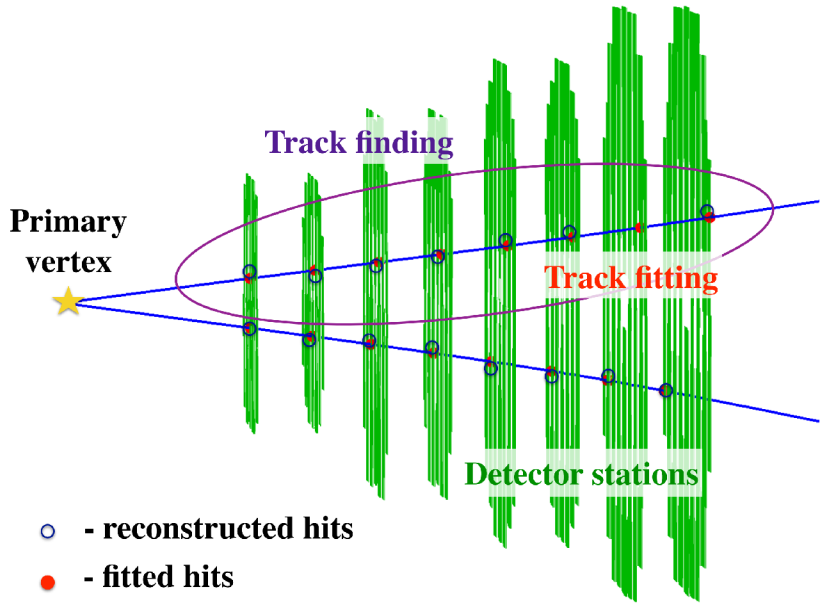
Vzorce srážek rekonstruujeme tak, že nejprve zrekonstruujeme trajektorie jednotlivých částic, které při průletu detektorem zanechávají měřitelnou stopu na jednotlivých detektorových hladinách, tzv. **zásahy**. Proces rekonstrukce každé události probíhá v experimentu CBM ve Čtyřech fázích:

- hledání trajektorií,
- prokládání trajektorií,
- identifikace částic podle proložených trajektorií,
- hledání částic s krátkou dobou života.

Přestože hledání a prokládání trajektorií částic spolu do značné míry souvisí, tradičně na ně pohlížíme jako na dvě odlišné fáze rekonstrukce. Zatímco hledání trajektorií je problém rozpoznávání vzorů, kde jednotlivé zásahy rozdělujeme do skupin, o kterých předpokládáme, že by mohly být vyvolány stejnou částicí. Po tom, co jednotlivé zásahy takto seskupíme, lze pro každou skupinu odhadnout nejpravděpodobnější parametry dráhy částice, a to i za přítomnosti velkého množství šumu. Tomuto postupu říkáme prokládání trajektorií.

Proces hledání trajektorií pracuje zpravidla nad nezpracovanými, silně zašuměnými daty, která nemohla být předem nijak předzpracována ani zredukována. Z tohoto důvodu je proces hledání trajektorií považován za nejsložitější a časově nejnáročnější fázi rekonstrukce trajektorií.

[1] z angl. accelerator physics – odnož aplikované fyziky, zabývající se návrhem, stavbou a provozem urychlovačů částic



Obrázek 3.1: První dvě fáze rekonstrukce trajektorií. Převzato z [1].

Jednotlivé fáze rekonstrukce trajektorií se ale postupně stávají čím dál provázanějšími a čím silnější jsou souvislosti mezi nimi, tím jsou hranice mezi nimi mlhavější. V poslední době je běžné během první rekonstrukční fáze odhadovat jednotlivé parametry dráhy částice v každém kroku, díky čemuž získáváme s každým dalším měřením lepší povědomí o parametrech proložené dráhy, což nám dále umožňuje již v této první fázi vylučovat zásahy, u kterých je nepravděpodobné, že budou patřit k právě rekonstruované dráze. Tento trend je primárně určován existencí rekurzivních algoritmů užívajících metody Kalmanovy Filtrace. Tyto algoritmy jsou hlavním tématem této práce a metodou Kalmanova Filtru se budeme dopodrobna zabývat v příští podkapitole.

3.1 Rekurzivní Bayesovský odhad

Rekurzivní Bayesovský odhad, též Bayesovský Filtr, je obecný algoritmus pro výpočet odhadu stavu systému na základě jednotlivých měření systému:

$$bel\left(x_t\right) = p\left(x_t | z_{1:t}, u_{1:t}\right) \tag{3.1}$$

kde x_t značí stav systému v čase t, $bel(x_t)$ značí předpokládaný stav v čase t, z_t značí měření v čase t a u_t značí **ovládání systému**[2] v čase t. Stěžejní zákon pro odvození Bayesovského Filtru je *Bayesův teorém*, který zní následovně:

$$p\left(A|B\right) = \frac{p\left(B|A\right)p\left(A\right)}{p\left(B\right)} \tag{3.2}$$

kde A a B jsou náhodné jevy, $p\left(A\right)$ je pravděpodobnost jevu A a $p\left(A|B\right)$ je pravděpodobnost jevu A za předpokladu, že je jev B pravdivý. Pro účely odvození Bayesovských filtrů je užitečnější aplikace Bayesova teorému pro 3 náhodné jevy:

[2] z angl. system control commands

$$p\left(A|B,C\right) = \frac{p\left(B|A,C\right)p\left(A|C\right)}{p\left(B|C\right)} \tag{3.3}$$

Když dosadíme definici Bayesovského filtru (rovnice 3.1) do rovnice výše, dostaneme:

$$p\left(x_t|z_t, z_{1:t-1}, u_{1:t}\right) = \frac{p\left(z_t|x_t, z_{1:t-1}, u_{1:t}\right)p(x_t|z_{1:t-1}, u_{1:t})}{p\left(z_t|z_{1:t-1}, u_{1:t}\right)} \tag{3.4}$$

Pro lepší čitelnost zavedeme za jmenovatel rovnice 3.4 normalizační konstantu η:

$$\eta = \frac{1}{p\left(z_t|z_{1:t-1}, u_{1:t}\right)} \tag{3.5}$$

$$bel(x_t) = \eta\, p\left(z_t|x_t, z_{1:t-1}, u_{1:t}\right)\ p\left(x_t|z_{1:t-1}, u_{1:t}\right) \tag{3.6}$$

Pro odvození rekurzivního vzorce Bayesovského filtru použijeme fakt, že skutečný stav systému x je **Markovův proces se skrytými stavy**. Na rozdíl od klasických *Markovových řetězců* neznáme vnitřní stav systému, o tomto stavu získáváme informace pouze za pomoci měření z_k. Protože je vnitřní stav systému dán Markovovým procesem, víme, že bude systém vykazovat *Markovovu vlastnost*, která říká, že vývoj Markovova procesu je závislý pouze na jeho předchozím stavu. Díky této vlastnosti víme, že pokud známe stav x_t, k určení pravděpodobnosti měření z_t nepotřebujeme žádné znalosti, protože pravděpodobnost $p(z_t|x_t)$ nijak neovlivní:

$$p\left(z_t|x_t, z_{1:t-1}, u_{1:t}\right) = p\left(z_t|x_t\right) \tag{3.7}$$

Rovnici 3.6 tedy můžeme zjednodušit následujícím způsobem:

$$bel\left(x_t\right) = \eta\, p\left(z_t|x_t\right)\ p\left(x_t|z_{1:t-1}, u_{1:t}\right) \tag{3.8}$$

V dalším kroku aplikujeme **pravidlo úplné pravděpodobnosti**, které říká, že máme-li úplný systém jevů $\{B_n : n = 1, 2, 3, \dots\}$, pak pravděpodobnost libovolného jevu A lze určit jako:

$$p(A|C) = \sum_{i=1}^{n} p(A|B_i, C)\ p(B_i|C) \tag{3.9}$$

V našem případě vzorec úplné pravděpodobnosti (v jeho spojité podobě) umožní přidat k druhému členu rovnice 3.8 další podmínku, kterou využijeme následně při odvození rekurzivního kroku filtru:

$$p\left(a|c\right) = \int_{-\infty}^{\infty} p(a|b, c)\ p(b|c)\ db \tag{3.10}$$

Dosadíme-li do původní rovnice, dostaneme:

$$bel\left(x_t\right) = \eta\, p\left(z_t|x_t\right) \int p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t})\ p(x_{t-1}|z_{1:t-1}, u_{1:t})\ dx_{t-1} \tag{3.11}$$

Nyní můžeme zjednodušit člen $p\left(x_t|x_{t-1}, z_{1:t}, u_{1:t}\right)$ pomocí Markovovy vlastnosti, díky které víme, že ke stanovení pravděpodobnosti, že se nacházíme ve stavu x_t, nám stačí znát předchozí stav systému x_{t-1} a informaci o tom, jak se systém vyvíjí ze stavu x_{t-1} do stavu

x_t (tuto znalost reprezentujeme jako u_t). Všechny ostatní znalosti jsou pro tento výpočet irelevantní, takže tento člen můžeme pomocí Markovovy vlastnosti přepsat jako:

$$p\left(x_t|x_{t-1}, z_{1:t}, u_{1:t}\right) = p\left(x_t|x_{t-1}, u_t\right) \tag{3.12}$$

Výsledná rovnice má pak následující tvar:

$$bel\left(x_t\right) = \eta\, p\left(z_t|x_t\right) \int p(x_t|x_{t-1}, u_t)\; p(x_{t-1}|z_{1:t-1}, u_{1:t})\; dx_{t-1} \tag{3.13}$$

V následujícím kroku zkoumáme člen $p(x_{t-1}|z_{1:t-1}, u_{1:t})$: zkoumáme pravděpodobnost, že minulý stav systému byl x_{t-1} za předpokladu, že známe veškerá předešlá měření až do času $t-1$, stejně tak víme, jak se systém vyvíjel v každém okamžiku až do času $t-1$. Zároveň je ale přítomna podmínka u_t, která říká, že víme, jakým způsobem se systém vyvine ze stavu x_{-1} do stavu x_t. Můžeme předpokládat, že znalost, jakým způsobem se bude systém v budoucnosti vyvíjet, nemá na pravděpodobnost, že v čase $t-1$ bude systém ve stavu x_{t-1} žádný vliv, a tuto podmínku ze členu $p(x_{t-1}|z_{1:t-1}, u_{1:t})$ vyloučit:

$$p\left(x_{t-1}|z_{1:t-1}, u_{1:t}\right) = p\left(x_{t-1}|z_{1:t-1}, u_{1:t-1}\right) \tag{3.14}$$

Dosazením získáváme tvar rovnice:

$$bel\left(x_t\right) = \eta\, p\left(z_t|x_t\right) \int p(x_t|x_{t-1}, u_t)\; p(x_{t-1}|z_{1:t-1}, u_{1:t-1})\; dx_{t-1} \tag{3.15}$$

Konečně, z původní definice odhadu $bel\left(x_t\right)$ v rovnici 3.1 vyplývá, že člen

$$p\left(x_{t-1}|z_{1:t-1}, u_{1:t-1}\right)$$

můžeme přepsat jako:

$$p\left(x_{t-1}|z_{1:t-1}, u_{1:t-1}\right) = bel\left(x_{t-1}\right) \tag{3.16}$$

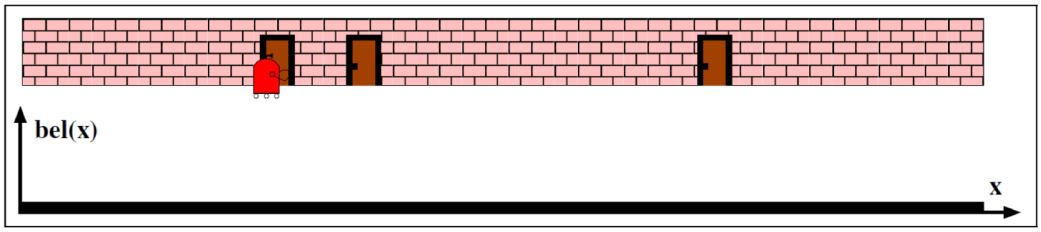
Čímž získáme potřebný rekurzivní krok a můžeme dosazením do rovnice 3.13 dokončit odvození Bayesovského filtru:

$$bel\left(x_t\right) = \eta\, p\left(z_t|x_t\right) \int p(x_t|x_{t-1}, u_t)\; bel\left(x_{t-1}\right)\; dx_{t-1} \tag{3.17}$$

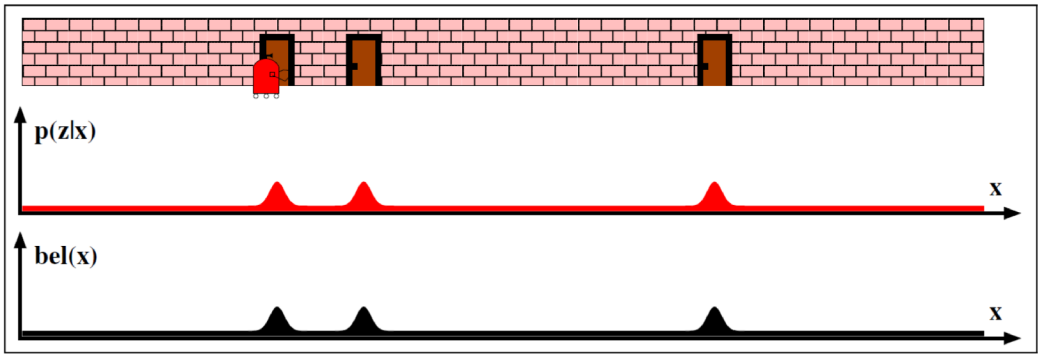
Výsledný filtr můžeme dále rozepsat (a typicky rozepisujeme) na následující dva kroky:

- predikční krok
$$\overline{bel}\left(x_t\right) = \int p(x_t|x_{t-1,u_t})\; bel\left(x_{t-1}\right)\; dx_{t-1} \tag{3.18}$$
- korekční krok
$$bel\left(x_t\right) = \eta\, p\left(z_t|x_t\right)\; \overline{bel}\left(x_t\right) \tag{3.19}$$

kde $\overline{bel}\left(x_t\right)$ je odhad x_t na základě předchozích měření, $bel(x_t)$ je upravený odhad x_t na základě nového měření, $p\left(k_t|u_t, x_{t-1}\right)$ v predikčním kroku nazýváme *pohybový model* a $p\left(z_t|x_t\right)$ v korekčním kroku nazýváme *model měření.*



Obrázek 3.2: Robot využívající k lokalizaci v 1D prostoru Bayesův Filtr. Do okamžiku prvního pozorování dveří nemá robot žádnou informaci o své poloze, $bel(x)$ je proto rovnoměrně rozloženo. Převzato z [25]



Obrázek 3.3: Robot využívající k lokalizaci v 1D prostoru Bayesův Filtr. V okamžiku prvního pozorování dveří robot aktualizuje odhad své polohy podle tak, že $bel(x)$ nabude normálního rozložení u každých dveří. Převzato z [25]

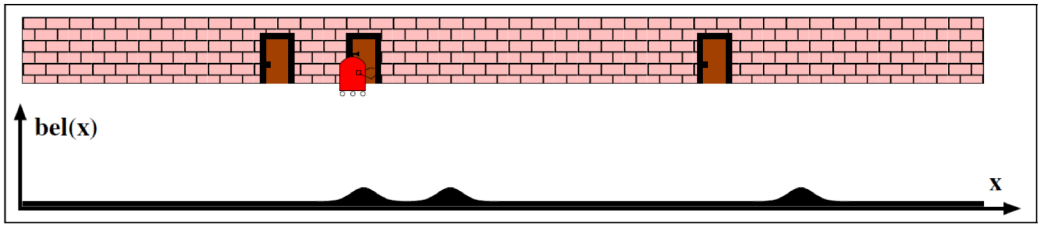
Příklad použití Bayesovského Filtru

Mějme robota, pohybujícího se v jednorozměrném vesmíru podél zdi s dveřmi. Robot má pouze jeden senzor, který rozezná, jestli má před sebou dveře, nebo zeď. Robot zná pozici všech dveří, dveře jsou ale identické a robot rozezná pouze, jestli stojí přede dveřmi, neví ale, před kterými.

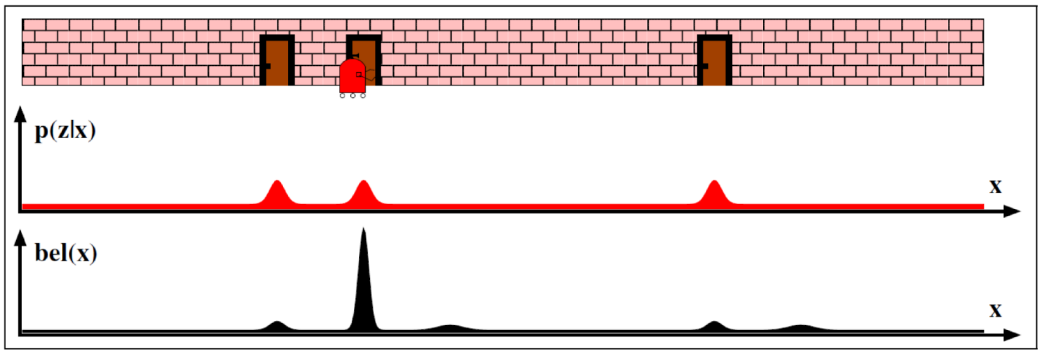
Do okamžiku, kdy poprvé zaregistruje dveře (obr. 3.2), nemá robot žádnou informaci o své poloze: hodnota $bel(x)$ je pro všechna x stejná. V okamžiku, kdy robotův senzor dveří dveře zaregistruje (obr. 3.3), upraví svůj odhad $bel(x)$ na základě posledního pozorování $p(z|x)$. Robot ale neví, před kterými z dveří se právě nachází, všem z nich tedy přiřadí stejnou pravděpodobnost.

Vzhledem k tomu, že robot nedokáže zaručit, že se mezi jednotlivými měřeními pohnul o přesnou vzdálenost (s každým krokem mohl urazit vzdálenost mírně odchýlenou od vzdálenosti, kterou urazit zamýšlel), modeluje tuto nejistotu jako normální rozložení se středy μ v místech, kde předpokládá, že by se nacházel, kdyby se pohyboval přesně. Metoda taky s každým dalším pozorováním, kdy nebyly pozorovány dveře, způsobuje, že se u normálních rozložení v místech, kde robot předpokládá, že se nachází, zvětšuje jejich rozptyl (v důsledku agregace chyb jednotlivých kroků, obr. 3.4).

V okamžiku, kdy robot zaregistruje druhé dveře, jeho pozorování $p(z|x)$ bude mít stejná rozložení, jako když stál před prvními dveřmi. Tentokrát ale ve chvíli, kdy pomocí pozorování upraví odhad své polohy $bel(x)$, získá mnohem jistější odhad své polohy, protože



Obrázek 3.4: Normální rozložení pravděpodobnosti v místech, kde robot pozoroval dveře, se díky tomu, že robot nepozoruje dveře, postupně vyhlazují a pohybují ve směru pohybu robota. Převzato z [25]



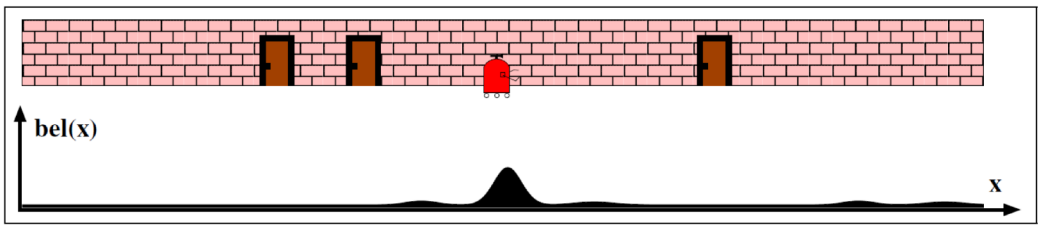
Obrázek 3.5: Druhé pozorování dveří díky metodě Bayesova Filtru výrazně zlepší robotovo povědomí o jeho poloze. Převzato z [25]

hodnota pravděpodobnosti, že se robot nachází u druhých dveří výrazně posílí existující odhad v tomto místě (obr. 3.5).

Robot má v tuto chvíli poměrně dobrou představu o tom, kde se nachází, a i když se s dalšími pozorováními bude jeho odhad dále „vyhlazovat," jeho skutečná poloha bude mít i nadále nejvyšší pravděpodobnost, přestože rozptyl jeho rozdělení se bude zvětšovat až do okamžiku, kdy robot zaregistruje poslední dveře (obr. 3.6).

Implementace Bayesovského Filtru

Rekurzivní Bayesův odhad je obecný pravděpodobnostní přístup k rekurzivnímu odhadu neznámého rozložení pravděpodobnosti; v závislosti na typu řešeného problému vybíráme jednu z implementací Bayesovského filtru. Existuje více možných implementací Bayesovských filtrů, zejména pak:



Obrázek 3.6: Po druhém pozorování dveří se robotův odhad polohy $bel(x)$ dále vyhlazuje, nadále ale správně odhaduje nejpravděpodobnější polohu. Převzato z [25]

- **Kalmanův filtr (KF)**, optimální Bayesovský filtr pro filtrování nad normálními rozděleními a lineárními modely,
- **Rozšířený Kalmanův filtr (EKF)**, rozšíření Kalmanova filtru pro nelineární modely. Rozšířený Kalmanův filtr je v praxi nejrozšířenější metoda pro aproximaci pohybu částic.
- **Částicové filtry**[3], jinak také metody SMC[4], pracují nad nelineárními modely s arbitrárními rozloženími pravděpodobností. Díky tomu jsou ze zde zmíněných metod použitelné pro nejširší spektrum problémů, pro naše účely jsou ale z důvodu nízké rychlosti nevhodné.

Právě Kalmanovými filtry a EKF se budeme v další podkapitole podrobně zabývat.

3.2 Kalmanův Filtr

Kalmanův Filtr, známý také jako metoda **LQE**[5], je implementace Bayesova Filtru, která poskytuje optimální odhad stavu dynamického systému za předpokladu, že:

- pohybový model i model měření jsou lineární
- všechny náhodné proměnné v systému mají *normální rozdělení*

Metoda slouží k získání optimálního odhadu $\boldsymbol{r}$ neznámého stavového vektoru $\boldsymbol{r}^t$ za pomocí *vektorů měření* (jinak také *pozorování*) $\boldsymbol{m}_k, k = 1 \dots n$ vektoru $\boldsymbol{r}^t$. Metoda je inicializována volbou počátečního odhadu $\boldsymbol{r} = \boldsymbol{r}_0$[6] a postupně s každým dalším měřením zpřesňuje odhad $\boldsymbol{r}$ stavového vektoru $\boldsymbol{r}^t$. [16].

Hodnota vektoru $\boldsymbol{r}^t$ se může v čase měnit: jeho změnu u lineárních modelů reprezentujeme maticí F_k, která obsahuje informaci o tom, jak se v čase mění každá složka stavového vektoru $\boldsymbol{r}^t$. Stav systému se tak mezi jednotlivými měřeními mění, nemusíme ale přesně vědět, jak a/nebo kdy k těmto změnám dochází. Veškeré náhodné jevy mezi měřeními $\boldsymbol{m}_{k-1}$ a $\boldsymbol{m}_k$, které nedokážeme modelovat v rámci propagační matice F_k, modelujeme pomocí takzvaného *procesního šumu*, který je reprezentován vektorem $\boldsymbol{\nu}_k$. Výsledná rovnice změny stavového vektoru $\boldsymbol{r}^t$ vypadá následovně:

$$\boldsymbol{r}_k^t = F_k \boldsymbol{r}_{k-1}^t + \boldsymbol{\nu}_k \tag{3.20}$$

kde F_k je lineární propagační operátor a $\boldsymbol{\nu}_k$ je vektor procesního šumu mezi měřeními $\boldsymbol{m}_{k-1}$ a $\boldsymbol{m}_k$. V případě Kalmanova Filtru předpokládáme, že měření $\boldsymbol{m}_k$ je lineárně závislé na stavu systému $\boldsymbol{r}_k^t$, tedy říkáme, že **model měření** H_k je lineární:

$$\boldsymbol{m}_k = H_k \boldsymbol{r}_k^t + \boldsymbol{\eta}_k \tag{3.21}$$

kde $\boldsymbol{\eta}_k$ je vektor chyby měření $\boldsymbol{m}_k$, H_k je *model měření*. Model měření je matice s dimenzí $m \times n$, kde m je rovno velikosti stavového vektoru $\boldsymbol{r}^t$ a n je rovno velikosti vektoru měření $\boldsymbol{m}_k$. Model měření určuje, jaký je vztah mezi jednotlivými měřeními systému a

[3] z angl. Particle Filters
[4] SMC – Sequential Monte Carlo (sekvenční Monte Carlo metody)
[5] LQE – Linear Quadratic Estimation (lineárně-kvadratický odhad)
[6] U EKF musíme být narozdíl od klasického KF při volbě $\boldsymbol{r}_0$ opatrní, metoda může při špatně zvolené počáteční hodnotě divergovat

jeho reálným stavem. Kalmanův Filtr předpokládá, že chyba měření a procesní šum nijak nekorelují a že známe jejich **kovarianční matice**:

$$\begin{aligned} \langle \boldsymbol{\eta}_i \rangle = \langle \boldsymbol{\nu}_j \rangle &= \mathbf{0}, \\ \left\langle \boldsymbol{\eta}_k \cdot \boldsymbol{\eta}_k^T \right\rangle &= V_k, \\ \left\langle \boldsymbol{\nu}_k \cdot \boldsymbol{\nu}_k^T \right\rangle &= Q_k. \end{aligned} \tag{3.22}$$

kde notace $\langle \boldsymbol{X} \rangle = E(\boldsymbol{X})$ značí střední hodnotu náhodné veličiny $\boldsymbol{X}$. **Kovarianční matice** (také variačně-kovarianční matice) reálné vícerozměrné náhodné veličiny X o n složkách je čtvercová matice $(n \times n)$, která obsahuje **kovarianci** i-té a j-té složky náhodné veličiny $\boldsymbol{X}$:

$$X = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2 & \cdots & \sigma_1\sigma_n \\ \sigma_2\sigma_1 & \sigma_2^2 & \cdots & \sigma_2\sigma_n \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_n\sigma_1 & \sigma_n\sigma_2 & \cdots & \sigma_n^2 \end{bmatrix}$$

kde $\sigma_i\sigma_j = cov(X_i, X_j) = E\left[(X_i - E[X_i])\,(X_j - E[X_j])\right]$ je kovariance veličin X_i a X_j a $E[X_i] = \langle X_i \rangle$ je střední hodnota veličiny X_i.

Klasický Kalmanův Filtr běžně členíme do čtyř základních kroků:

1. **inicializace**, kdy zvolíme počáteční hodnoty odhadu stavového vektoru $\boldsymbol{r}_0^+$ a kovarianční matice C_0^+,
2. **predikce**, kdy pomocí propagačního operátoru F_k stanovíme odhad dalšího stavu stavového vektoru a kovarianční matice,
3. **zpracování procesního šumu**, kdy upravíme odhad příští kovarianční matice C_k^- pomocí kovarianční matice procesního šumu Q_k,
4. **filtrace**, kdy pomocí nového měření upravíme odhad stavového vektoru a kovarianční matice tak, aby příští odhady byly přesnější.

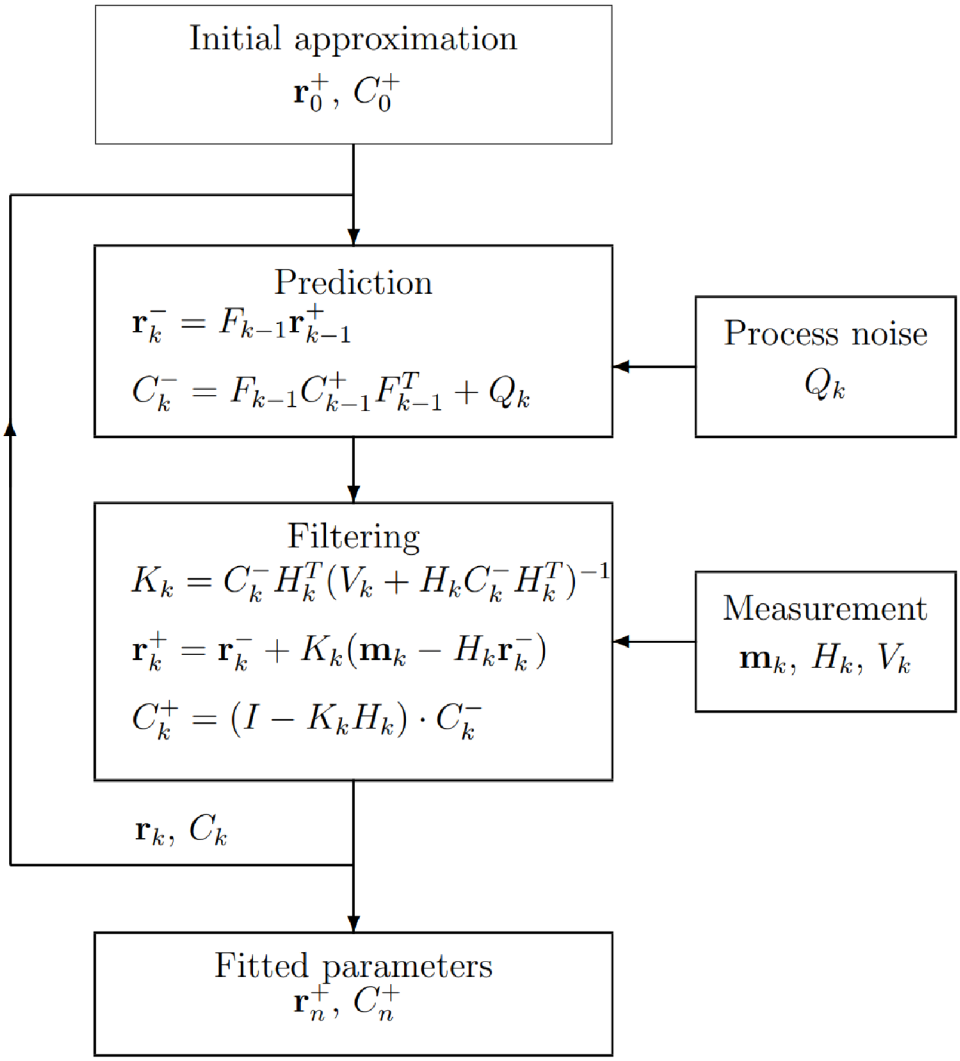
Inicializace Během inicializace Kalmanova Filtru musíme určit první aproximaci stavového vektoru $\boldsymbol{r}_0^+$ a kovarianční matice C_0^+: Mějme reálný stav systému, vyjádřený vektorem $\boldsymbol{r}^t$ a jeho odhad $\boldsymbol{r}$. Odhad stavu systému $\boldsymbol{r}$ má vždy konečnou přesnost: zavedeme vektor chyby $\boldsymbol{\xi}$, který vyjadřuje rozdíl mezi reálným stavem systému a jeho odhadem a kovarianční matici C, které definujeme následovně:

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{r}^t + \boldsymbol{\xi} \\ C &= \left\langle \boldsymbol{\xi} \cdot \boldsymbol{\xi}^T \right\rangle \end{aligned} \tag{3.23}$$

Podle dostupných informací zvolíme počáteční odhad vektoru $\boldsymbol{r}_0^+$, a jeho kovarianční matici potřebujeme zvolit tak, aby reprezentovala fakt, že před prvním měřením nemáme o stavu systému žádné znalosti. Kovarianční matici C_0 proto zvolíme následovně:

$$C_0^+ = \boldsymbol{I} \cdot \inf^2 \tag{3.24}$$

kde $\boldsymbol{I}$ je jednotková matice a *inf* značí velké kladné číslo.



Obrázek 3.7: Blokové schéma klasického Kalmanova Filtru. Převzato z [16]

Predikce a zpracování procesního šumu Pokud očekáváme, že se stav systému mezi jednotlivými měřeními změní, musí se odpovídajícím způsobem změnit i jeho odhad a kovarianční matice: Pomocí propagačního operátoru F_{k-1} získáme predikci stavu systému $\boldsymbol{r}_k^-$ a jeho kovarianční matice C_k^-:

$$\begin{aligned} \boldsymbol{r}_k^- &= F_{k-1}\boldsymbol{r}_{k-1}^+ \\ C_k^- &= F_{k-1}C_{k-1}^+F_{k-1}^T + Q_k \end{aligned} \tag{3.25}$$

kde operátor F_{k-1} modeluje vývoj systému mezi stavy $\boldsymbol{r}_{k-1}$ a $\boldsymbol{r}_k$, F_{k-1}^T je transponovaná matice F_{k-1}, Q_k je kovarianční matice procesního šumu, kterou metoda Kalmanova Filtru předpokládá, že předem známe. Zatímco operátor F_{k-1} modeluje deterministické změny uvnitř systému, přidáním kovarianční matice procesního šumu Q_k zohledňujeme stochastické procesy, které popsat v pohybovém modelu je buď nemožné, nebo zbytečně složité.

Zatímco $\boldsymbol{r}_k^-$ a C_k^-, které vypočítáváme v rámci predikčního kroku na základě minulých měření $\boldsymbol{m}_i$, $i = 1 \dots k-1$, jsou predikce hodnot $\boldsymbol{r}_k$ a C_k, $\boldsymbol{r}_{k-1}^+$ a C_{k-1}^+ jsou již opravené (resp. filtrované) predikce z předchozí iterace Kalmanova Filtru. Vypočítáváme je v rámci filtračního kroku pomocí hodnoty nejnovějšího měření m_k (viz. níže).

V případě, že jsme dosud neprovedli žádné měření, propagujeme původní odhad stavu systému $\boldsymbol{r}_0^+$ a jeho kovarianční matici C_0^+, které jsme určili během inicializačního kroku:

$$\begin{aligned} \boldsymbol{r}_1^- &= F_0 \boldsymbol{r}_0^+ \\ C_1^- &= F_0 C_0^+ F_0^T + Q_1 \end{aligned} \tag{3.26}$$

Filtrace V závěrečném kroku Kalmanova filtru upravíme odhad hodnoty $\boldsymbol{r}_{k-}^-$ (a její kovarianční matice C_{k-1}^-) pomocí nejnovějšího měření $\boldsymbol{m}_k$, díky čemuž získáme nové, optimální odhady těchto veličin $\boldsymbol{r}_k^+$ a C_k^+. Zároveň v tomto kroku vypočítáme χ^2 odchylku odhadu r_k^+ od dosud proběhlých měření $\boldsymbol{m}_i$, $i = 1 \dots k$, díky čemuž dokážeme pro každou iteraci Kalmanova Filtru určit, jak dobře odpovídá odhad měřením, což je pro naše účely velmi užitečné, protože díky χ_k^2 odchylce můžeme rychle vypočítat, které zásahy v detektoru s nejvyšší pravděpodobností patří ke stejné trajektorii.

Pro zjednodušení dalších výpočtů (rovnice 3.31 a 3.32) zavedeme nejprve pomocný člen – takzvané **reziduum** (resp. reziduál, z anglického *residual*, v praxi používáno nejčastěji), korekční člen, který vyjadřuje rozdíl mezi odhadem Kalmanova Filtru $\boldsymbol{r}_k^-$ a vlastním měřením $\boldsymbol{m}_k$.

$$\boldsymbol{\zeta}_k = \boldsymbol{m}_k - H_k \boldsymbol{r}_k^- \tag{3.27}$$

Abychom mohli aktualizovat odhad stavu $\boldsymbol{r}_k^+$, potřebujeme spočítat takzvanou *matici Kalmanova zesílení*[7] K_k:

$$K_k = C_k^- H_k^T \left(V_k + H_k C_k^- H_k^T \right)^{-1} \tag{3.28}$$

kde C_k^- je v predikčním kroku předpokládaná hodnota kovarianční matice, V_k je kovarianční matice chyby měření a H_k je model měření. Rovnici Kalmanova zesílení můžeme ještě drobně zjednodušit tak, že zavedeme takzvanou *váhovou matici* W_k:

$$W_k = \left(V_k + H_k C_k^- H_k^T \right)^{-1} \tag{3.29}$$

Rovnice Kalmanova zrychleni pak bude mít následující tvar:

$$K_k = C_k^- H_k^T W_k \tag{3.30}$$

Jakmile máme Kalmanovo zesílení spočítané, můžeme aktualizovat odhad stavového vektoru $\boldsymbol{r}_k^+$ (a jeho kovarianční matice):

$$\begin{aligned} \boldsymbol{r}_k^+ &= \boldsymbol{r}_k^- + K_k \boldsymbol{\zeta}_k \\ C_k^+ &= (\boldsymbol{I} - K_k H_k) \cdot C_k^- \end{aligned} \tag{3.31}$$

kde $\boldsymbol{I}$ je jednotková matice a $\boldsymbol{m}_k$ je poslední měření systému. V závěrečné fázi spočítáme novou odchylku χ_k^2 na základě posledního měření $\boldsymbol{m}_k$:

[7] z angl. Gain Matrix

$$\chi_k^2 = \chi_{k-1}^2 + \boldsymbol{\zeta}_k^T W_k \boldsymbol{\zeta}_k \tag{3.32}$$

V tuto chvíli získaná hodnota $\boldsymbol{r}_k^+$ s kovarianční maticí C_k^+ je optimální odhad současného stavu systému. Tato dvojice je zároveň výstup dokončené iterace a vstup následující iterace Kalmanova Filtru. Hodnota χ_k^2 nám později pomocí celulárního automatu pomůže ke hledání zásahů, které s nejvyšší pravděpodobností patří ke trajektorii stejné částice.

Kalmanův filtr v experimentu CBM

Následující podsekce byla převzata z disertační práce Dr. Akishiny [1]. Výsledkem hledání trajektorií (fáze 1: Track finding, viz. kapitola 3) by měly být sady měření detektoru uskupené do souborů zásahů, které by v ideálním případě byly všechny vytvořené stejnou částicí. Úkolem fáze 2 (Track fitting) rekonstrukce trajektorií je odhadnout parametry trajektorií a jejich chyby, aby bylo možné získat kinematické vlastnosti částic, které nám později umožní rekonstrukci částic s krátkou *střední dobou života* a nakonec i analýzu fyzikálních procesů, které v detektoru proběhly.

Algoritmus ve fázi 2 využívá tzv. *modelu trajektorií*, což je teoretický předpoklad o pohybových rovnicích pro nabité částice v rámci objemu detektoru. Na pohyb částic v detektoru má vliv vícero jevů, které nelze v rámci modelu trajektorií jednoduše zohlednit, zejména se jedná o:

- mnohonásobný rozptyl,
- ionizace částic a
- ztráty energie způsobené radiací.

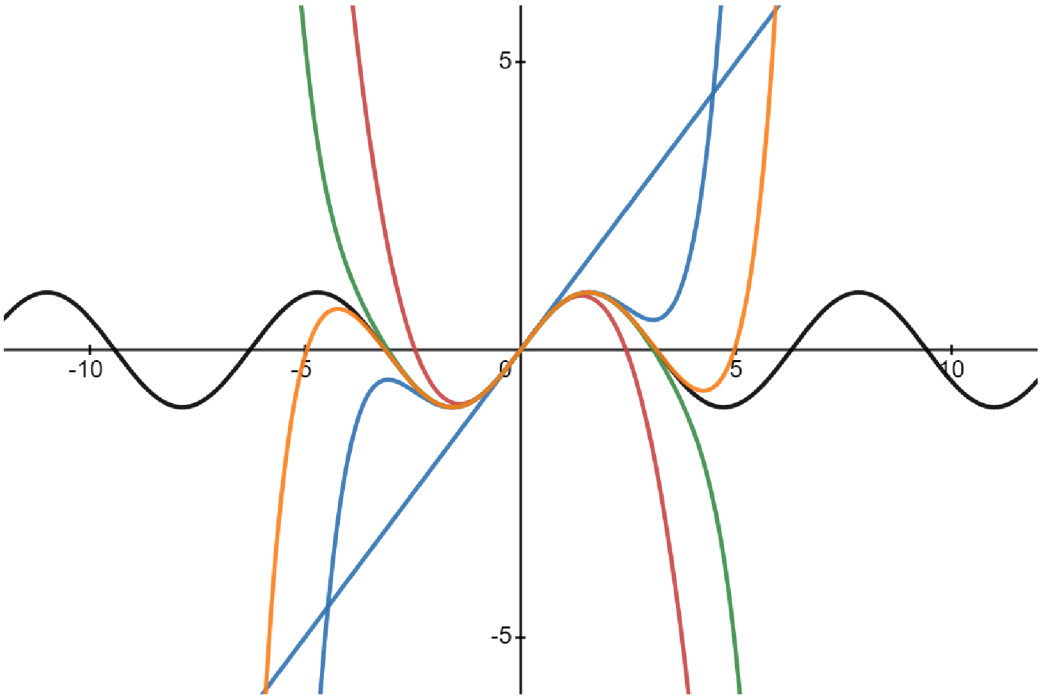
Tyto vlivy zavádějí do rekonstrukce kinematických vlastností částic perturbace a celý proces tak narušují. Použití správné rekonstrukční metody nám umožňuje dopad těchto rušivých jevů na měření minimalizovat. V dnešní době se ve většině moderních HEP experimentů používá pro odhad parametrů trajektorií částic Kalmanův Filtr.

I když se parametry trajektorií dají z jednotlivých měření odvodit pomocí *metody nejmenších čtverců*, v praxi upřednostňujeme Kalmanův filtr z důvodu, že díky své rekurzivní povaze je výpočetně mnohem výhodnější. I když obě metody počítají nad maticemi, zatímco mocnost matice u metody nejmenších čtverců závisí na počtu měření, u Kalmanova filtru na počtu stupňů volnosti systému.

Nedostatky konvenčního Kalmanova Filtru

Konvenční Kalmanův Filtr je určený pro lineární dynamické systémy s normálními rozděleními pravděpodobnosti, což znamená, že vývoj systému mezi jednotlivými měřeními $\boldsymbol{m}_{k-1}$ a $\boldsymbol{m}_k$ dokážeme popsat pomocí **lineární** rovnice (viz rovnice 3.20). V případě, že systém pracuje s lineárními modely a pouze s normálními rozloženími pravděpodobnosti, Kalmanův Filtr je **optimální metoda aproximace** stavu takového systému.

Bohužel, v praxi se často setkáváme se situacemi, kdy model závislostí mezi jednotlivými měřeními nelze popsat pomocí lineární funkce, případně stochastické jevy systému nelze modelovat jako normální rozdělení pravděpodobnosti. V případě nelineárních modelů (případ užití, který platí i pro aproximaci trajektorií částic v detektoru CBM) je potřeba nejprve najít takový lineární model, který bude co nejlépe aproximovat chování původního,



Obrázek 3.8: Funkce $y = \sin(x)$ a její Taylorovy rozvoje prvního, třetího, pátého, sedmého a devátého stupně. Každý vyšší stupeň Taylorova rozvoje aproximuje původní funkci přesněji. Vytvořeno pomocí nástroje Desmos [9].

nelineárního modelu. K nalezení takového modelu nám pomůže speciální případ aproximace pomocí **Taylorova rozvoje**, kterému říkáme **linearizace**.

Taylorův rozvoj

Mějme nepolynomiální funkci $f : \mathbb{R} \to \mathbb{R}$, (například $f(x) = \sin(x)$): Pokud potřebujeme použít takovou funkci a z jakéhokoli důvodu je pro nás nepraktické vypočítávat její výstup přesně (například pokud potřebujeme vypočítat mnoho hodnot pro vzájemně blízká x a pokaždé vypočítávat přesnou hodnotu je příliš výpočetně náročné), můžeme funkci vhodně *aproximovat* a dále používat aproximovanou funkci, která je pro náš případ užití výhodná.

Taylorova řada je mocninná řada, pomocí které můžeme za určitých předpokladů vyjádřit nepolynomiální funkci v okolí bodu x_{lin} jako polynom. Taylorovu řadu reálné, **hladké** funkce $f(x)$ zapisujeme následujícím způsobem:

$$f(x) = f(x_{lin}) + \frac{f'(x_{lin})}{1!}(x - x_{lin}) + \frac{f''(x_{lin})}{2!}(x - x_{lin})^2 + \cdots = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_{lin})}{k!}(x - x_{lin})^k \tag{3.33}$$

kde $f^{(n)}$ značí n-tou derivaci funkce f. Funkce je **hladká** tehdy, kdy nejenom funkce samotná, ale i všechny její derivace jsou spojité.

Uveďme jako příklad funkci $f(x) = \sin x$, kterou chceme aproximovat Taylorovým polynomem v bodě 0 (v případě Taylorovy řady v bodě 0 mluvíme o tzv. *Maclaurinově řadě*):

$$
\begin{aligned}
T_1^{f,0} &= f(0) + \frac{f'(0)}{1!}(x-0)^1 = 0 + \cos(0)x = x \\
T_2^{f,0} &= T_1^{f,0} + \frac{f''(0)}{2!}(x-0)^2 = x - \frac{\sin(0)}{2}x^2 = x \\
T_3^{f,0} &= T_2^{f,0} + \frac{f^{(3)}(0)}{3!}(x-0)^3 = x - \frac{\cos(0)}{6}x^3 = x - \frac{x^3}{6} \\
T_4^{f,0} &= T_3^{f,0} + \frac{f^{(4)}(0)}{4!}(x-0)^4 = x - \frac{x^3}{6} + \frac{\sin(0)x^4}{4!} = x - \frac{x^3}{6}
\end{aligned}
\tag{3.34}
$$

Můžeme si všimnout, že sudé stupně Maclaurinovy řady se u funkce $f(x) = \sin x$ vždy rovnají nule, díky čemuž víme, že nám pro její aproximaci stačí liché stupně rozvoje:

$$
\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \tag{3.35}
$$

pro $x \in (-\infty, \infty)$. Analogicky můžeme zapsat například i Maclaurinův rozvoj funkce $f(x) = \cos x$, kde jsou nenulové naopak sudé členy rozvoje. Jednotlivé polynomy jsou ukázány na obrázku 3.8.

Linearizace je speciální případ aproximace Taylorovým rozvojem, kdy aproximujeme pouze prvním stupněm Taylorova polynomu (tj. přímkou):

$$
L(x)_f = f(x^{lin}) + f'(x^{lin})(x - x^{lin}) \tag{3.36}
$$

Taková lineární funkce nám poskytuje vyhovující aproximaci pro x blízká x^{lin}. Mějme funkci $f(x) = \sqrt{x}$, a hledejme hodnotu pro $\sqrt{4,41}$: Víme, že $\sqrt{4} = 2$, a tak vhodně zvolíme $x^{lin} = 4$:

$$
L(x) = \sqrt{4} + \frac{1}{2\sqrt{4}}(x-4) = 1 + \frac{x}{4} \tag{3.37}
$$

Linearizovaná funkce $L(x)$ aproximuje $\sqrt{4,41} \approx 2,1025$, přesné řešení je přitom $\sqrt{4,41} = 2,1$. Linearizace nám tedy poskytuje solidní aproximaci pomocí relativně jednoduchého výpočtu. Pokud ale chceme linearizaci použít pro funkce, které pracují nad vektory, narazíme na problém, jak získat první derivaci matice. K tomuto účelu nám poslouží tzv. **Jacobiho matice**.

Jacobiho matice

Mějme funkci $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$. Jacobiho matice J je **matice parciálních derivací** takové funkce ve tvaru $m \times n$:

$$
J = \begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n}
\end{bmatrix} \tag{3.38}
$$

Jacobiho matice nám pomůže provádět linearizaci nelineárních modelů pro použití v rámci Kalmanova Filtru. Této verzi algoritmu pak říkáme **Rozšířený Kalmanův Filtr (EKF)**[8].

[8] EKF – Extended Kalman Filter (rozšířený Kalmanův Filtr)

Rozšířený Kalmanův filtr

Rozšířený Kalmanův Filtr, jak název napovídá, je rozšíření původní metody Kalmanova Filtru, které pracuje nad nelineárními modely měření a/nebo modely vývoje systému.

Rozšířený Kalmanův Filtr sice umožňuje pracovat s nelineárními modely, z tohoto faktu ale plyne také jeho zásadní nevýhoda: Kvůli procesu linearizace není EKF optimální algoritmus pro odhad stavu systému, a při nevhodně zvolených hodnotách počátečního odhadu systému nebo linearizačního bodu $\boldsymbol{r}_{k-1}^{lin}$ může metoda divergovat, je tedy velmi důležité tyto hodnoty zvolit vhodně.

V případě rozšířeného Kalmanova Filtru musíme některé rovnice pozměnit tak, aby mohly pracovat nad nelineárními systémy. V případě, že pracujeme s nelineárním modelem vývoje systému $\boldsymbol{f}_k$, potřebujeme rovnici 3.20 upravit následujícím způsobem:

$$\boldsymbol{r}_k^- \equiv \boldsymbol{f}_k(\boldsymbol{r}_{k-1}^+) \approx \boldsymbol{f}_k(\boldsymbol{r}_{k-1}^{lin}) + F_k(\boldsymbol{r}_{k-1}^+ - \boldsymbol{r}_{k-1}^{lin}) \tag{3.39}$$

kde $\boldsymbol{f_k}$ je **nelineární** model vývoje systému, $\boldsymbol{r}_{k-1}^{lin}$ je linearizační bod, typicky zvolený podle minulého stavu systému, a F_k je Jacobiho matice nelineárního modelu vývoje $\boldsymbol{f}_k$:

$$F_{k(ij)} = \left.\frac{\partial \boldsymbol{f}_k(\boldsymbol{r}_{k-1}^+)_{(i)}}{\partial \boldsymbol{r}_{k-1(j)}^+}\right|_{\boldsymbol{r}_{k-1}^+ = \boldsymbol{r}_{k-1}^{lin}} \tag{3.40}$$

Podobně potřebujeme upravit rovnici vztahu mezi měřením a reálným stavem systému (rovnice 3.21) v případě nelineárního modelu měření:

$$\boldsymbol{m}_k(\boldsymbol{r}_k^t) \equiv \boldsymbol{h}_k(\boldsymbol{r_k^t}) + \boldsymbol{\eta}_k \approx \boldsymbol{h}_k(\boldsymbol{r}_k^{lin}) + H_k(\boldsymbol{r}_k^t - \boldsymbol{r}_k^{lin}) + \boldsymbol{\eta}_k \tag{3.41}$$

Kde $H_{k(ij)}$ je Jacobiho matice modelu měření v následujícím tvaru:

$$H_{k(ij)} = \left.\frac{\partial \boldsymbol{h}_k(\boldsymbol{r}_k)_{(i)}}{\partial \boldsymbol{r}_{k(j)}}\right|_{\boldsymbol{r}_{k-1}^+ = \boldsymbol{r}_{k-1}^{lin}} \tag{3.42}$$

V případě nelineárního modelu měření se také změní rovnice pro výpočet rezidua:

$$\boldsymbol{\zeta} = \boldsymbol{m}_k - (\boldsymbol{h}_k(\boldsymbol{r}_k^{lin}) + H_k(\boldsymbol{r}_k^- - \boldsymbol{r}_k^{lin})) \tag{3.43}$$

Již jsme zmínili, že rozšířený Kalmanův filtr může mít v některých případech problémy se stabilitou. Z použití rozšířeného Kalmanova filtru vyplývají některé potíže s implementací stabilního a efektivního algoritmu a jeho následnou akcelerací na různých hardwarových platformách. Některými možnými řešeními se budeme dále zabývat v kapitole 4.

3.3 Metody rekonstrukce trajektorií

Časově nejnáročnější část rekonstrukce trajektorií je jejich rekonstrukce, proto je potřeba vhodná volba sledovacího algoritmu. Náročnost této části analýzy událostí v detektoru vychází z faktu, že v této fázi algoritmus pracuje s hrubými, dosud nijak nezpracovanými daty a objem dat, který musí algoritmus zpracovat nelze nijak omezit. Často se tedy stává, že algoritmus musí v této fázi zpracovávat data až o objemu jednotek TB/s. Zároveň je časté, že v závislosti na konkrétním experimentu vyžadujeme, aby byla data zpracována v reálném čase, což úkol vývoje rekonstrukčního programu dále komplikuje a je tedy velmi důležité takový algoritmus vhodně zvolit. Situaci dále komplikuje fakt, že každý detektor je jiný, je

tedy v případě každého detektoru potřeba výběr algoritmu přizpůsobit konkrétním vlastnostem detektoru. V následující sekci shrneme několik takových algoritmů, které můžeme pro rekonstrukci trajektorie zvolit:

- Houghova transformace,
- konformní zobrazování,
- sledování stop a
- celulární automaty.

Právě metodou rekonstrukce trajektorií pomocí celulárních automatů se pak budeme podrobněji zabývat v kontextu experimentu CBM.

Důležitým faktorem při výběru vhodného algoritmu je otázka, jak moc je možné problém rekonstrukce trajektorií zjednodušit, tedy jaké zjednodušující předpoklady, které můžeme zavést, usnadní (popřípadě umožní zrychlení např. paralelizací algoritmů tam, kde to u obecného problému bylo nemožné nebo nepraktické) řešení daného problému, aniž by (příliš) snížil přesnost výpočtu. Pro nás je pravděpodobně nejdůležitější vědět, jestli pro rekonstrukci trajektorií daná metoda pracuje se všemi vstupními daty, nebo jestli dokáže pracovat i pouze nad nějakou jejich podmnožinou. Pokud metoda vyžaduje přístup ke všem vstupním datům, říkáme, že mluvíme o **globální** metodě, v opačném případě mluvíme o **lokální** metodě. Pro naše účely je výhodné pracovat s metodami lokálními, protože se přirozeně nabízí možnost metodu do jisté míry paralelizovat. Níže se budeme zabývat dvěma lokálními a dvěma globálními metodami zmíněnými výše: Houghova transformace a konformní mapování jsou příklady **globálních** metod, zatímco sledování stop a metoda celulárních automatů jsou příklady **lokálních** metod.

Houghova transformace

Houghova transformace je technika extrakce prvků, původně navržena Paulem V. C. Houghem za účelem strojové analýzy fotografií bublinkových komor [20]. Ve své generalizované formě [10] je hojně používaná v analýze obrazu, počítačovém vidění nebo zpracování obrazu. Cílem metody je najít nedokonale vyobrazené objekty (zpravidla objekty stejného charakteru, například přímky nebo kružnice) pomocí takzvaného **hlasování**.

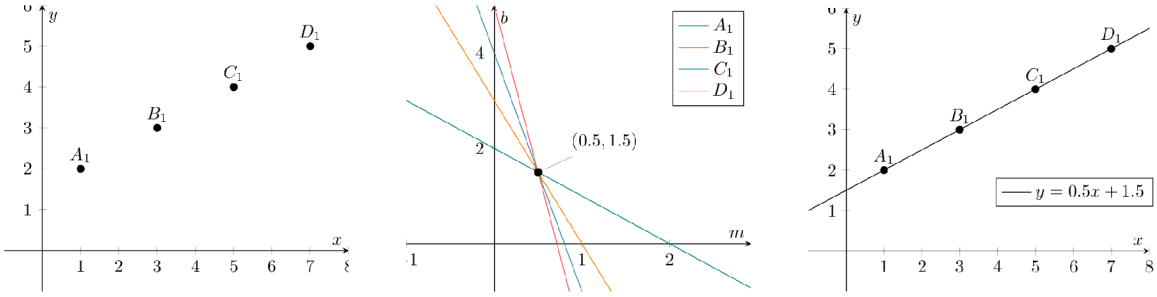
Mějme nejjednodušší verzi Houghovy transformace, a to takovou, která detekuje v obraze přímky: Přímku definujeme jednoduše pomocí rovnice

$$y = mx + b$$

v důsledku čehož můžeme přímku zapsat jako uspořádanou dvojici parametrů (m, b), což je informace, která jednoznačně definuje danou přímku jako bod (m, b) v takzvaném **parametrovém prostoru**[9].

Uveďme jako příklad hledání přímek pomocí Houghovy transformace následující situaci, znázorněnou na obrázku 3.9: mějme přímku v reálném prostoru, definovanou funkcí $y = 0.5x + 1.5$, o které máme informaci v podobě čtyř bodů: $A_1 = (1, 2)$, $B_1 = (3, 3)$, $C_1 = (5, 4)$, $D_1 = (7, 5)$. Tak, jako můžeme zapsat jakoukoli přímku jako bod v parametrovém prostoru, můžeme obdobným způsobem pracovat s jednotlivými body. Každý bod z reálného prostoru se promítne do parametrového prostoru jako přímka, která obsahuje body

[9] z angl. parameter space



Obrázek 3.9: Příklad Houghovy transformace. Body v **reálném prostoru** (obrázek vlevo) jsou převedeny do **parametrového prostoru** jako přímky. Pokud se všechny body nachází na jedné přímce v reálném prostoru, budou mít jim odpovídající přímky v parametrovém prostoru právě jeden průsečík (obrázek uprostřed). Průsečík v parametrovém prostoru obsahuje parametry přímky, na které se všechny tyto body nachází (obrázek vpravo). Obrázky převzaty z [26].

s parametry všech přímek, které v reálném prostoru procházejí daným bodem. Všechny čtyři body tak vyneseme jako přímky v parametrovém prostoru a pozorujeme jejich podobu: Z povahy Houghovy transformace vyplývá, že všechny přímky v parametrovém prostoru, které mají v parametrovém prostoru společný průsečík, leží v reálném prostoru na stejné přímce. Průsečík těchto přímek v parametrovém prostoru nám zároveň sděluje, jaké jsou parametry takové přímky, v případě příkladu na obrázku 3.9 bodu $(0.5, 1.5)$ v parametrovém prostoru odpovídá přímka $y = 0.5x + 1.5$, která prochází všemi body A_1 až D_1.

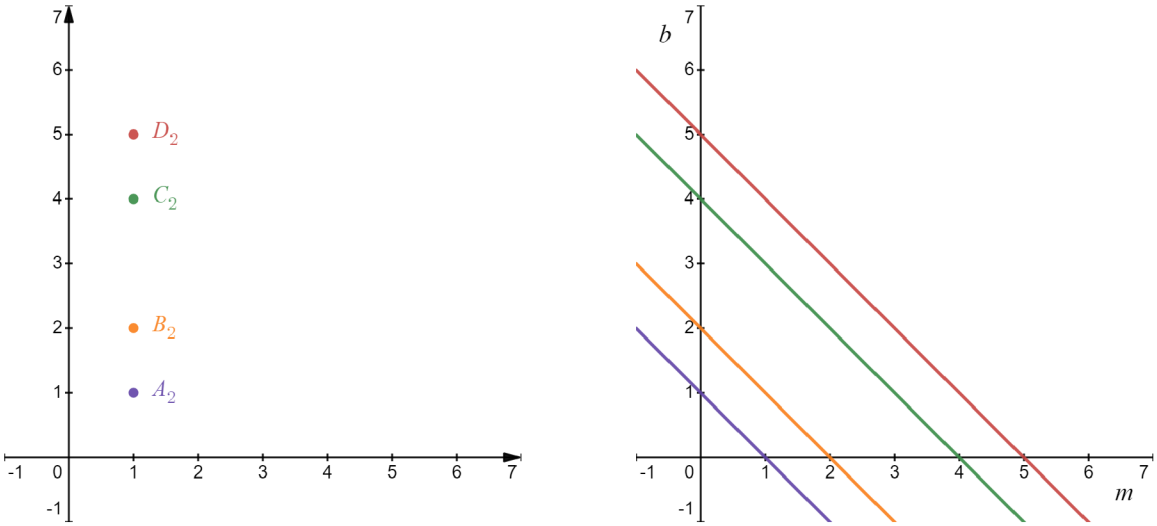
Tento přístup má ale nevýhodu v případě, že chceme takto zapsat svislou přímku, což by mělo za následek parametr m jdoucí k nekonečnu. Když se pokusíme převést do parametrového prostoru body, které jsou přímo nad sebou (tj. mají stejnou x souřadnici, viz obrázek 3.10), zjistíme, že jsou všechny odpovídající přímky rovnoběžné a přestože lze všemi body vést jednu přímku, v parametrovém prostoru žádný průsečík nenajdeme.

Tento nedostatek Houghovy transformace můžeme vyřešit tak, že změníme způsob, jakým přímku popisujeme: Jedním z řešení je použít pro popis přímky takzvanou **Hesseho normální formu**: Prvním parametrem ρ udáváme vzdálenost přímky od počátku souřadného systému (jinými slovy délku úsečky, která je ohraničena počátkem souřadného systému a bodem na popisované přímce a která svírá s popisovanou přímkou pravý úhel), druhým parametrem θ pak popisujeme úhel mezi osou x a úsečkou, spojující počátek souřadného systému s nejbližším bodem popisované přímky [10]:

$$\rho = x \cos\theta + y \sin\theta \tag{3.44}$$

Uveďme opět příklad se svislou přímkou z obrázku 3.10: Vzhledem k tomu, že používáme jinou transformaci, body se již nebudou do parametrového prostoru promítat jako přímky, nýbrž jako sinusovky. V bodě, kde se všechny sinusovky protnou, můžeme analogicky jako u předchozích příkladů vyčíst parametry přímky, na které se body nachází (viz. obrázek 3.12).

Všimněme si u transformace v normální formě, že se průsečíky jednotlivých sinusovek periodicky opakují: Tento jev pozorujeme z toho důvodu, že přímka se pomocí Hesseho normální formy dá vyjádřit vícero způsoby, což vyplývá z faktu, že úsečka ρ může s osou x svírat úhly $\theta + 2k\pi$ radiánů (první a třetí průsečík v obrázku 3.12), popřípadě $\theta + (2k + 1)\pi$ radiánů (druhý průsečík, v takovém případě nám ale délka úsečky ρ vyjde „záporná“).



Obrázek 3.10: Příklad nedostatku Houghovy transformace. Obrázek vpravo zobrazuje body na jedné svislé přímce v reálném prostoru, obrázek vpravo zobrazuje tyto body v parametrovém prostoru. Přestože body náleží jedné přímce, v parametrovém prostoru nemají jednotlivé přímky průsečík. Vytvořeno pomocí nástroje Desmos [9].

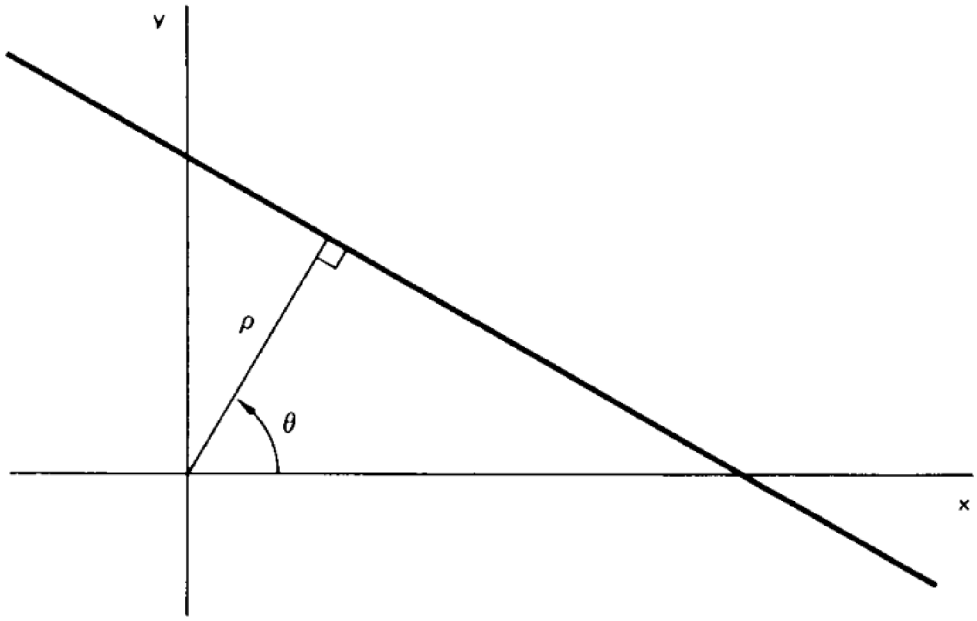
Generalizovaná forma Houghovy transformace, jak píše Duda a kolektiv [10], může krom přímek v obraze hledat jakýkoli geometrický útvar, který lze popsat dvěma parametry: pro nás může být relevantní například hledání kružnic: mějme například k dispozici čtyři body, které náleží jedné kružnici: $A_4 = (1, 2), B_4 = (2, 1), C_4 = (2, 3), D_4 = (3, 2)$. Rovnice pro popis kružnice má následující tvar:

$$r^2 = (x - h)^2 + (y - k)^2 \tag{3.45}$$

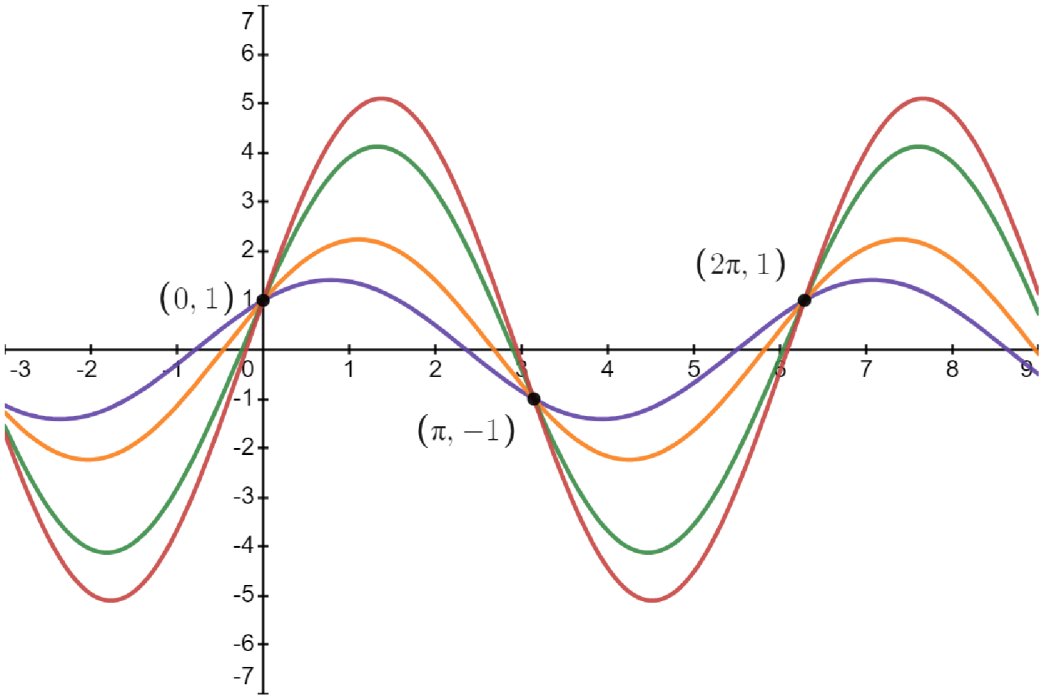
kde r značí poloměr kružnice a bod (h, k) značí její střed. V tomto případě přichází drobná komplikace, protože kružnici definujeme pomocí tří parametrů. Tento problém obvykle vyřešíme tak, že jeden z hledaných parametrů známe. Pro náš příklad řekněme, že víme, že kružnice má poloměr $r = 1$: parametrový prostor tak zůstává dvojrozměrný a můžeme tak promítnout jednotlivé body do parametrového prostoru jako kružnice. V bodě, kde se všechny kružnice v parametrovém prostoru protnou, najdeme parametry pro kružnici v reálném prostoru (konkrétně pak polohu středu kružnice).

Houghovu transformaci můžeme využít pro rekonstrukci trajektorií následujícím způsobem: V parametrovém prostoru vytvoříme histogram, kde zaznamenáváme, kolik obrazů bodů v parametrovém prostoru prochází každým bodem. Poté postupujeme globálně po celém prostoru a hledáme body s největším počtem procházejících přímek. Z takových bodů postupně vyčítáme parametry pravděpodobných trajektorií a body v reálném prostoru, které se do daného bodu v parametrovém prostoru promítly, vyjmeme a proces opakujeme. Tímto způsobem zároveň potlačujeme šum, protože šumem vzniklé „falešné“ zásahy zpravidla nebudou tvořit v parametrovém prostoru průsečíky.

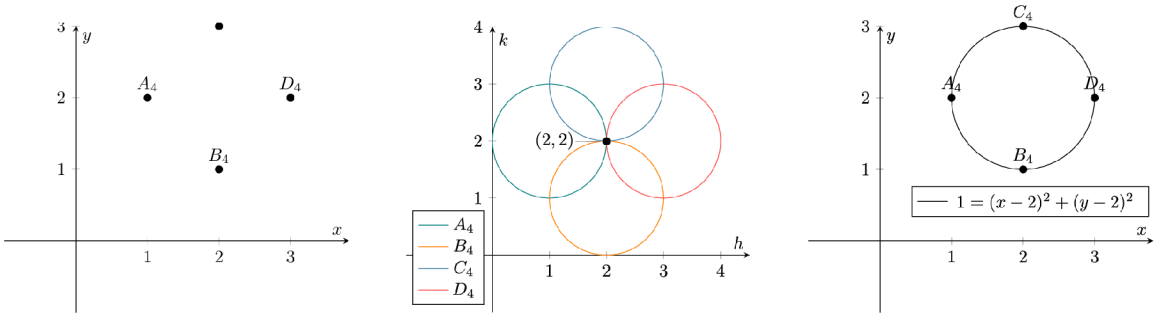
Zbytek této podsekce čerpá z práce Dr. Akishiny [1]. Tato metoda má ale zásadní nevýhodu, která spočívá v potřebě algoritmu pracovat globálně nad celým stavovým prostorem, což vnáší dva zásadní problémy pro implementaci: Za prvé, transformace celého prostoru detektoru do parametrového prostoru je velmi paměťově náročný úkol (přibližně 1.2GB paměti), což přináší problém se *svapováním adresního prostoru*, kdy vyhodnocení jedné



Obrázek 3.11: Vizualizace parametrů Houghovy transformace v Hesseho normální formě. Délka úsečky ρ značí nejkratší vzdálenost přímky od počátku souřadnicového systému, θ je úhel mezi úsečkou ρ a osou x (viz rovnice 3.44). Převzato z [10].



Obrázek 3.12: Parametrový prostor Houghovy transformace bodů z obrázku 3.10. Vytvořeno pomocí nástroje Desmos [9].



Obrázek 3.13: Verze generalizované Houghovy transformace, hledající kružnice. Body, náležící jedné kružnici (obrázek vlevo) promítneme do parametrového prostoru (obrázek uprostřed). Jejich průsečík udává střed hledané kružnice (vpravo). Převzato z [26].

události v detektoru zabralo až 15 minut. Za druhé, globálnost a paměťová náročnost algoritmu dále omezuje možnosti akcelerace na hardwarových platformách a výrazně komplikuje jeho paralelizaci. Pro naše účely je proto Houghova transformace bohužel nevhodná.

Konformní zobrazování

Další příklad globální metody rekonstrukce trajektorií je metoda konformního zobrazení. Konformní zobrazení je takové spojité zobrazení $f : \mathbb{R}^n \to \mathbb{R}^n$, které zachovává úhly, tedy platí, že velikost všech úhlů v zobrazení f zůstává stejná: $\forall\alpha(f(\alpha) = \alpha)$. Hlavní cíl konformního zobrazování je zjednodušit hledání jednotlivých trajektorií tak, že použijeme takovou projekci, která každou trajektorii transformuje jako přímku v konformním zobrazení. Částice v homogenním magnetickém poli nabývají zpravidla kruhové trajektorie čehož můžeme pomocí konformního zobrazení využít. Analytickou rovnici kružnice (viz rovnice 3.45) transformujeme do konformního zobrazení následujícím způsobem:

$$\begin{aligned} u &= \frac{x}{x^2 + y^2}, \\ v &= \frac{y}{x^2 + y^2} \end{aligned} \tag{3.46}$$

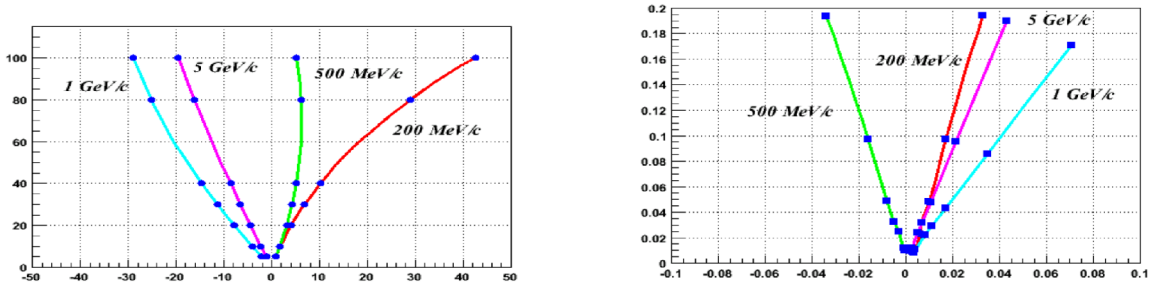
Kde u, v jsou koordináty v konformním zobrazení. Hned v tento okamžik narazíme na první nevýhodu této metody: Rovnice 3.46 totiž předpokládá, že počátek oblouku trajektorie je v bodě $(0, 0)$, což přináší potíže s rekonstrukcí sekundárních částic (tj. částic, které vznikly později, než během prvotní reakce na terči). Pro tento okamžik ale předpokládejme, že počátek trajektorie částice je uprostřed terče (tj. bod $(0, 0)$). Z tohoto předpokladu vyplývá, že poloměr oblouku trajektorie určíme pomocí vztahu:

$$R^2 = a^2 + b^2 \tag{3.47}$$

kde $(a, b) = (h, k)$ jsou koordináty středu kružnice (viz rovnice 3.45). Kružnice v konformním zobrazení tedy můžeme popsat jako přímky v následujícím tvaru:

$$v = \frac{1}{2b} - u\frac{a}{b} \tag{3.48}$$

Přestože je metoda konformního zobrazení rychlá a snadná na implementaci, její nevýhoda spočívá v příliš velkém množství zjednodušení systému: Za prvé metoda předpokládá



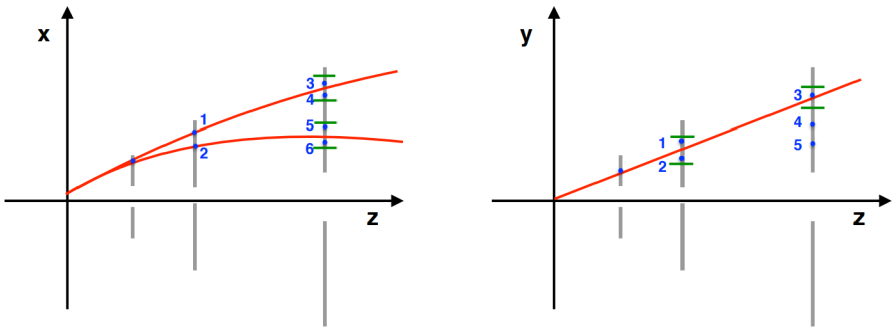
Obrázek 3.14: Trajektorie částic a jejich zásahy v reálném prostoru (vlevo), a jejich projekce pomocí konformního zobrazení jako přímky (vpravo). Převzato z [1].

homogenní magnetické pole, což v praxi zpravidla nebývá pravda, za druhé metoda nijak neřeší trajektorie sekundárních částic (tedy částice, které nemají počátek v terči). Z tohoto důvodu je metoda sice zajímavá (a pro některé případy užití velmi výhodná), pro náš případ užití se bohužel nehodí.

Sledování stop

Hlavní myšlenkou metody sledování stop[10] je předpověď polohy následujícího zásahu zkoumané trajektorie za předpokladu, že známe lokální model vývoje systému a polohy zásahů na předchozích hladinách detektoru.

Metoda spočívá v předpovědi polohy zásahu v následující detektorové hladině na základě lokálního modelu systému a zásahu v předchozí detektorové hladině. Oblast hledání je dána jako oblast okolo předvídané příští polohy, která je ovlivněna přesností (resp. rozlišením) detektoru, přesnosti modelu systému a stochastickými procesy v systému, zejména pak jevem mnohonásobného rozptylu, ke kterému v každém detektoru (a jejich jednotlivých hladinách) dochází v různé míře v závislosti na šířce a materiálu detektorových hladin.



Obrázek 3.15: Lokální metoda sledování stop v projekcích XZ a YZ. Červené linie značí možné kandidáty na trajektorie, které metoda vytvořila, šedé svislé úsečky značí detektorové hladiny, modré body značí jednotlivé zásahy a zelené linie značí hranice tolerance algoritmu. Převzato z [1]

[10] z angl. track following

Algoritmus pracuje ve dvou různých (XZ a YZ) projekcích, které průběžně střídá: počínaje ve středu terče detektoru (tedy v koordinátách $[0, 0, 0]$) program nejdříve spojí tento bod se všemi zásahy v první detektorové hladině, protože všechny trajektorie by v projekci YZ měly tvořit trajektorii blízkou přímce. Vycházíme z faktu, že magnetické pole v detektoru je orientováno ve směru osy Y, takže na pohyb částice v této projekci nebude mít žádný vliv (Lorenzova síla působí kolmo na směr magnetického pole, k zakřivení tedy bude, za předpokladu homogenního magnetického pole, docházet pouze ve směru XZ). Přímky takto definované pomocí zásahu na první hladině a středu terče detektoru následně můžeme použít jako směrnici ve směru YZ tak, že promítneme každou z těchto přímek v projekci YZ na druhou hladinu detektoru, a pokud se tyto přímky promítly (s určitou tolerancí) na zásah v druhé detektorové hladině, vytvoříme dvojici zásahů na této přímce. V dalším kroku tohoto algoritmu vezmeme všechny takto vytvořené dvojice zásahů a takto vzniklou trojicí zásahů (zásah na druhé hladině, zásah na první hladině, střed terče detektoru) proložíme v projekci XZ **parabolu** a promítneme ji do třetí hladiny, čímž získáme předpokládané parametry trajektorie, které můžeme dále promítat na další detektorové hladiny a pokud v předpokládané poloze sledujeme zásah, předpokládáme, že patří k dané trajektorii.

Zbytek této podsekce čerpá z práce Dr. Akishiny [1]. Velký problém této metody spočívá v tom, že s rostoucím počtem částic v detektoru množství kombinací možných trajektorií exponenciálně roste. Algoritmus je tedy efektivní jen v případě, že pracujeme s rozumnou hustotou zásahů v detektoru, což neplatí v případě experimentu CBM. Další problém spočívá v tom, že algoritmus ze své podstaty vyžaduje náhodný přístup do paměti, což je časově velmi náročný úkon a způsobuje tak značné zpomalení, nehledě na fakt, že vzhledem k tomu, že program po přístupu do paměti data zahazuje, je stejný výpočet často zbytečně proveden nad stejnými daty několikrát.

Experiment CBM v raných fázích vývoje počítal s odlišnou podobou STS detektoru (předpokládal se tzv. pixel detector), a pro tuto podobu detektoru byl jako rekonstrukční algoritmus nejdříve vybrán algoritmus sledování stop. Detektor STS byl ale později přepracován jako moduly dvoustranných polovodičových proužků[11], což mělo za důsledek výrazné zvýšení počtu falešných zásahů, což má za následek výrazné zvýšení náročnosti kombinatorického vyhledávání. Tato metoda je proto pro náš případ užití považována za nevhodnou.

Celulární automat

Celulární automat (CA[12]) je souhrnné označení pro typicky diskrétní výpočetní model, studovaný v rámci **teorie automatů**. Jedná se o překvapivě multifunkční jednoduchý nástroj, hojně používaný pro simulace v oblastech dopravy, epidemiologie, fyziky a mnoho dalších.

Celulární automat můžeme zpravidla definovat pomocí těchto čtyř kritérií:

- **pole buněk**, které určuje rozměr celulárního automatu. Pole buněk může mít libovolný počet rozměrů, obvykle jeden nebo dva. Dále rozlišujeme, jestli je pole nekonečné, nebo ohraničené (v případě, že je ohraničené, rozlišujeme, jak jsou definovány stavy hranic pole). V neposlední řadě určujeme tvar buněk v poli (například čtvercové, šestiúhelníkové, krychlové u 3D atd.), což dále ovlivňuje tvar okolí každé buňky,

[11] z angl. double sided strip detector
[12] CA – Cellular Automaton (Buněčný automat)

- **Množina stavů buňky (S)**, automat může počítat s libovolným **konečným** počtem možných stavů, kterých mohou jednotlivé buňky nabývat (v nejjednodušším případě například dvojice stavů 0,1),
- **okolí buněk** určuje, které buňky (a jejich stavy) bude každá buňka brát v potaz, když bude rozhodovat o svém příštím stavu. Rozlišujeme například Mooreovo okolí (čtvercové okolí 8 buněk, majících s buňkou společnou stranu nebo vrchol), rozšířené Mooreovo okolí (stejné, jako Mooreovo, rozšířené o Mooreovo okolí všech osmi sousedů, dohromady tedy 24 buněk v okolí), nebo například Von-Neumannovo okolí (pouze čtyři buňky, které mají společnou stranu s buňkou patří do jejího okolí),
- **přechodová funkce**: pravidla, která určují, jak na základě stavu a okolí buňky vypočítat příští stav buňky. Všechny změny stavu probíhají zároveň na základě posledních stavů pole buněk.

Nejznámější příklad celulárního automatu je hra „Život“ Johna Conwaye. Následující odstavec čerpá ze článku Martina Gardnera a skript Jarkka Kari [15, 22]: Hra „Život“ je celulární automat s nekonečným 2D polem buněk o dvou možných stavech (0 – „mrtvá“ a 1 – „živá“) a jednoduchým Mooreovým okolím. Pravidla pro hru „Život“ byla Conwayem navržena tak, aby splňovala následující 3 požadavky:

1. Neměla by existovat žádná počáteční konfigurace, pro kterou by existoval jednoduchý důkaz, že populace (počet „živých“ buněk, nesoucích hodnotu 1) může neomezeně růst.
2. Měly by existovat počáteční konfigurace, které *zřejmě* způsobují neomezený růst populace.
3. Měly by existovat počáteční konfigurace, které rostou a vyvíjejí se po delší dobu, než se doberou k jednomu ze tří možných konců: úplně „vymřou“ (tj. v určitý čas nezbude žádná „živá“ buňka, ať už kvůli příliš vysoké, nebo příliš nízké hustoty „zalidnění“), ustálí se ve stabilním rozložení, které se dále již nezmění, nebo se ustálí v oscilující konfiguraci, během které se s určitou periodou donekonečna opakuje konečné množství konfigurací.

Tyto požadavky mají zařídit, že se samotný celulární automat bude chovat **nepředvídatelně**. Conway tedy navrhl pravidla vývoje populace, která Martin Gardner popsal jako „rozkošně jednoduchá“ [15].

Conwayova hra „Život“ je příklad tzv. **totalistického celulárního automatu**, což znamená, že nezáleží na konkrétní poloze „živých“ a „mrtvých“ buněk v rámci okolí buňky, což snižuje počet pravidel ze $2^9 = 512$ (počet možných konfigurací Mooreova okolí krát dva, protože pravidla jsou různá pro „živou“ a „mrtvou“ buňku) na $2 \times 9 = 18$ (výběr pravidla je závislý pouze na stavu buňky a počtu „živých“ buněk).

Pravidla celulárního automatu tak můžeme definovat následovně:

1. Přežití: Pokud je buňka živá a má právě dva nebo tři živé sousedy, přežívá do další evoluce
2. Smrt: Každá živá buňka, která má čtyři a více živé sousedy, zemře kvůli přelidnění. Taktéž každá živá buňka, která má pouze jednoho nebo žádného živého souseda, zahyne samotou.

3. Zrození: Každá „neživá“ buňka, která má **právě** tři živé sousedy, se „narodí“ a v příští evoluci algoritmu bude živá.

V průběhu let hra Život přilákala velké množství nadšenců, kteří postupně shromáždili rozměrnou knihovnu vzorů s různými chováními, které lze kategorizovat pomocí mezi nadšenci uznávané terminologie [22]:

- *still life* (česky *stabilní život*): stabilní vzor, ve kterém se nerodí žádné nové buňky a žádné naopak neumírají, vzor tedy v této konfiguraci, pokud není zvenčí narušen zůstane donekonečna. Nejjednodušší představitel tohoto vzoru je **blok**, čtyři živé buňky, poskládané do 2×2 čtverce. Další příklad je ukázán na obrázku 3.16(a).

- *oscilátor*: Periodická konfigurace. Konfigurace se může mezi jednotlivými generacemi měnit, avšak po n generacích se vrátí do původní konfigurace, říkáme tedy, že vzor má periodu n. Můžeme také tvrdit, že stabilní život je oscilátor s periodou 1. Nejmenší příklad oscilátoru je takzvaná *blikačka* (anglicky *blinker*), která se skládá ze tří sousedících buněk v jedné přímce. S každou další evolucí se změní ze svislé na vodorovnou a naopak. Další příklad oscilátoru (takzvaná *žába*) je vyobrazen na obrázku 3.16(b).

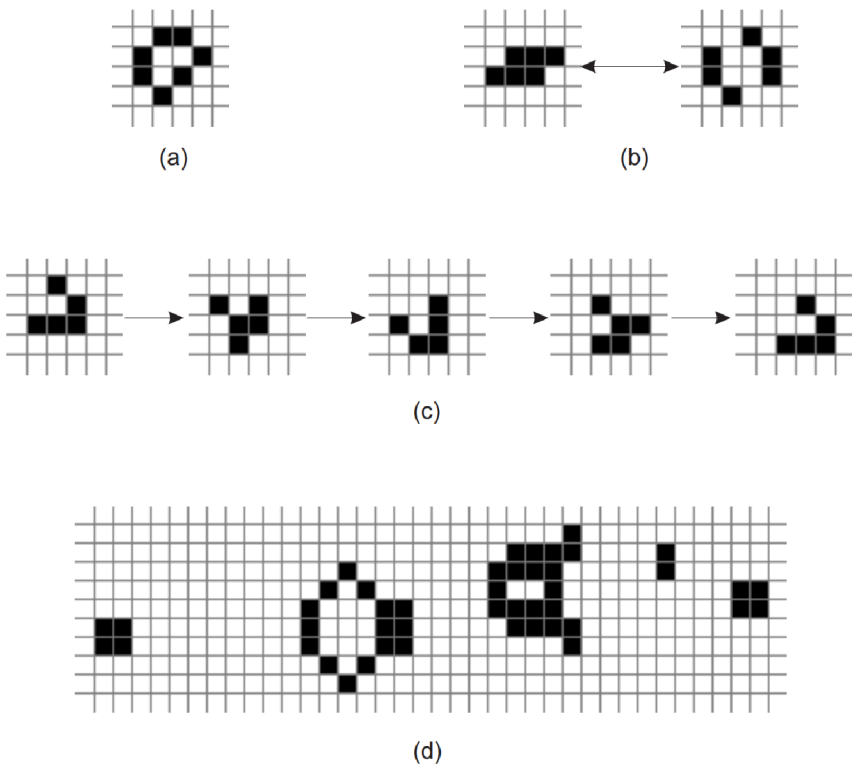
- *vesmírné lodě*: Konfigurace, která se s určitou periodou objeví znovu, oproti oscilátorům se liší v tom, že se znovu objeví na jiném místě v poli, působí tak, že „cestuje“. Na obrázku 3.16(c) vidíme nejjednodušší vesmírnou loď, takzvaný *kluzák*.

- *dělo*: oscilující konfigurace, která se stejně jako oscilátor s určitou periodou vrátí do původní podoby, na rozdíl od oscilátoru ale během této periody vygeneruje vesmírnou loď. Na obrázku 3.16(d) můžeme vidět takzvané *kluzákové dělo*.

Navzdory své jednoduchosti vykazuje hra Život natolik sofistikovanou vlastnost jako *sebeorganizace*: I když celulární automat inicializujeme v chaotické počáteční konfiguraci, s postupem času začne metoda vykazovat organizovanou strukturu v podobě (zpravidla) oscilátorů a stabilních vzorů. Tuto vlastnost můžeme využít pro rekonstrukci částic, zejména pro potlačení šumu. Můžeme se na hru Život (a celulární automaty obecně) dívat jako na lokální verzi **rekurentních neuronových sítí (RNN)**[13]. Navíc, díky jejich lokální povaze, jsou celulární automaty vhodné pro paralelní přístup, kdy v extrémním případě můžeme na každou buňku nahlížet jako na samostatné vlákno programu [1].

Celulární automaty jsou v dnešní době hojně využívány pro rekonstrukci trajektorií v mnoha HEP experimentech, mezi nimi ALICE[14], HERA-B[15], K2K[16], LHCb[17], NEMO[18], STAR[19] nebo ARES[20], který byl prvním experimentem, který používá celulární automaty pro rekonstrukci trajektorií a konfigurace automatu je velmi podobná Conwayově původní hře Život. Každý z experimentů má jiné požadavky na algoritmus rekonstrukce trajektorií, tak i jednotlivé celulární automaty jsou rozdílné. My se budeme dále zabývat celulárním automatem použitým v experimentu CBM.

[13] RNN – Recurrent Neural Network (rekurentní neuronová síť)
[14] ALICE – A Large Ion Collider Experiment
[15] HERA – Hadron-Elektron-RingAnlage (Hadron-Elektronový prstencový urychlovač)
[16] K2K – KEK to Kamioka
[17] LHCb – Large Hadron Collider beauty experiment
[18] NEMO – Neutrino Ettore Majorana Observatory
[19] STAR – Solenoidal Tracker
[20] ARES – Automotive Research Experiment Station



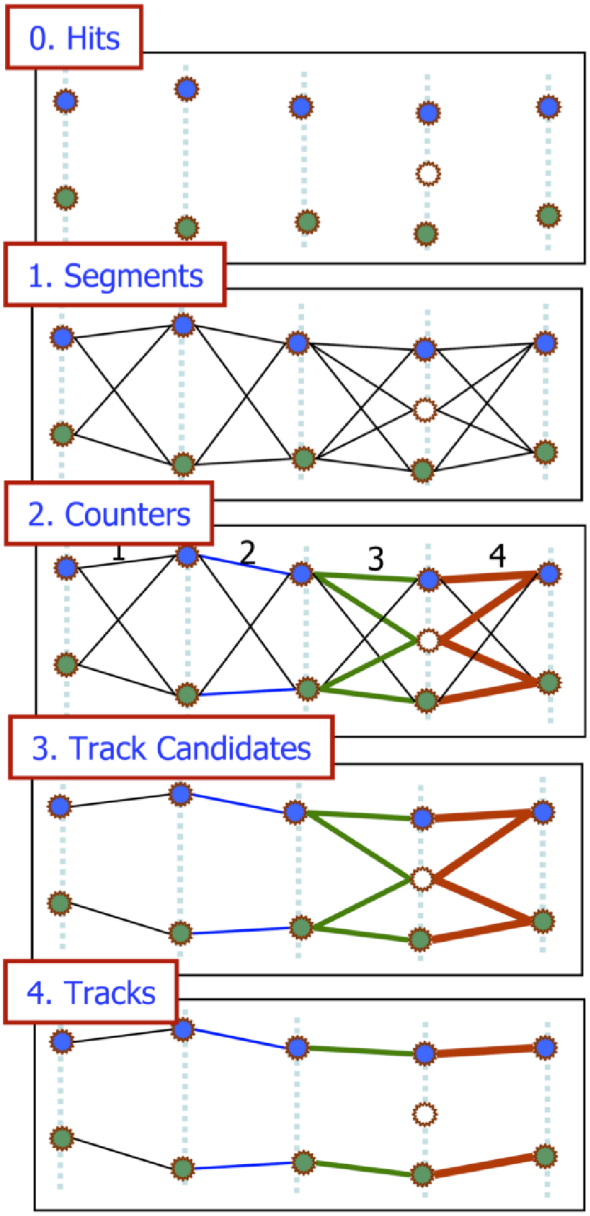
Obrázek 3.16: Vybrané objekty Conwayovy hry Život: stabilní život (a), oscilátor (b), kluzák (c), kluzákové dělo (d). Převzato z [22].

Celulární automat pro experiment CBM

Následující podsekce čerpá z práce Dr. Akishiny [1]. Celulární automat na podobné bázi jako Conwayova hra Život sice funguje dobře pro relativně nízké množství trajektorií, jako je experiment ARES, pro detektory s výrazně vyšším počtem trajektorií jako je CBM budeme ale nuceni zavést velké změny do tohoto algoritmu. V prvé řadě je potřeba výrazně změnit pohled na pojem buňky automatu: Buňka by měla být schopna vyjadřovat co největší množství parametrů možné trajektorie. Tam, kde detektor experimentu ARES poskytoval nejen informace o pozici, ale i směru zásahu, bude strategie rekonstrukce výrazně odlišná od experimentů, používajících pixelové nebo proužkové detektory (jako v případě CBM), kde máme k dispozici pouze informace o polohách zásahů. Protože neobsahuje informace o směru trajektorie, nenese jeden zásah v hladině detektoru dostatečně velké množství informací, aby mohl popsat trajektorii částice, která ho způsobila. Na druhou stranu, povědomí o parametrech trajektorie můžeme zlepšit, když v rámci jedné buňky automatu zkombinujeme data z vícero po sobě jdoucích zásahů.

Jak je ukázáno na obrázku 3.17, v případě experimentů s vysokou hustotou trajektorií definujeme buňku automatu odlišně od intuitivního přístupu: Zde je každá buňka potenciální segment trajektorie, skládající se ze dvou sousedních zásahů v detektoru. Vstupní informace algoritmu jsou vyčtené zásahy v jednotlivých detektorových hladinách: Zeleně vybarvené značky reprezentují zásahy způsobené jednou částicí, modré značky způsobila částice jiná. Bílá značka značí falešný zásah způsobený šumem. V první fázi algoritmus spojí všechny zásahy na sousedních detektorových hladinách a takto vytvořené segmenty projde a zahodí takové kombinace, které by nikdy nemohly patřit reálné částici. Od této chvíle už algoritmus nikdy nebude pracovat s jednotlivými zásahy, ale vždy s buňkami v podobě dvojic zásahů na sousedních detektorových hladinách.

Jakmile jsou „buňky“ vytvořeny, může začít evoluce celulárního automatu: Metoda hledá pro každou buňku živé sousedy (sousedy v tomto případě definujeme tak, že mají s buňkou společný zásah), kteří mají přibližně stejný směr, přičemž bere v potaz fyzikální model částic (zde nám pomůže výše zmíněný Kalmanův Filtr), protože takoví sousedé mají vyšší pravděpodobnost, že patří ke stejné trajektorii. Zároveň je každé buňce postupně přiřazeno číslo, které reprezentuje počet sousedů vlevo (čímž ohodnocujeme délku tohoto kandidáta na trajektorii, čím delší, tím lepší). Tímto způsobem získáváme stromovou strukturu kandidátů na trajektorie. V tomto okamžiku dochází k soutěži jednotlivých kandidátů: Přežít mohou pouze ti nejdelší kandidáti, u kterých Kalmanův Filtr spočítal nejlepší hodnotu χ^2 a kteří nesdílí žádné společné zásahy s lepšími kandidáty [1].



Obrázek 3.17: Zjednodušená ilustrace celulárního automatu, používaného například v experimentu CBM pro rekonstrukci trajektorií částic. Svislé přerušované čáry představují jednotlivé detektorové hladiny, zelené a modré značky značí zásahy způsobené dvěma různými částicemi, bílá značka je falešný zásah způsobený šumem. Segmenty trajektorií jsou zaznamenány jako černé, plné úsečky. Přejato z [1].

Kapitola 4

Návrh a implementace simulátorů částic v detektoru CBM

Tato kapitola se zabývá podrobným rozborem vlastního algoritmu sledování částic. Podrobně se tato kapitola bude zabývat návrhem, implementací, verifikací a optimalizací simulátoru dráhy částic za účelem získání testovacích dat pro účely návrhu a verifikace výsledného algoritmu. Zvláštní pozornost je pak věnována demonstraci a verifikaci sledovacího algoritmu (a jeho součástí) a rozboru možností jeho akcelerace na alternativních hardwarových platformách, zejména pak na **programovatelných hradlových polích** (dále jen **FPGA**[1]). Závěrem se tato kapitola také zabývá možnými problémy, které efektivní akceleraci algoritmu komplikují, a návrhy řešení těchto problémů.

4.1 Použití Kalmanova Filtru

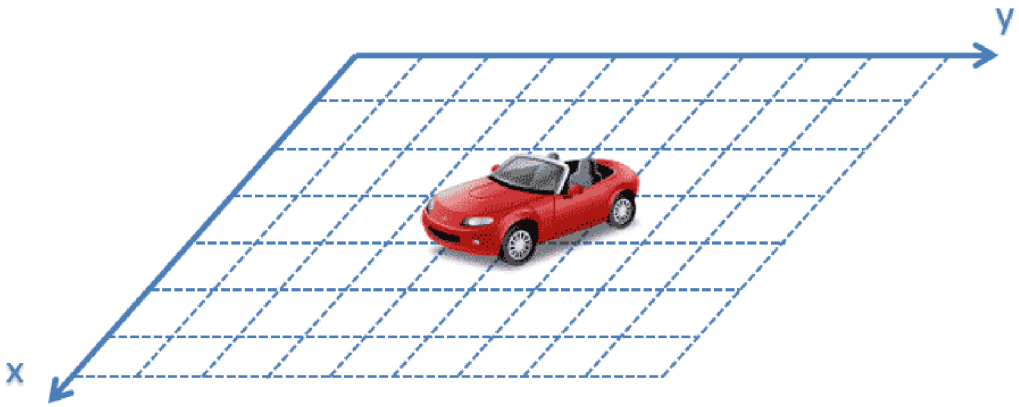
Algoritmus sledování částic, diskutovaný v této práci, využívá do značné míry technologie Kalmanova filtru (respektive jeho rozšířené varianty, **EKF**). Implementace vhodného Kalmanova filtru je komplikovaný proces s mnoha faktory, které mohou drasticky ovlivnit kvalitu jeho výsledku, je proto velmi důležité správně porozumět jeho fungování. Z tohoto důvodu tato práce uvádí za účelem lepšího porozumění algoritmu příklad, který by měl napomoci snazšímu pochopení funkce, kterou Kalmanův filtr vykonává při rekonstrukci trajektorií částic v detektoru.

Pokusíme se implementovat příklad vícerozměrného Kalmanova Filtru v podobě teoretického vozidla, které se pohybuje po rovině. Teoretické vozidlo je vybaveno dvěma senzory, které měří x a u souřadnice vozidla. Mějme stavový vektor $\boldsymbol{r}^t$ definovaný následujícím způsobem:

$$\boldsymbol{r}_k = \begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \\ y_k \\ \dot{y}_k \\ \ddot{y}_k \end{bmatrix} \tag{4.1}$$

Tečkou nad proměnnou značíme derivaci proměnné v čase, dvěma tečkami druhou derivaci atd. K tomu, abychom mohli sestavit pohybový model systému (a na jeho základě pak

[1] FPGA – Field Programmable Gate Array (programovatelné hradlové pole)



Obrázek 4.1: Automobil, používající ke své lokalizaci pouze dva senzory: první poskytuje informaci poloze automobilu ve směru osy x, druhý ve směru osy y. Převzato z webu Alexe Beckera [5]

operátor F, viz rovnice 3.20), musíme nejprve definovat vztahy mezi jednotlivými proměnnými a jejich předešlými stavy. K tomu nám v tomto případě pomohou tyto kinematické rovnice pro konstantní zrychlení:

$$v = v_0 + a\Delta t \tag{4.2}$$

$$\Delta x = v_0 \Delta t + \frac{1}{2} a \Delta t^2 \tag{4.3}$$

Kde v je výsledná rychlost, v_0 je počáteční rychlost, a je konstantní zrychlení, Δx je změna pozice x, Δt je časový interval. Začněme s rovnicí pro zrychlení $a_x = \dot{v}_x = \ddot{x}$: Vzhledem k tomu, že počítáme s konstantním zrychlením, hodnota $\ddot{x}$ se v čase nebude měnit:

$$a_x = \text{const.} \implies \ddot{x}_k = \ddot{x}_{k-1}$$

K určení vztahu mezi $\dot{x}_k$ a $\dot{x}_{k-1}$ využijeme první ze zmíněných kinematických rovnic (4.2), kde jen dosadíme $v = \dot{x}_{k-1}$, $v_0 = \dot{x}_{k-1}$ a $a = \ddot{x}_{k-1}$:

$$\dot{x}_k = \dot{x}_{k-1} + \ddot{x}_{k-1}\Delta t$$

Nakonec určíme vztah mezi x_k a x_{k-1}: víme, že platí vztah:

$$x_k = x_{k-1} + \Delta x$$

kam nám jen stačí dosadit kinematickou rovnici 4.3 tak, že opět dosadíme $a = \ddot{x}_{k-1}$ a $v_0 = \dot{x}_{k-1}$:

$$x_k = x_{k-1} + \dot{x}_{k-1}\Delta t + \frac{1}{2}\ddot{x}_{k-1}\Delta t^2$$

Vztahy pro pohyb po ose y definujeme analogicky a výsledný model můžeme definovat jako následující soustavu rovnic:

$$
\begin{aligned}
x_k &= x_{k-1} + \dot{x}_{k-1}\Delta t + \frac{1}{2}\ddot{x}_{k-1}\Delta t^2 \\
\dot{x}_k &= \dot{x}_{k-1} + \ddot{x}_{k-1}\Delta t \\
\ddot{x}_k &= \ddot{x}_{k-1} \\
y_k &= y_{k-1} + \dot{y}_{k-1}\Delta t + \frac{1}{2}\ddot{y}_{k-1}\Delta t^2 \\
\dot{y}_k &= \dot{y}_{k-1} + \ddot{y}_{k-1}\Delta t \\
\ddot{y}_k &= \ddot{y}_{k-1}
\end{aligned}
\tag{4.4}
$$

Protože je tento model lineární, můžeme tuto soustavu přepsat jako matici a definovat tak operátor $F_k = F_{k-1} = F$:

$$
\boldsymbol{r}_k^- = F\boldsymbol{r}_{k-1}^+ = \begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \\ y_k \\ \dot{y}_k \\ \ddot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \\ \ddot{x}_{k-1} \\ y_{k-1} \\ \dot{y}_{k-1} \\ \ddot{y}_{k-1} \end{bmatrix} \tag{4.5}
$$

Nyní potřebujeme určit podobu kovariančních matic C_k. Pro tento model by obecné kovarianční matice C_k vypadaly následovně:

$$
C_k = \begin{bmatrix} c_x & c_{x\dot{x}} & c_{x\ddot{x}} & c_{xy} & c_{x\dot{y}} & c_{x\ddot{y}} \\ c_{\dot{x}x} & c_{\dot{x}} & c_{\dot{x}\ddot{x}} & c_{\dot{x}y} & c_{\dot{x}\dot{y}} & c_{\dot{x}\ddot{y}} \\ c_{\ddot{x}x} & c_{\ddot{x}\dot{x}} & c_{\ddot{x}} & c_{\ddot{x}y} & c_{\ddot{x}\dot{y}} & c_{\ddot{x}\ddot{y}} \\ c_{yx} & c_{y\dot{x}} & c_{y\ddot{x}} & c_y & c_{y\dot{y}} & c_{y\ddot{y}} \\ c_{\dot{y}x} & c_{\dot{y}\dot{x}} & c_{\dot{y}\ddot{x}} & c_{\dot{y}y} & c_{\dot{y}} & c_{\dot{y}\ddot{y}} \\ c_{\ddot{y}x} & c_{\ddot{y}\dot{x}} & c_{\ddot{y}\ddot{x}} & c_{\ddot{y}y} & c_{\ddot{y}\dot{y}} & c_{\ddot{y}} \end{bmatrix} \tag{4.6}
$$

kde $c_{ab} = \mathrm{cov}(a, b)$ značí kovarianci veličin a a b a $c_a = \mathrm{var}(a)$ značí varianci veličiny a. Předpokládáme, že chyby v odhadu složek stavového vektoru ve směrech x a y jsou vzájemně nezávislé, lze proto předpokládat, že jejich vzájemné kovariance boudou všechny nulové. Přepíšeme tedy kovarianční matici C_k následujícím způsobem:

$$
C_k = \begin{bmatrix} c_x & c_{x\dot{x}} & c_{x\ddot{x}} & 0 & 0 & 0 \\ c_{\dot{x}x} & c_{\dot{x}} & c_{\dot{x}\ddot{x}} & 0 & 0 & 0 \\ c_{\ddot{x}x} & c_{\ddot{x}\dot{x}} & c_{\ddot{x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{yy} & c_{y\dot{y}} & c_{y\ddot{y}} \\ 0 & 0 & 0 & c_{\dot{y}y} & c_{\dot{y}} & c_{\dot{y}\ddot{y}} \\ 0 & 0 & 0 & c_{\ddot{y}y} & c_{\ddot{y}\dot{y}} & c_{\ddot{y}} \end{bmatrix} \tag{4.7}
$$

Po určení kovarianční matice je potřeba určit kovarianční matici procesního šumu: předpokládejme, že model procesního šumu je **diskrétní stochastický proces** (procesní šum je každé měření jiný, ale mezi jednotlivými měřeními zůstává konstantní) a že pohybový model počítá s konstantním zrychlením:

$$Q_k = \begin{bmatrix} \sigma^2_{xx} & \sigma^2_{x\dot{x}} & \sigma^2_{x\ddot{x}} & \sigma^2_{xy} & \sigma^2_{x\dot{y}} & \sigma^2_{x\ddot{y}} \\ \sigma^2_{\dot{x}x} & \sigma^2_{\dot{x}\dot{x}} & \sigma^2_{\dot{x}\ddot{x}} & \sigma^2_{\dot{x}y} & \sigma^2_{\dot{x}\dot{y}} & \sigma^2_{\dot{x}\ddot{y}} \\ \sigma^2_{\ddot{x}x} & \sigma^2_{\ddot{x}\dot{x}} & \sigma^2_{\ddot{x}\ddot{x}} & \sigma^2_{\ddot{x}y} & \sigma^2_{\ddot{x}\dot{y}} & \sigma^2_{\ddot{x}\ddot{y}} \\ \sigma^2_{yx} & \sigma^2_{y\dot{x}} & \sigma^2_{y\ddot{x}} & \sigma^2_{yy} & \sigma^2_{y\dot{y}} & \sigma^2_{y\ddot{y}} \\ \sigma^2_{\dot{y}x} & \sigma^2_{\dot{y}\dot{x}} & \sigma^2_{\dot{y}\ddot{x}} & \sigma^2_{\dot{y}y} & \sigma^2_{\dot{y}\dot{y}} & \sigma^2_{\dot{y}\ddot{y}} \\ \sigma^2_{\ddot{y}x} & \sigma^2_{\ddot{y}\dot{x}} & \sigma^2_{\ddot{y}\ddot{x}} & \sigma^2_{\ddot{y}y} & \sigma^2_{\ddot{y}\dot{y}} & \sigma^2_{\ddot{y}\ddot{y}} \end{bmatrix} \tag{4.8}$$

kde σ^2_{ab} značí kovarianci dvou náhodných proměnných a a b, $\sigma^2_{aa} = \sigma^2_a = \text{var}(a)$ značí varianci náhodné proměnné a. Podobně jako u kovarianční matice C_k předpokládáme, že šum ve směru X nijak nekoreluje se šumem ve směru Y (teoretické vozidlo je vybaveno dvěma na sobě nezávislými senzory, proto předpokládáme, že nepřesnost jejich nijak nekoreluje), můžeme tedy společné členy kovarianční matice opět určit jako nulové:

$$Q_k = \begin{bmatrix} \sigma^2_{xx} & \sigma^2_{x\dot{x}} & \sigma^2_{x\ddot{x}} & 0 & 0 & 0 \\ \sigma^2_{\dot{x}x} & \sigma^2_{\dot{x}\dot{x}} & \sigma^2_{\dot{x}\ddot{x}} & 0 & 0 & 0 \\ \sigma^2_{\ddot{x}x} & \sigma^2_{\ddot{x}\dot{x}} & \sigma^2_{\ddot{x}\ddot{x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{yy} & \sigma^2_{y\dot{y}} & \sigma^2_{y\ddot{y}} \\ 0 & 0 & 0 & \sigma^2_{\dot{y}y} & \sigma^2_{\dot{y}\dot{y}} & \sigma^2_{\dot{y}\ddot{y}} \\ 0 & 0 & 0 & \sigma^2_{\ddot{y}y} & \sigma^2_{\ddot{y}\dot{y}} & \sigma^2_{\ddot{y}\ddot{y}} \end{bmatrix} \tag{4.9}$$

Přestože víme, jak budou jednotlivé kovariance rozloženy, kovarianční matici procesního šumu lze popsat lépe. Využijeme předpokladu, že pohybový model má konstantní zrychlení jak ve směru x, tak ve směru y a definujme kovarianční matici zrychlení Q_a, která tento fakt modeluje:

$$Q_a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \sigma^2_a \tag{4.10}$$

kde σ^2_a je rozptyl zrychlení modelu. Matici procesního šumu je pak možné (v případě konstantního zrychlení v obou osách) zapsat následujícím způsobem:

$$Q_k = F Q_a F^T \tag{4.11}$$

kde F je pohybový model systému (viz rovnice 4.4). Kovarianční matici procesního šumu pak lze definovat následujícím způsobem:

$$Q_k = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma^2_a \tag{4.12}$$

kde σ^2_a je variance zrychlení a Δt je čas mezi jednotlivými měřeními. Podrobné odvození kovarianční matice (pro analogický příklad v 1D) je k dispozici na stránce Alexe Beckera [4].

V dalším kroku je potřeba určit podobu jednotlivých vektorů měření $\boldsymbol{m}_k$ a jejich modelu H_k: V případě tohoto příkladu má vozidlo pouze dva senzory, měřící pozici v osách x a y (vozidlo není vybaveno ani rychloměrem, ani akcelerometrem), proto má vektor měření velikost 2:

$$\boldsymbol{m}_k = \begin{bmatrix} x_{k,m} \\ y_{k,m} \end{bmatrix}$$

kde $x_{k,m}$ a $y_{k,m}$ jsou hodnoty x a y, získané během k-tého měření. Vzhledem k tomu, že stavový vektor má velikost 6, model měření H musí mít rozměr 2×6 a protože měření nevyžaduje žádnou dodatečnou transformaci, bude vztah mezi vektorem měření a jeho modelem vypadat následovně:

$$\boldsymbol{m}_k = H\boldsymbol{r}_k = \begin{bmatrix} x_{k,m} \\ y_{k,m} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \\ y_k \\ \dot{y}_k \\ \ddot{y}_k \end{bmatrix} \tag{4.13}$$

Poslední člen, jehož podobu je potřeba určit, je podoba kovarianční matice měření V_k. Kovarianční matice měření má podobu $V = \langle \boldsymbol{\eta}_k \cdot \boldsymbol{\eta}_k^T \rangle$ (viz rovnice 3.22), kde $\boldsymbol{\eta}_k$ je vektor chyby měření $\boldsymbol{m}_k$ ve tvaru $\boldsymbol{\eta}_k = \left[\begin{smallmatrix} \eta_x \\ \eta_y \end{smallmatrix} \right]$. Kovarianční matice V_k má tedy následující tvar:

$$V_k = \begin{bmatrix} \sigma_{x,m}^2 & \sigma_{yx,m}^2 \\ \sigma_{xy,m}^2 & \sigma_{y,m}^2 \end{bmatrix} \tag{4.14}$$

Opět předpokládáme, že měření na senzorech x a y jsou nezávislé (a tedy jejich kovariance jsou nulové), kovarianční matici lze opět přepsat v následujícím tvaru:

$$V_k = \begin{bmatrix} \sigma_{x,m}^2 & 0 \\ 0 & \sigma_{y,m}^2 \end{bmatrix} \tag{4.15}$$

V reálných situacích očekáváme, že se míra nejistoty v měření může mezi jednotlivými měřeními měnit. Obvykle míra nejistoty u měření závisí na mnoha faktorech, mezi nimi například:

- poměr signálu a šumu (SNR[2]),
- úhel mezi senzory a cílem nebo
- frekvence měření

V zájmu zjednodušení tohoto příkladu předpokládáme, že se míra nejistoty během jednotlivých měření nemění, a tím pádem je kovarianční matice V_k konstantní:

$$V_1 = V_2 \ldots V_{k-1} = V_k = V \tag{4.16}$$

Nyní již zbývá určit jen míru Kalmanova zesílení: z rovnice 3.28 víme, že matice K_k bude mít stejný tvar, jako transponovaná matice modelu měření H^T:

[2]SNR – Signal to Noise Ratio (poměr signálu a šumu)

$$K_k = C_k^- H^T \left(V_k + H C_k^- H^T\right)^{-1} = \begin{bmatrix} K_{1,1_k} & K_{1,2_k} \\ K_{2,1_k} & K_{2,2_k} \\ K_{3,1_k} & K_{3,2_k} \\ K_{4,1_k} & K_{4,2_k} \\ K_{5,1_k} & K_{5,2_k} \\ K_{6,1_k} & K_{6,2_k} \end{bmatrix} \tag{4.17}$$

Pro poslední výpočetní kroky, tj. dopočítání hodnoty χ_k^2 (určení hodnoty χ_k^2 není nutné, ale prospěšné pro určení přesnosti filtru), nového stavu $\boldsymbol{r}_k$ a jeho kovarianční matice C_k již máme všechny potřebné informace, aplikujeme tedy rovnice 3.31 a 3.32, čímž je první iterace Kalmanova Filtru u tohoto příkladu u konce.

Průběh Kalmanova filtru

Data a grafy k následujícímu příkladu byly vygenerovány v rámci skriptu, podrobně popsaného v Jupyter notebooku *KF-demo.ipynb*. V témž notebooku je podrobně rozepsán také chod tohoto Kalmanova filtru.

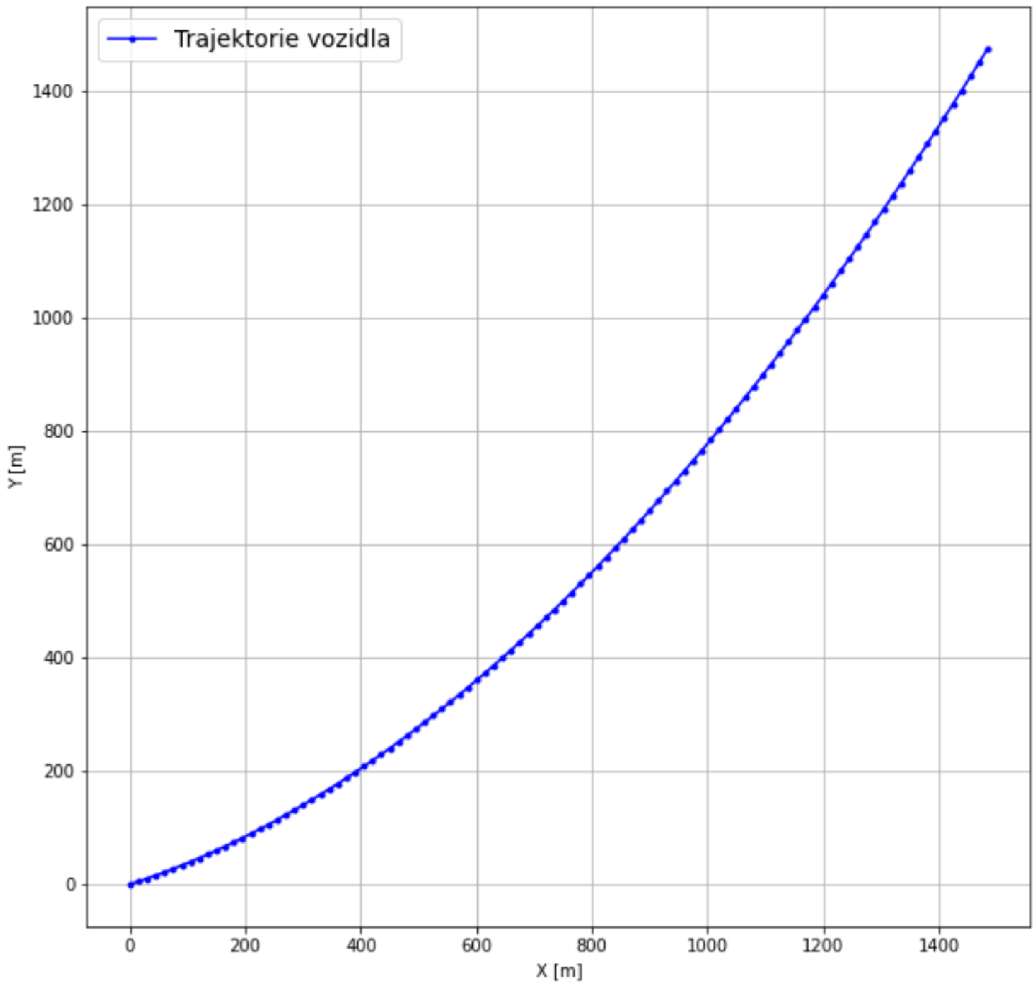
Teoretické vozidlo začíná měření v poloze $[0, 0]$. Ve směru osy x si udržuje konstantní rychlost $v_x = \dot{x} = 15ms^{-1}$ po celou dobu měření, ve směru osy y se pohybuje s konstantním zrychlením $a_y = \ddot{y} = 0.2ms^{-2}$ s počáteční rychlostí $v_y = \dot{y} = 5ms^{-1}$. Vozidlo měří svoji polohu ve směrech os x a y po dobu 100 sekund (viz obr. 4.2). Z takto vytvořené trajektorie byla vytvořena měření zavedením šumu, modelovaného jako normální rozložení se směrodatnou odchylkou $\sigma = 5m$ a středem v $[0, 0]$.

Kalmanův Filtr inicializujeme s následujícími parametry:

- měření probíhá s intervalem $\Delta t = 1s$,
- zrychlení v obou směrech má směrodatnou odchylku $\sigma_a = 0.1ms^{-2}$
- chyba měření má směrodatnou odchylku $\sigma_{x,m} = \sigma_{y,m} = 5m$

Protože interval mezi jednotlivými měřeními je známý, určíme podobu operátoru F, kovarianční matice procesního šumu Q a kovarianční matice měření V:

$$\begin{aligned}
F &= \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0.5 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
Q &= \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 = \begin{bmatrix} 0.25 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 1 & 1 & 0 & 0 & 0 \\ 0.5 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0.5 & 1 & 1 \\ 0 & 0 & 0 & 0.5 & 1 & 1 \end{bmatrix} 0.1^2 \\
V &= \begin{bmatrix} \sigma_{x,m}^2 & 0 \\ 0 & \sigma_{y,m}^2 \end{bmatrix} = \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix}
\end{aligned} \tag{4.18}$$



Obrázek 4.2: Graf trajektorie teoretického vozidla. Vozidlo měří svoji pozici od bodu $[0, 0]$ s konstantní rychlostí $v_x = 15ms^{-1}$ a $v_y = 5ms^{-1}$ a konstantním zrychlením ve směru osy y $a_y = 0.1ms^{-2}$.

k	x [m]	y [m]
1	14.8306	-5.1908
2	9.6017	4.7004
3	30.0431	-0.9286
4	51.9834	19.1439
5	54.9756	21.1406
6	71.3209	30.2757
7	98.8382	47.0236
8	101.3138	39.5408
9	126.2106	46.9865
10	131.5967	52.1862
11 . . . 99	. . .	. . .
100	1477.2950	1478.3722

Tabulka 4.1: Tabulka zašuměných měření polohy vozidla (viz obrázek 4.2).

Inicializace Vzhledem k tomu, že zpočátku Kalmanův filtr nemá žádné informace o pozici, rychlosti ani zrychlení vozidla, nastavíme prvotní odhad stavového vektoru jako nulový:

$$\boldsymbol{r}_0^+ = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.19}$$

Vzhledem k tomu, že prvotní odhad nemá žádný základ v dostupných informacích (jedná se o arbitrární odhad), musíme nastavit také vysokou míru nejistoty, která tento fakt zohlední. Vysoká míra nejistoty v kovarianční matici způsobí vysokou míru Kalmanova zesílení, čímž bude dána výrazně větší váha prvnímu měření.

$$C_0^+ = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix} \tag{4.20}$$

První predikce / propagace V této fázi Kalmanova Filtru bychom za normálních okolností prováděli predikci na základě předchozích informací o systému, vzhledem ale k tomu, že jsme v první iteraci algoritmu a informace o systému nemáme dosud žádné, ve výsledku pouze propagujeme hodnoty z kroku inicializace algoritmu.

$$\boldsymbol{r}_1^- = F\boldsymbol{r}_0^+ = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.21}$$

$$C_1^- = FC_0^+F^T + Q = \begin{bmatrix} 1125.002 & 750.005 & 250.005 & 0 & 0 & 0 \\ 750.005 & 1000.01 & 500.01 & 0 & 0 & 0 \\ 250.005 & 500.01 & 500.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1125.002 & 750.005 & 250.005 \\ 0 & 0 & 0 & 750.005 & 1000.01 & 500.01 \\ 0 & 0 & 0 & 250.005 & 500.01 & 500.01 \end{bmatrix} \tag{4.22}$$

První měření a filtrace Po získání hodnoty prvního měření $\boldsymbol{m}_1 = \left[\begin{smallmatrix} 14.8306 \\ -5.1908 \end{smallmatrix}\right]$ (viz tabulka 4.1) můžeme spočítat první hodnoty váhové matice H_1 a Kalmanova zesílení K_1:

$$\begin{aligned} W_1 &= \left(V + HC_0^- H^T\right)^{-1} = \begin{bmatrix} 0.0008 & 0 \\ 0 & 0.0008 \end{bmatrix} \\ K_1 &= C_1^- H^T W_1 = \begin{bmatrix} 0.9782 & 0 \\ 0.6521 & 0 \\ 0.2173 & 0 \\ 0 & 0.9782 \\ 0 & 0.6521 \\ 0 & 0.2173 \end{bmatrix} \end{aligned} \tag{4.23}$$

Zde je vhodné se pozastavit nad významem Kalmanova zesílení: Kalmanův filtr v průběhu měření v závislosti na přesnosti odhadu filtru a nepřesnosti měření automaticky rozhoduje jak velkou váhu dává které z veličin. Obvykle je Kalmanovo zesílení ze začátku velmi nízké, což odráží fakt, že první odhad stavového vektoru je zvolen libovolně, čemuž odpovídá také podoba první kovarianční matice C_0^+.

Nový odhad stavového vektoru a kovarianční matice Když máme vypočítáno Kalmanovo zesílení, můžeme stanovit nové odhady stavového vektoru $\boldsymbol{r}_1^+$ a jeho kovarianční matice C_1^+. Před tím je ale třeba nejprve určit hodnotu *rezidua*:

$$\boldsymbol{\zeta}_1 = \boldsymbol{m}_1 - H\boldsymbol{r}_1^- = \begin{bmatrix} 14.8306 \\ -5.1908 \end{bmatrix} \tag{4.24}$$

Reziduum $\boldsymbol{\zeta}_1$ se rovná x a y složce prvního měření díky faktu, že první odhad stavového vektoru je nulový. S určeným reziduem lze nyní vypočítat hodnotu opraveného (*a posteriori*) odhadu stavu systému $\boldsymbol{r}_1^+$:

$$\boldsymbol{r}_1^+ = \boldsymbol{r}_1^- + K_1\boldsymbol{\zeta}_1 = \begin{bmatrix} 14.5082 \\ 9.6722 \\ 3.2241 \\ -5.0780 \\ -3.3853 \\ -1.1284 \end{bmatrix} \tag{4.25}$$

$$C_1^+ = (\boldsymbol{I} - K_k H_k)\, C_1^- = \begin{bmatrix} 24.456 & 16.304 & 5.4348 & 0 & 0 & 0 \\ 16.304 & 510.87 & 336.96 & 0 & 0 & 0 \\ 5.4348 & 336.96 & 445.66 & 0 & 0 & 0 \\ 0 & 0 & 0 & 24.456 & 16.304 & 5.4348 \\ 0 & 0 & 0 & 16.304 & 510.87 & 336.96 \\ 0 & 0 & 0 & 5.4348 & 336.96 & 445.66 \end{bmatrix} \tag{4.26}$$

Postupně se bude odhad stavového vektoru zpřesňovat, se zpřesňujícím se odhadem stavového vektoru se budou měnit i váhy mezi měřeními a předchozími odhady (bude se postupně snižovat váha měření a zvyšovat váha předchozího odhadu) a bude se postupně snižovat míra nejistoty, vyjádřená kovarianční maticí C. Podívejme se dále až na poslední iteraci algoritmu:

Hodnota posledního měření $\boldsymbol{m}_{100}$ (viz tabulka 4.1):

$$\boldsymbol{m}_{100} = \begin{bmatrix} 1477.295 \\ 1478.3722 \end{bmatrix} \tag{4.27}$$

Dále přepočítáme Kalmanovo zesílení tak, abychom zjistili, jakou váhu bude v posledním odhadu mít předchozí odhad a jakou váhu hodnota měření:

$$K_{100} = C_{100}^- H^T W_k = \begin{bmatrix} 0.4189 & 0 \\ 0.113 & 0 \\ 0.0152 & 0 \\ 0 & 0.4189 \\ 0 & 0.113 \\ 0 & 0.0152 \end{bmatrix} \tag{4.28}$$

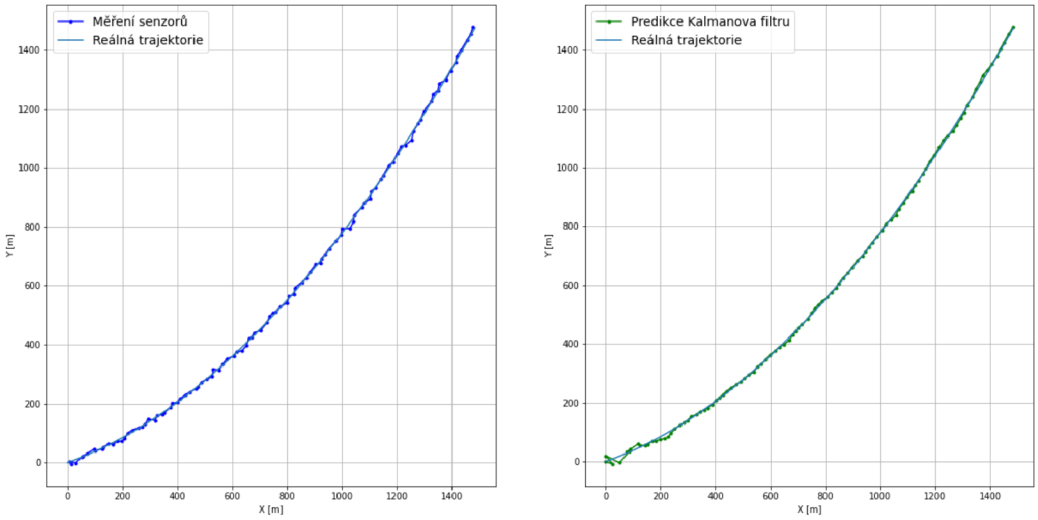
Od první iterace se Kalmanovo zesílení u pozice automobilu změní z původní hodnoty 0.9782, kdy prvotní odhad $\boldsymbol{r}_0^+$ měl pouze zanedbatelnou váhu 0.0218, v poslední iteraci už je hodnota Kalmanova zesílení pro pozici 0.4189, z čehož vyplývá, že předchozí odhad a nové měření mají v tento okamžik téměř stejnou váhu.

Nyní zbývá poslední odhad stavového vektoru $\boldsymbol{r}_{100}$:

$$\boldsymbol{r}_{100}^+ + K_{100}\boldsymbol{\zeta}_{100} = \begin{bmatrix} 1481.0866 \\ 14.5917 \\ -0.0262 \\ 1478.7709 \\ 25.1824 \\ 0.246 \end{bmatrix} \tag{4.29}$$

$$C_{100}^+ = (\boldsymbol{I} - K_{100} H)\, C_{100}^- = \begin{bmatrix} 10.4731 & 2.8255 & 0.3811 & 0 & 0 & 0 \\ 2.8255 & 1.2622 & 0.2377 & 0 & 0 & 0 \\ 0.3811 & 0.2377 & 0.0641 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10.4731 & 2.8255 & 0.3811 \\ 0 & 0 & 0 & 2.8255 & 1.2622 & 0.2377 \\ 0 & 0 & 0 & 0.3811 & 0.2377 & 0.0641 \end{bmatrix} \tag{4.30}$$

Z poslední kovarianční matice vyplývá, že poloha vozidla na osách x a y má nyní rozptyl 10.47 metrů, což znamená, že směrodatná odchylka odhadu polohy metodou Kalmanova



Obrázek 4.3: Grafy měření (vlevo) a odhadů stavového prostoru pomocí Kalmanova filtru (vpravo).

filtru je po 100 sekundách $\sqrt{c_{x,100}} = \sqrt{c_{y,100}} = \sqrt{10.47} = 3.2362$. Metoda je tedy výrazně přesnější než při použití pouze dat obou senzorů, které mají směrodatnou odchylku 5 m, navíc můžeme předpokládat, že při delším měření se budou odhady dále zpřesňovat. Rychlost konvergence a přesnost filtru můžeme dále zlepšovat, jedna z možností zpřesnění filtru spočívá například ve vhodném odhadu prvotní predikce stavu r_0^+ (a s tím související kovarianční matice).

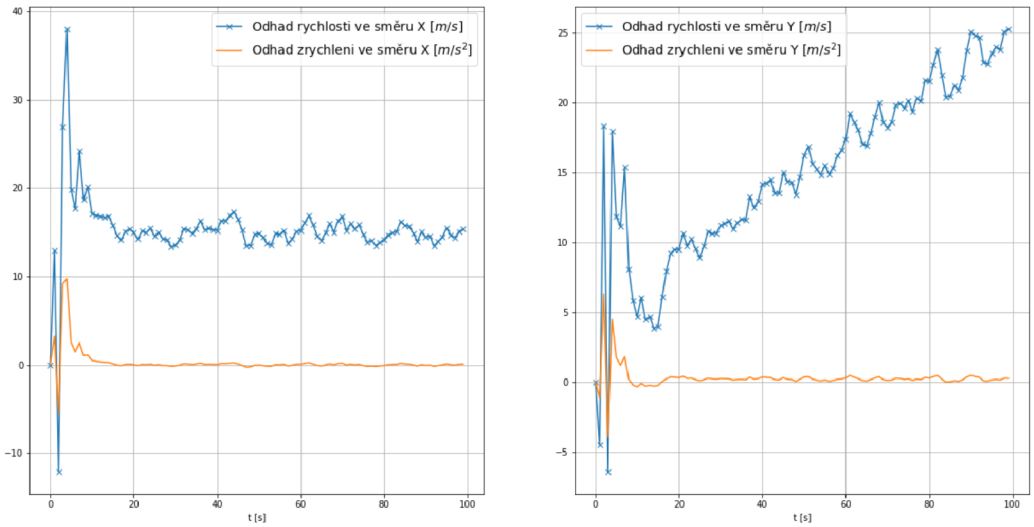
Jak je vidět na obrázku 4.3, Kalmanův Filtr je velmi efektivní metodou pro sledování dynamických systémů. Všimněme si, že v okamžiku, kdy auto začne zatáčet, jednotlivé složky zrychlení $\ddot{x}$ a $\ddot{y}$ již nejsou konstantní, míra (uhlového) zrychlení ale konstantní zůstává. Z tohoto důvodu (dobře jsme si stanovili hodnotu σ_a^2) zůstává Kalmanův Filtr spolehlivý i přesto, že jednotlivé složky konstantní nezůstávají. Všimněme si snížení přesnosti odhadů v okamžiku, kdy automobil začne zatáčet, která se ale o několik iterací filtru později opět zlepší.

4.2 Simulátor částic v detektoru

V předchozím příkladu bylo poměrně jednoduché vygenerovat experimentální data z toho důvodu, že kinematické rovnice pro pohyb s konstantním zrychlením je poměrně snadné vyřešit analyticky. Bohužel, v případě trajektorií částic je fyzikální model výrazně složitější a pro účely této práce je výhodnější generovat teoretické částice numericky za pomoci spojité simulace. Navíc vzhledem k tomu, že experiment CBM v době publikace této práce nebyl uveden do provozu, nejsou k dispozici žádná reálná experimentální data.

Pro účely generování testovacích dat se lze dopustit některých zobecnění, která výrazně usnadní jak vlastní generování částic, tak vlastní práci s daty. Konkrétně zjednodušíme fyzikální model následovně:

- **homogenní magnetické pole**: Simulátor si klade za cíl modelovat pouze trajektorii v rámci prvního stanoviště detektoru (STS), který se celý nachází uprostřed velkého



Obrázek 4.4: Graf vývoje odhadu rychlosti a zrychlení v čase podle osy X (vlevo) a Y (vpravo).

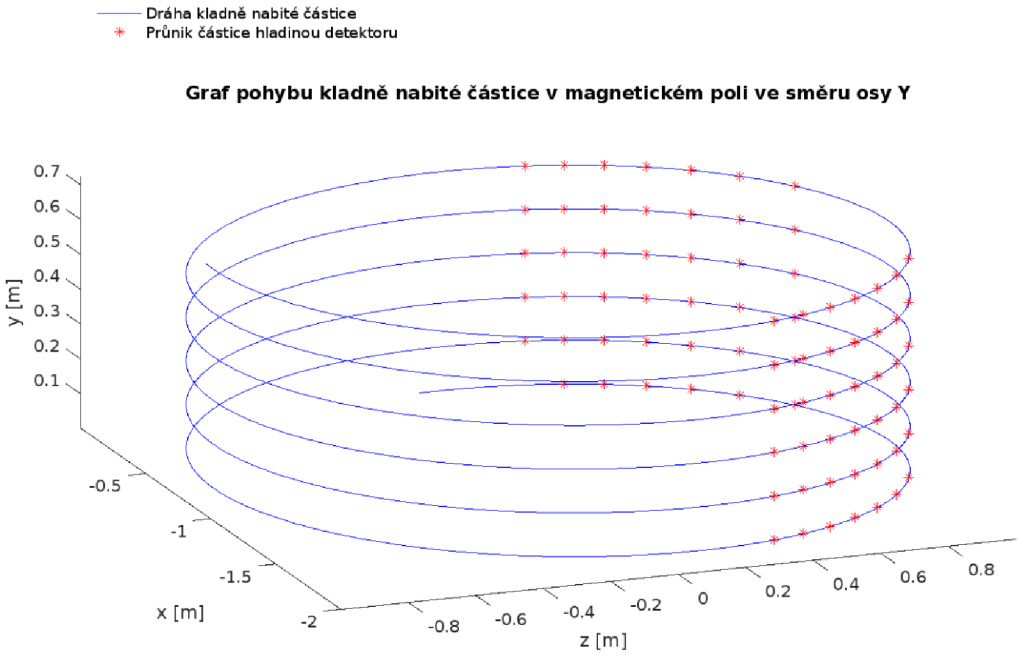
supravodivého magnetu. který vytváří silné, a díky vlastnostem magnetu magnetické pole, které lze uvnitř objemu STS zobecnit jako homogenní.

- **nedochází k interakcím mezi částicemi**: Ve chvíli, kdy částice vstoupí do detektoru, interaguje pouze s detektorem, nikoli mezi sebou. Díky tomuto předpokladu si můžeme dovolit simulovat každou částici zvlášť, a v případě potřeby můžeme generování částic paralelizovat, aniž by bylo nutné jednotlivá vlákna synchronizovat.

- **všechny částice jsou primární s vysokou energií**: Předpokládáme, že všechny částice v detektoru vznikly v místě terče, tedy během prvotní srážky. Nepředpokládáme tedy, že některé částice vznikly během průletu detektorem, protože vytvořit takový model by vyžadovalo výrazně hlubší porozumění problematice kvantové chromodynamiky a kvantové fyziky obecně, nehledě na fakt, že se jedná o *cutting edge* odvětví, které je momentálně intenzivně zkoumáno a výzkum takovýchto modelů je jedna z motivací experimentu CBM.

- **všechny částice mají dlouhou střední délku života**: Předpokládáme, že všechny částice, které v rámci detektoru simulujeme, mají dostatečně dlouhou střední délku života, aby stihly ze sledovaného prostoru detektoru vyletět. Toto zjednodušení zavádíme kvůli tomu, aby bylo chování částic konzistentní s ostatními zjednodušeními modelu: Pokud bychom připustili, že některé částice mají kratší délku života, než je potřebné na opuštění prostoru stanoviště STS, a museli bychom v tom případě řešit problém, kdy uvnitř detektoru dochází k sekundárním rozpadům částic, které generují částice nové, sekundární. Tohle chování by bylo ale v rozporu s předchozím předpokladem

- **neuvažujeme mnohonásobný rozpad**: Pro první verzi simulátoru je zbytečné brát v úvahu mnohonásobný rozpad částic vzhledem k tomu, že bychom museli na místě každé detektorové hladiny přepočítávat nejen trajektorii částice, ale i její energii, což je netriviální úkon, který sice zpřesní simulaci částic, toto zpřesnění ale pro nás během implementace sledovacího algoritmu nemá velký přínos.



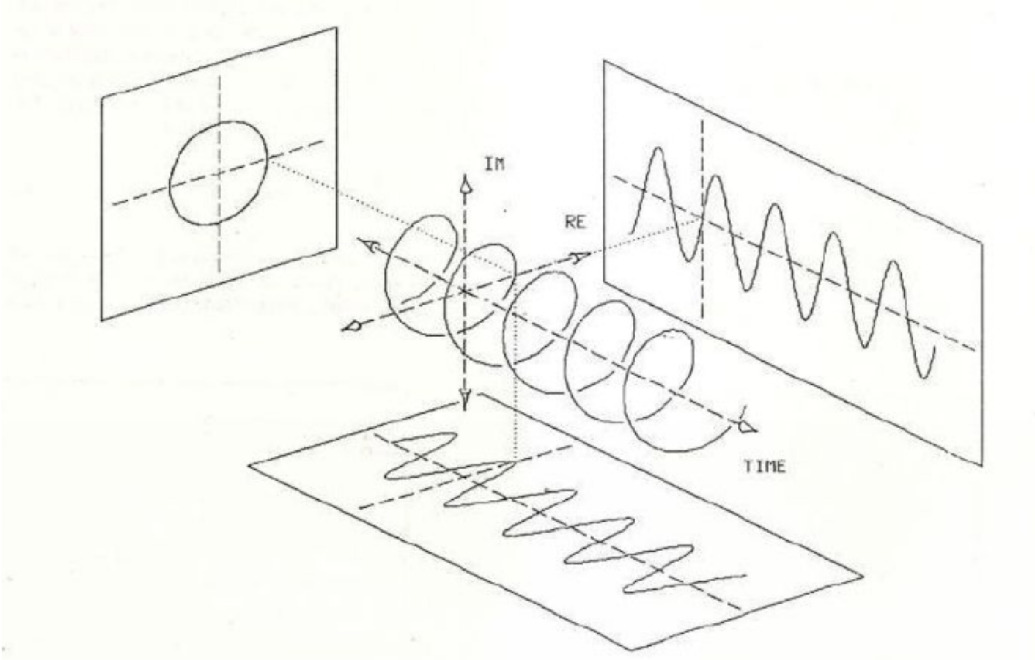
Obrázek 4.5: Trajektorie pomalé, kladně nabité teoretické částice v detektoru s homogenním magnetickým polem ve směru osy Y. Červeně jsou vyznačeny průniky částice s hladinami detektoru, modře pak vlastní trajektorie částice. Trajektorie byla vytvořena pomocí simulačního modelu a výsledný graf byl vytvořen v programu MATLAB.

- **nulové elektrické pole**: Protože se částice pohybují relativistickými rychlostmi, je výpočet jejího chování výrazně složitější. Přítomnost nenulového elektrického pole by způsobilo neúměrné ztížení práce se simulátorem, přičemž jeho zohlednění přinese pouze minimální zpřesnění modelu (magnetické pole je v tomto případě výrazně dominantní silou). Díky tomuto předpokladu získáme velkou výhodu z toho důvodu, že na rozdíl od magnetického pole elektrické pole ovlivňuje množství energie částice, což v důsledku zákonů speciální relativity způsobuje, že se mj. hmotnost částice stává proměnnou. V opačném případě je možné vlivy speciální relativity vypočítat již během inicializace simulace a během simulace *de facto* pracujeme s klasickou mechanikou.

Všechna tato zobecnění nám umožní vytvořit vhodný simulátor, který bude generovat částice vhodné pro vývoj a validaci sledovacího algoritmu. Takový přístup má nevýhodu v tom, že takto vytvořená testovací data nejsou vhodná pro hodnocení výkonu výsledného algoritmu, vývoj takového simulátoru by ale byl nad rámec této práce, zjednodušený simulátor je proto pro tuto práci vyhovující.

4.3 Metody řešení diferenciálních rovnic

Jedna z důležitých metrik při hodnocení kvality simulátoru částic pro tuto práci rychlost, s jakou dokáže částice generovat. Z tohoto důvodu je důležité zhodnotit každý aspekt simulátoru a optimalizovat tak, aby výsledný program generoval cvičné částice co nejrychleji, zároveň si ale zachoval vysokou přesnost. Proto je nutné správně zvolit metodu řešení diferenciálních rovnic, na které lze potom celý simulační model postavit. Pro účely této práce



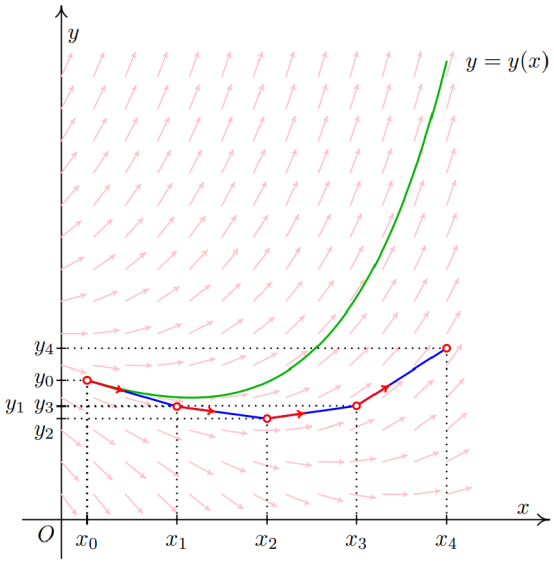
Obrázek 4.6: Komplexní exponenciála a její projekce do reálné roviny (kosinusovka), imaginární roviny (sinusovka) a komplexní roviny (kružnice). Převzato z [2].

bylo zvoleno numerické řešení, za zmínku ale stojí i možnost **analytických řešení** trajektorie, přestože pro naše účely nejsou zcela vhodné (simulační model je použitý i v rámci vlastního sledovacího algoritmu, kde již s homogenním magnetickým polem nelze počítat, numerické řešení je zde vhodnější): V případě, že se nabitá částice pohybuje ve směru kolmém na směr magnetického pole, její trajektorie pak nabývá tvaru **kružnice**. V případě, že se směr částice kolmý není (tj. její má nenulovou složku $\dot{y}$, popřípadě θ_y), trajektorie nabývá tvaru **šroubovice**, kterou můžeme vyjádřit jako **komplexní exponenciálu** s osou ve směru magnetického pole (v našem případě ve směru osy y):

$$f(y) = R \times e^{i2\pi ky} + w \tag{4.31}$$

kde R je poloměr otáčení šroubovice, e je Eulerovo číslo, i je imaginární jednotka, k je perioda exponenciály a $w \in \mathbb{I}$ je posunutí osy otáčení exponenciály v komplexní rovině, která je kolmá na směr magnetického pole (v našem případě rovina xz). R můžeme také definovat jako poloměr oblouku trajektorie, jeho znaménko pak určuje směr zakřivení, který je určen nábojem částice. Vzhledem k tomu, že simulační model předpokládá, že všechny částice mají počátek v terči, posunutí w a poloměr R se stávají lineárně závislými, a tak můžeme nabitou částici v homogenním magnetickém poli vyjádřit pomocí jednoho reálného parametru k a jednoho komplexního parametru w, kde parametr k udává úhel θ_y a parametr w osu otáčení komplexní exponenciály. Toto řešení má ale dvě okrajové situace, kvůli kterým je pro účely tohoto simulátoru nevhodné:

- **částice s nulovým nábojem**: v případě, že takto modelovaná částice má nulový náboj, vzniká problém s parametrem w, který v tomto případě nemá řešení (u přímé trajektorie nelze určit poloměr, proto nelze určit ani bod na komplexní rovině, kterým by procházela osa komplexní exponenciály).



Obrázek 4.7: Graf fungování jednoduché Eulerovy metody. Zeleně je vyznačena reálná funkce, modře jednotlivé kroky Eulerovy metody. Obrázek byl převzat ze skript pro předmět Numerická matematika a pravděpodobnost na Fakultě Informačních Technologií VUT [13].

- **částice s nulovým úhlem** θ_y: částice, jejíž trajektorie je rovnoběžná s rovinou xz taktéž nemá v tomto vyjádření řešení, protože taková komplexní exponenciála by měla teoreticky nulovou periodu, narážíme tedy na analogický problém, jako při Houghově transformaci svislé přímky (viz obrázek 3.10).

Z tohoto důvodu se tato práce nebude možnostmi analytického řešení trajektorií v detektoru dále zabývat.

Numerické metody

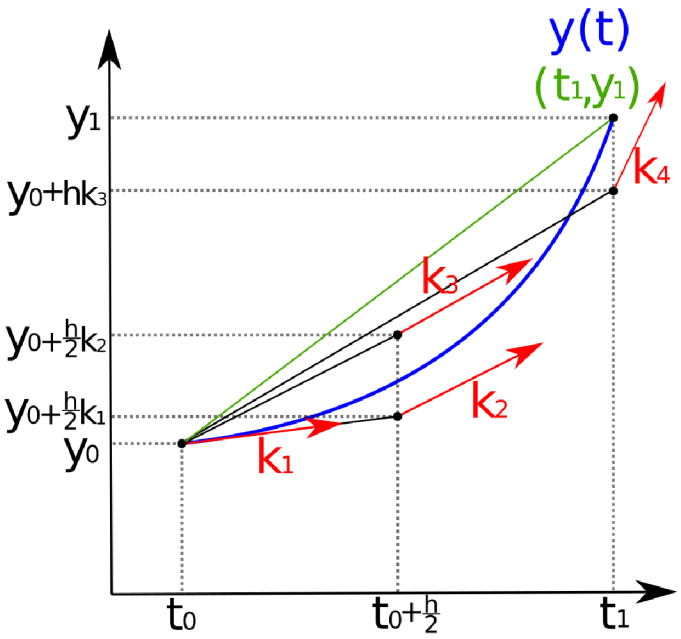
Po vyloučení použití analytických metod je třeba přistoupit k použití numerických metod. V úvahu jsou brány zejména dvě jednokrokové metody: **Eulerova metoda** a **metoda Runge-Kutta** čtvrtého řádu (dále RK4):

Eulerova metoda je nejjednodušší metodou numerického řešení obyčejných diferenciálních rovnic s danými počátečními podmínkami. Mějme obyčejnou diferenciální rovnici prvního řádu:

$$y'(t) = f\left(t, y(t)\right), y(t_0) = y_0 \tag{4.32}$$

jejíž hodnotu chceme numericky spočítat. Zvolíme délku kroku numerické metody h (pro každé t_n platí $t_n = t_0 + nh$) a pomocí rekurentního vzorce Eulerovy metody začneme s numerickou integrací:

$$\begin{aligned} y_{n+1} &= y_n + hf\left(t_n, y_n\right) \\ t_{n+1} &= t_n + h \end{aligned} \tag{4.33}$$



Obrázek 4.8: Graf fungování Runge-Kuttovy metody čtvrtého řádu. Modře je vyobrazená reálná funkce, zeleně je vyobrazena výsledná hodnota y_{n+1}, červeně jsou zobrazeny jednotlivé koeficienty. Obrázek byl převzat z dokumentace metody RK4 na stránkách *Read the Docs* [24].

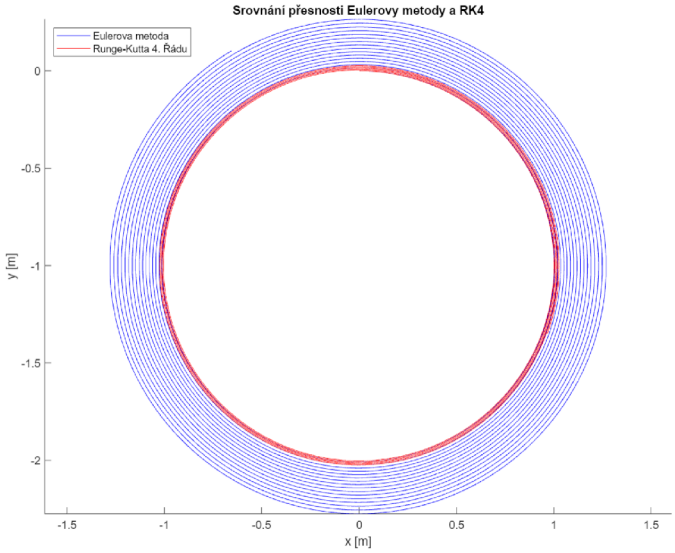
přičemž platí, že počáteční hodnota y_0 v čase t_0 je zadána. Myšlenka Eulerovy metody je, že i když neznáme podobu křivky, kterou zkoumáme, známe počáteční bod, ze kterého vycházíme a z diferenciální rovnice dokážeme vypočítat sklon funkce v tomto počátečním bodě, čímž získáme tečnu, po které se posuneme o krátký časový úsek (krok) h.

Alternativou Eulerovy metody jsou Runge-Kuttovy metody. Samotná Eulerova metoda odpovídá Runge-Kuttově metodě prvního řádu, pro použití v simulátoru částic byla ale použita nejznámější z Runge-Kuttových metod, konkrétně metoda Runge-Kutty čtvrtého řádu:

$$\begin{aligned} t_{n+1} &= t_n + h, \\ y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2_3 + k_4\right) h, \\ k_1 &= f\left(t_n, y_n\right), \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right), \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right), \\ k_4 &= f\left(t_n + h, y_n + hk_3\right), \end{aligned} \tag{4.34}$$

Kde $k_1, \ldots, k_4$ jsou koeficienty, definující:

- k_1 je sklon funkce v bodě y_n (Eulerova metoda),



Obrázek 4.9: Srovnání propagace chyby numerických metod Runge-Kutta 4. řádu (červeně) a Eulerovy metody (modře). Data byla vygenerována simulátorem částic a zpracována do grafu programem MATLAB.

- k_2 je sklon funkce ve středu intervalu, v bodě získaným pomocí koeficientu k_1 (metoda **Midpoint**),
- k_3 je opět sklon funkce ve středu intervalu, tentokrát bod získává na základě koeficientu k_2,
- k_4 je sklon na konci intervalu, k jeho výpočtu je potřeba koeficient k_3

Stabilita numerických metod

Vzhledem k povaze simulačního modelu, kdy se nabitá částice ve směru kolmém na magnetické pole (za předpokladu, že se nemění kinetická energie částice působením elektrického pole) pohybuje po kružnici (respektive po šroubovici za předpokladu, že složka hybnosti částice ve směru magnetického pole je nenulová), je třeba se pozastavit nad problematikou stability vybraných numerických metod. Bohužel v případě nabitých částic bude jak Eulerova metoda, tak metoda RK4 vždy nestabilní (chování Eulerovy metody a RK4 viz obrázek 4.9). Metoda n-tého řádu je schopna přesně určit řešení diferenciální rovnice za předpokladu, že přesné řešení diferenciální rovnice je polynom řádu maximálně n. Eulerova metoda tedy přesně určí řešení, pokud je přesné řešení přímka, u RK4 polynom maximálně čtvrtého řádu atd. Vzhledem k tomu, že pohyb po kružnici (respektive po šroubovici) se jako polynom zapsat nedá, lze přesně spočítat pouze tu složku pohybu, která je rovnoběžná se směrem magnetického pole.

V důsledku tohoto faktu je důležité, aby byla vhodně zvolena délka kroku metody, která sice nezaručí stabilitu, ale poskytne vhodný kompromis mezi rychlostí výpočtu a přesností metody. V tomto ohledu je metoda Runge-Kutta výrazně spolehlivější než Eulerova metoda, přitom poskytuje dobrý kompromis mezi přesností a výpočetní náročností: RK4 je nejvyšší řád Runge-Kuttových metod, kde řád metody odpovídá počtu jejích kroků (RK1/Eulerova

metoda: 1 krok, ... RK4: 4 kroky, ale již RK5: 6 kroků), metody vyšších řádů pak rychle nabývají na výpočetní náročnosti a pro nás jsou tak méně vhodné.

Použití numerických metod pro rovnice vyšších řádů

Přestože jsou Eulerova metoda i metoda Runge-Kutta určeny pro řešení diferenciálních rovnic prvního řádu, můžeme ji použít i pro obyčejné diferenciální rovnice vyšších řádů tak, že převedeme diferenciální rovnici vyššího řádu na soustavu rovnic prvního řádu. Mějme diferenciální rovnici N-tého řádu:

$$y^{(N)}(t) = f\left(t, y(t), y'(t), \ldots, y^{(N-1)}(t)\right) \tag{4.35}$$

zavedeme pomocné proměnné z následujícím způsobem:

$$\begin{aligned} z_1(t) &= y(t) \\ z_2(t) &= y'(t) \\ &\vdots \\ z_N(t) &= y^{(N-1)}(t) \end{aligned} \tag{4.36}$$

Obyčejnou diferenciální rovnici vyššího řádu pak můžeme přepsat jako následující soustavu diferenciálních rovnic prvního řádu:

$$\boldsymbol{z}'(t) = \begin{bmatrix} z_1'(t) \\ z_2'(t) \\ \vdots \\ z_{N-1}'(t) \\ z_N'(t) \end{bmatrix} = \begin{bmatrix} z_2(t) \\ z_3(t) \\ \vdots \\ z_N(t) \\ f\left(t, z_1(t), \ldots, z_N(t)\right) \end{bmatrix} = \begin{bmatrix} y'(t) \\ y''(t) \\ \vdots \\ y^{(N-1)}(t) \\ y^{(N)}(t) \end{bmatrix} \tag{4.37}$$

4.4 Pohybové rovnice simulačního modelu, první simulátor

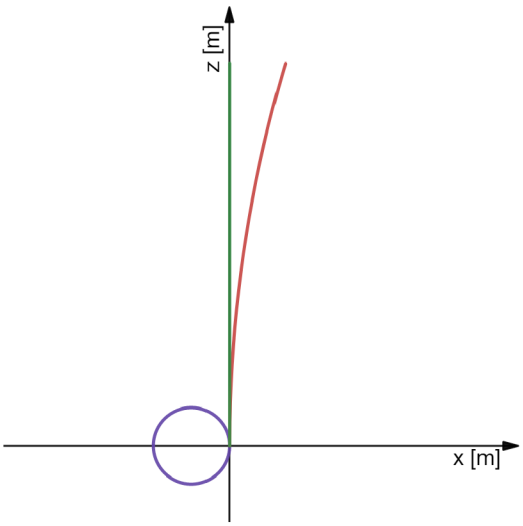
První verze simulátoru dráhy částic byl navržen tak, aby co nejpřesněji modeloval dráhu teoretických, „pomalých“ částic. Pro popis dráhy pomalé částice v elektromagnetickém poli slouží jev zvaný **Lorentzova síla**, která popisuje sílu působící na nabitou částici v elektromagnetickém poli.

$$\boldsymbol{F} = q\left(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}\right) \tag{4.38}$$

kde $\boldsymbol{E}$ je vektor elektrického pole, $\boldsymbol{v}$ je vektor rychlosti částice v elektromagnetickém poli a $\boldsymbol{B}$ je vektor magnetického pole. Jak bylo zmíněno výše, v rámci simulátoru je elektrické pole $\boldsymbol{E}$ zanedbáno, lze tedy rovnici přepsat pro účely simulátoru následujícím způsobem:

$$\boldsymbol{F} = q\boldsymbol{v} \times \boldsymbol{B} \tag{4.39}$$

Všimněme si, že v tuto chvíli v rovnici nefiguruje hmotnost částice, protože velikost Lorentzovy síly je na ní nezávislá. Hmotnost částice ale potřebujeme znát pro výpočet zrychlení částice, aplikujeme tedy druhý Newtonův zákon $F = ma$, díky čemuž dostáváme rovnici:



Obrázek 4.10: Demonstrace simulace pohybu různých, nerelativistických částic. Červeně je vyznačena dráha těžké / slabě nabité částice s kladným nábojem, modře je vyznačena lehká / silně nabitá částice se záporným nábojem a zeleně je vyznačená dráha částice s nulovým nábojem. Směr magnetického pole je rovnoběžný s osou y, všechny částice mají shodnou počáteční rychlost $\boldsymbol{v} = (0, 0, 1)\, ms^{-1}$. Graf byl vytvořen za pomoci nástroje Desmos [9].

$$\dot{\boldsymbol{v}} = \frac{q}{m}\boldsymbol{v} \times \boldsymbol{B} \tag{4.40}$$

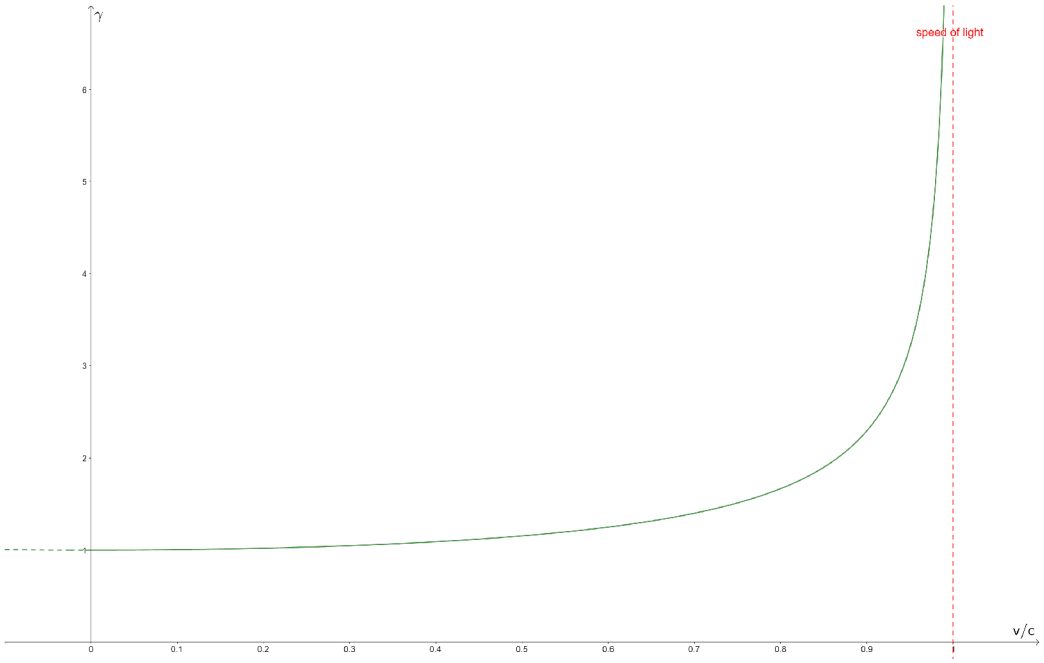
Kde $\dot{\boldsymbol{v}} = \boldsymbol{a}$ je vektor zrychlení částice a m je její hmotnost. V této rovnici lze pro potřeby simulátoru rozepsat vektory následujícím způsobem:

$$\begin{aligned} \ddot{x} &= \frac{q}{m}\left(\dot{y}B_z - \dot{z}B_y\right) \\ \ddot{y} &= \frac{q}{m}\left(\dot{z}B_x - \dot{x}B_z\right) \\ \ddot{z} &= \frac{q}{m}\left(\dot{x}B_y - \dot{y}B_x\right) \end{aligned} \tag{4.41}$$

kde $\dot{x}$ je rychlost částice ve směru x, $\ddot{x} = \dot{v}_x = a_x$ značí zrychlení ve směru dané osy, B_x je velikost intenzita magnetického pole ve směru x (analogicky pro směry y a z). Tímto způsobem jsme získali soustavu tří obyčejných diferenciálních rovnic, které nám poslouží jako první simulační model. Za pozornost stojí ještě fakt, že v případě homogenního magnetického pole, které má směr rovnoběžný s některou z os je možné model dále zjednodušit: Uvažujme případ detektoru, ve kterém má magnetické pole směr rovnoběžný s osou y, model lze pak zjednodušit následujícím způsobem:

$$\begin{aligned} \ddot{x} &= -\frac{q}{m}\dot{z}B \\ \ddot{z} &= \frac{q}{m}\dot{x}B \end{aligned} \tag{4.42}$$

Kde $B = B_y$ je magnetické pole. Všimněme si, že rovnici pro výpočet $\ddot{y}$ můžeme zcela zanedbat, zrychlení v tomto směru se vždy vyhodnotí jako nulové, v tomto směru se tedy



Obrázek 4.11: Graf funkce rychlosti a Lorentzova faktoru. Při rychlostech blížících se rychlosti světla Lorentzův faktor stoupá k nekonečnu. Graf byl vytvořen pomocí nástroje Desmos [9].

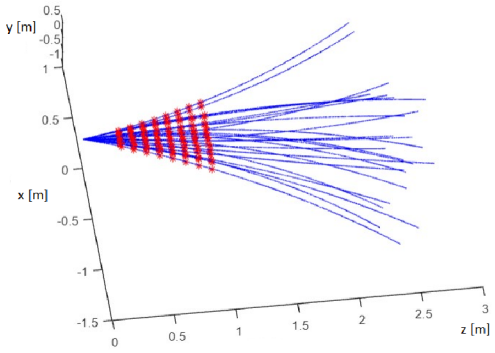
částice budou pohybovat rovnoměrným přímočarým pohybem. Zjednodušení simulátoru tímto způsobem je sice možné, pro účely simulátoru je ale výhodné mít možnost velikost a orientaci magnetického pole nastavovat, bylo tedy učiněno rozhodnutí implementovat obecnější variantu z rovnice 4.41.

4.5 Speciální relativita v pohybových rovnicích, druhý simulátor

Simulační model popsaný výše má zásadní nedostatek, který je patrný ve chvíli, kdy pomocí tohoto simulátoru vygenerujeme první částice s úkolem „modelovat“ reálné částice (například protony či tauony): Protože používáme pro výpočet zrychlení klasické zákony mechaniky, nastává problém ve chvíli, kdy simulátor začne generovat částice při relativistických rychlostech. Protože model nebere zákony speciální relativity v potaz, simulátor může generovat částice, které v praxi neexistují a vzhledem k tomu, že u částic v detektoru CBM očekáváme rychlosti přes $0.9c$, spoléhat se na klasickou mechaniku je nevhodné. Další z nedostatků je, že simulátoru teoreticky nic nebrání generovat částice s rychlostmi přesahujícími rychlost světla.

Abychom mohli simulační model upravit tak, aby zohledňoval relativistické chování částic, musíme nejprve zavést takzvaný **Lorentzův faktor**:

$$\gamma \equiv \frac{\mathrm{d}t}{\mathrm{d}\tau} = \frac{c}{\sqrt{c^2 - v^2}} = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{4.43}$$



Obrázek 4.12: Graf 25 náhodně vygenerovaných částic, vlastnostmi odpovídajících částicím $p^+, \Lambda_c^+, \Sigma^-$ a Σ_b^-. Modře jsou naznačeny vygenerované trajektorie, červeně jsou vyznačeny zásahy částic v hladinách detektoru. Data byla vygenerována pomocí simulátoru částic a graf byl vytvořen pomocí programu MATLAB.

Kde c značí rychlost světla ve vakuu, τ je takzvaný **vlastní čas**. Pohybové rovnice předchozího modelu (rovnice 4.41), po korekci pro speciální relativitu mají pak následující tvar:

$$\begin{aligned} \ddot{x} &= \frac{q}{\gamma m}\left(\dot{y}B_z - \dot{z}B_y\right) \\ \ddot{y} &= \frac{q}{\gamma m}\left(\dot{z}B_x - \dot{x}B_z\right) \\ \ddot{z} &= \frac{q}{\gamma m}\left(\dot{x}B_y - \dot{y}B_x\right) \end{aligned} \tag{4.44}$$

Zavedení Lorentzova faktoru do pohybových rovnic znamená, že dosud konstantní hmotnost částice m se stává proměnnou, závislou na rychlosti částice. Rozhodnutí předpokládat v simulátoru homogenní magnetické pole a nulovou intenzitu elektrického pole se zde projeví jako výhodné z důvodu, že bez vlivu elektrického pole velikost rychlosti částice v zůstává konstantní, proto zůstává konstantní i její celková hmotnost, kterou pouze při inicializaci simulace upravíme na základě klidové hmotnosti a Lorentzova faktoru. V případě druhého, pro speciální relativitu upraveného simulátoru již dokážeme efektivně modelovat částice s relativistickou rychlostí.

4.6 Kombinovaný simulátor částic s logováním zásahů v detektoru

Generování cvičných částic bylo implementováno pro demonstrační účely ve formátu Jupyter notebook, původně v jazyce Python 3.9.4 (za účelem rychlého prototypování). V průběhu práce byl simulátor přepsán v jazyce Julia 1.6.1. díky podobnostem s jazykem Python a výrazně vyšší rychlostí zpracování [6]. Simulátor byl nakonec přepsán i do jazyka C, zčásti kvůli dalšímu zvýšení rychlosti zpracování, zčásti jako příprava pro implementaci vlastního algoritmu sledování částic, který v rámci svého fungování používá simulace tras částic pro odhad cesty částice mezi jednotlivými hladinami detektoru.

Původně byl simulátor navržen pro spojitou simulaci, která probíhá ve vlastním čase každé částice od času $t_0 = 0s$ do času $t_{max} = \frac{1.25}{v}$ sekund, kde 1.25 je délka části dráhy částice, která je ovlivněna supravodivým magnetem CBM. Mimo oblast magnetu není nutné trajektorii částice simulovat, protože oproti oblasti uvnitř magnetu, kde lze magnetické pole *téměř* považovat za homogenní, je výhodnější předpokládat, že mimo magnet je intenzita magnetického pole nulová než se pokoušet magnetické pole jinak aproximovat. Navíc algoritmus pro sledování trajektorií v detektoru používá ve své první fázi pouze data z detektoru STS, který je celý uvnitř magnetu, tvar trajektorie ve vzdálenosti od terče větší než 1 metr pro nás tedy již není zajímavá.

Spojitý simulátor používá Runge-Kuttovu metodu pro simulaci trajektorie, jednotlivé kroky pak zapisuje do výstupního *.csv* souboru. Pro tento účel je sice současný návrh simulátoru postačující, takový simulátor ale není schopen přesně určit, kde částice prošla hladinou detektoru. Z tohoto důvodu byl simulátor rozšířen na **kombinovaný simulátor**, díky čemuž jsme schopni obsluhovat *stavové události* (události, které nelze naplánovat, v případě simulátoru částic okamžiky, kdy částice proletí hladinou detektoru).

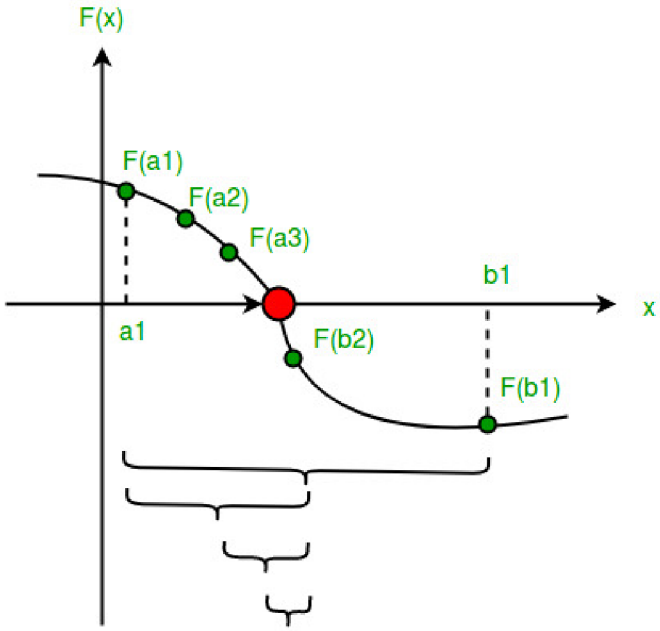
V okamžiku, kdy se změní stavová podmínka (v případě simulátoru částic taková změna souřadnice z, která ukazuje na průlet hladinou detektoru mezi jednotlivými měřeními), je potřeba najít čas, kdy ke změně stavové podmínky (a tedy ke stavové události) došlo. Toho lze docílit například pomocí metody **půlení intervalu**. Mějme numerickou metodu s krokem h, minimálním krokem h_{min} a stavovou podmínkou: krok metody opakujeme tak dlouho, než stavová podmínka změní hodnotu (tj. dojde ke stavové události), v takový okamžik krok zmenšíme na poloviční a obnovíme stav systému z okamžiku před provedením kroku, který stavovou událost vyvolal. Krok metody a postupné půlení délky kroku provádíme tak dlouho, dokud je délka kroku než h_{min}, v tu chvíli víme, že při vyvolání stavové události byl nalezen okamžik jejího vyvolání s přesností $\pm h_{min}$. Existuje varianta tohoto algoritmu, která neobnovuje původní stav systému, ale provádí zmenšený krok „pozpátku“. Tato verze metody sice konverguje rychleji, z programátorského hlediska bylo ale snazší implementovat původní metodu. Tímto způsobem simulátor nachází okamžiky, ve kterých ke stavovým událostem došlo a umožňuje její další zpracování. Stavová podmínka je v tu chvíli potvrzena, vyhodnotí se stavová událost (například zapsání hodnot stavu systému nebo vyhodnocení procesu mnohonásobného rozptylu, který sice tento simulátor zanedbává, při použití ve vlastním sledovacím algoritmu je ale jeho vyhodnocení na místě).

Stavové události jsou vnitřně zpracovávány pomocí jedné funkce, která při každém kroku porovná polohu částice před a po provedení kroku numerické metody. Funkce má uložené z souřadnice všech hladin detektoru, a pokud je některá z těchto hladin **mezi** dvěma hodnotami z souřadnice před a po provedení kroku, vyvolá stavovou událost. Stavové události jsou, podobně jako data o trajektorii částice, ukládány do samostatného *.csv* souboru, který bude po anonymizaci (program do *.csv* souboru zapisuje jednoznačné *id* částice, která zásah způsobila) a zašumění dat možné použít jako testovací data pro vývoj sledovacího algoritmu.

Struktura částice v detektoru

Každá částice je v simulátoru definována jako struktura, nesoucí data o všech vlastnostech a stupních volnosti, potřebných k její jednoznačné definici:

- jednoznačný identifikátor částice, sloužící ke kontrole správnosti rekonstruovaných částic (lze tímto způsobem kontrolovat, jestli zásahy detektoru, které algoritmus spojil jako trajektorii jedné částice, skutečně byly vyvolány stejnou částicí),



Obrázek 4.13: Metoda půlení intervalu: Metoda provede krok a vyhodnotí, jestli se *změnila hodnota stavové podmínky* a v závislosti na hodnotě stavové podmínky buďto provede další krok, nebo podělí délku kroku dvěma a provede krok znovu. Obrázek byl převzat z webu GeeksforGeeks [7]

- počáteční čas simulace částice t, který,
- koncový čas simulace t_{max}, který říká simulátoru, kdy má ukončit práci s danou částicí (ať už díky tomu, že částice opustila detektor, nebo došlo k sekundárnímu rozpadu),
- klidovou hmotnost částice m_0 na základě které dopočítáváme Lorentzův faktor částice pro úpravu vlastností při relativistickém pohybu,
- náboj částice q,
- stavový vektor částice s informacemi o poloze a rychlosti částice,
- Lorentzův faktor γ,
- Konstantu $C_{\gamma qm} = \frac{q}{\gamma m}$, kterou použijeme při výpočtu zrychlení.

simulace průletu částice detektorem

Průlet částice detektorem je simulován tak, že nejprve inicializujeme stavový vektor částice, počáteční a koncový čas simulace, její náboj a klidovou hmotnost. Pro zjednodušení generování nových částic jsou v rámci souboru `physics.h` definovány základní potřebné konstanty jako π, rychlost světla c nebo elementární náboj e. Dále je v souboru definován katalog klidových hmotností a kvantových nábojů (náboj částice q získáme tak, že její kvantové číslo náboje vynásobíme konstantou elementárního náboje) známých, pozorovatelných částic na základě kterých lze následně dopočítat Lorentzův faktor. Po zapsání hodnot částice je struktura částice předána funkci `calculate_gqm()`, která vypočítá a uloží hodnotu Lorentzova faktoru. V tuto chvíli je částice připravena k simulaci.

Zmiňme ještě funkci `simulation_loop()`, která zprostředkovává vlastní proces numerické simulace. Funkce má na starosti logování dat simulace (za tím účelem jako parametry přijímá dva souborové deskriptory, do kterých data zapisuje: první soubor je určen logování trajektorie, druhý soubor ukládá zásahy v hladinách detektoru), kontrolu (a obsluhu) stavových podmínek a vlastní krokování vybrané numerické metody. V zájmu obecnosti implementace má funkce parametr `num_method`, ukazatel na funkci, pomocí kterého můžeme při volání funkce `simulation_loop()` určit, jakou metodu si přejeme pro simulaci použít (implementovány jsou Eulerova metoda a metoda RK4).

Odvození fyzikálního modelu Kalmanova filtru

Protože Kalmanův filtr, který je nezbytný k fungování sledovacího algoritmu, potřebuje vhodný fyzikální model k tomu, aby správně odhadoval příští stav systému, je výhodné použít stejný model, který již používáme pro generování testovacích dat. Na rozdíl od příkladu použití Kalmanova filtru, kde jsme rovnice převedli na matici (rovnice 4.4), odvodíme fyzikální model tak, jak bychom model odvozovali v případě použití rozšířeného Kalmanova Filtru (EKF).

Mějme soustavu devíti rovnic, vycházejících ze soustavy 4.44 a kinematických rovnic v příkladu pro Kalmanův filtr (viz rovnice 4.4):

$$\begin{aligned}
x_k &= x_{k-1} + \Delta t\, \dot{x}_{k-1} + \frac{\Delta t^2}{2}\ddot{x}_{k-1} \\
\dot{x}_k &= \dot{x}_{k-1} + \Delta t\, \ddot{x}_{k-1} \\
\ddot{x}_k &= \frac{q}{\gamma m}\left(\dot{y}_{k-1}B_z - \dot{z}_{k-1}B_y\right) \\
y_k &= y_{k-1} + \Delta t\, \dot{y}_{k-1} + \frac{\Delta t^2}{2}\ddot{y}_{k-1} \\
\dot{y}_k &= \dot{y}_{k-1} + \Delta t\, \ddot{y}_{k-1} \\
\ddot{y}_k &= \frac{q}{\gamma m}\left(\dot{z}_{k-1}B_x - \dot{x}_{k-1}B_z\right) \\
z_k &= z_{k-1} + \Delta t\, \dot{z}_{k-1} + \frac{\Delta t^2}{2}\ddot{z}_{k-1} \\
\dot{z}_k &= \dot{z}_{k-1} + \Delta t\, \ddot{z}_{k-1} \\
\ddot{z}_k &= \frac{q}{\gamma m}\left(\dot{x}_{k-1}B_y - \dot{y}_{k-1}B_x\right)
\end{aligned} \tag{4.45}$$

Nyní můžeme definovat vektorově ohodnocenou funkci $\mathbf{f}_k(\boldsymbol{r}_k^+) : \mathbb{R}^9 \to \mathbb{R}^9$, která obsahuje pravé strany rovnice 4.45:

$$\boldsymbol{r}_k^- = \mathbf{f}_k\left(\boldsymbol{r}_{k-1}^+\right) = \begin{bmatrix}
x_{k-1} + \Delta t\, \dot{x}_{k-1} + 0.5\Delta t^2 \ddot{x}_{k-1} \\
\dot{x}_{k-1} + \Delta t\, \ddot{x}_{k-1} \\
q\gamma^{-1}m^{-1}\left(\dot{y}_{k-1}B_z - \dot{z}_{k-1}B_y\right) \\
y_{k-1} + \Delta t\, \dot{y}_{k-1} + 0.5\Delta t^2 \ddot{y}_{k-1} \\
\dot{y}_{k-1} + \Delta t\, \ddot{y}_{k-1} \\
q\gamma^{-1}m^{-1}\left(\dot{z}_{k-1}B_x - \dot{x}_{k-1}B_z\right) \\
z_{k-1} + \Delta t\, \dot{z}_{k-1} + \frac{\Delta t^2}{2}\ddot{z}_{k-1} \\
\dot{z}_{k-1} + \Delta t\, \ddot{z}_{k-1} \\
q\gamma^{-1}m^{-1}\left(\dot{x}_{k-1}B_y - \dot{y}_{k-1}B_x\right)
\end{bmatrix} \tag{4.46}$$

Za povšimnutí stojí fakt, že i když zrychlení v jednotlivých směrech oproti příkladu v podkapitole 4.1 již není konstantní, celkové zrychlení zůstává konstantní a model je stále lineární. Nyní, když máme funkci $\mathbf{f}_k$, vytvoříme matici fyzikálního modelu tak, že spočítáme **Jacobiho matici** této funkce:

$$F_k = \frac{\partial \mathbf{f}_k \left(\boldsymbol{r}^+_{k-1}\right)_{(i)}}{\partial \boldsymbol{r}^+_{k-1(j)}} \tag{4.47}$$

Jacobiho matice se v tomto případě vyhodnotí následovně:

$$F_k = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{qB_z}{\gamma m_0} & 0 & 0 & -\frac{qB_y}{\gamma m_0} & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & -\frac{qB_z}{\gamma m_0} & 0 & 0 & 0 & 0 & 0 & \frac{qB_x}{\gamma m_0} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & \frac{qB_y}{\gamma m_0} & 0 & 0 & -\frac{qB_x}{\gamma m_0} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.48}$$

Zde je na místě poznamenat, že protože je model lineární, nejsou v Jacobiho matici žádné proměnné stavového prostoru, a není tedy potřeba model linearizovat. Linearizace je nutná až v případě modelu, který počítá s nehomogenním magnetickým polem.

Jak bylo zmíněno výše, pro homogenní magnetické pole, které je rovnoběžné s některou z os, je možné model ještě mírně zjednodušit. Mějme homogenní magnetické pole, působící ve směru osy y. Pohybová složka ve směru y se tím značně zjednoduší, protože se vždy bude jednat jen o rovnoměrný přímočarý pohyb, což mimo jiné znamená, že můžeme zcela vynechat proměnnou $\ddot{y}_k$ a zmenšit tak velikost stavového vektoru na 8. Výsledná predikční matice pak vypadá následovně:

$$F_k = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{qB_y}{\gamma m_0} & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & \frac{qB_y}{\gamma m_0} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.49}$$

Nevýhody simulačních modelů s časovým rozměrem

Simulátor částic je sice funkční a spolehlivě modeluje i speciální teorii relativity, při pokusu o vytvoření Kalmanova filtru pro použití v rekonstrukčním algoritmu ale vyvstává další problém: Ačkoliv pro naše účely je v pořádku předpokládat, že magnetické pole je konstantní a dokonale rovnoběžné s osou y, v reálném detektoru tento předpoklad zavést nelze. Situaci dále komplikuje fakt, že reálné magnetické pole je natolik komplexní, že se nevyplatí ho popisovat analyticky. To je v současné době řešeno tak, že rekonstrukční program drží v paměti velmi detailní 3D mapu směru a intenzity magnetického pole, se kterým potom pracuje. Z tohoto důvodu v reálném rekonstrukčním programu není pro predikci příštího

stavu použit prostý výpočet $\boldsymbol{r}_k^- = \mathbf{f}_k\left(\boldsymbol{r}_{k-1}^+\right)$, ale následující odhad je získán pomocí numerické simulace od detektorové hladiny a k detektorové hladině b. Simulátor částic tedy využijeme i v rámci vlastního rekonstrukčního algoritmu. Simulátor ale vzhledem k tomu, že pracuje v čase, nedokáže předem určit, kdy simulaci zastavit a pro získání příští predikce musí čekat na stavovou událost. Kontrola stavových podmínek je výpočetně náročná (a v případě devítisložkového stavového vektoru její náročnost dále roste), proto pokud to není nutné, je žádoucí se v rámci vlastního rekonstrukčního programu kombinovaným simulátorům vyhnout.

Analogický problém vzniká v i případě obyčejného, lineárního Kalmanova filtru (viz rovnice 4.49), který používá pro získání dalšího odhadu propagační matici F_k. V příkladu v podkapitole 4.1 jsme mohli pracovat s konstantní predikční maticí $F_k = F_{k-1} = F$, protože jednotlivá měření měla mezi sebou konstantní časový úsek Δt. V případě částicového modelu nejenže je čas mezi jednotlivými měřeními pokaždé jiný, což lze vyřešit zavedením matice F_k s různou délkou kroku, tento čas je navíc pro nás předem neznámý (a ze vzorkovací frekvence detektoru vyplývá, že čas mezi jednotlivými průlety není možné změřit).

Z tohoto důvodu byl vytvořen třetí simulační model, tentokrát postaven tak, nesimuloval pohyb částice v závislosti na **čase**, ale na **souřadnici z**.

4.7 Simulační model pro predikční krok Kalmanova Filtru v experimentu CBM

V reakci na problémy s implementací simulátoru zmíněné výše byl na základě práce Dr. Akishiny [1] vytvořen poslední simulátor, tentokrát zásadně odlišný od předchozích verzí. Simulátor vychází z faktu, že částici v magnetickém poli lze jednoznačně popsat pomocí pěti proměnných (říkáme, že stavový prostor má pět stupňů volnosti):

$$\boldsymbol{r}\left\{x, y, t_x, t_y, q/p\right\} \tag{4.50}$$

Kde x a y jsou souřadnice částice pro danou souřadnici z, $t_x \equiv \tan\theta_x$ a $t_y \equiv \tan\theta_y$ označují sklon směru částice vůči rovinám ZX (t_y) a ZY (t_x) a q/p označuje poměr mezi nábojem a převrácenou hodnotou hybnosti částice. Rovnice pro výpočet Lorentzovy síly přepíšeme následujícím způsobem:

$$\begin{aligned}
\frac{\mathrm{d}t_x}{\mathrm{d}z} &= c(q/p)\, t_r(t_y(B_z + t_x B_x) - B_y(1 + t_x^2)) \\
\frac{\mathrm{d}t_x}{\mathrm{d}z} &= c(q/p)\, t_r(B_x(1 + t_y^2) - t_x(B_z + t_y B_y)) \\
t_r(z) &= \sqrt{t_x^2 + t_y^2 + 1}
\end{aligned} \tag{4.51}$$

Kde c je rychlost světla a Všimněme si, že tento model nepoužívá derivace v čase. To má za následek, že neznáme rychlost částice, v důsledku čehož je obtížné zjišťovat vliv relativity na dráhu konkrétní částice. Výhodou tohoto přístupu ale je, že i když nevíme, jakou rychlostí částice detektorem proletěla, z podstaty modelu ani není třeba rychlost částice znát, protože částice bude mít stejnou trajektorii nezávisle na tom, jestli je částice těžká a pomalá, nebo lehká a rychlá, protože se bude lišit pouze čas, za který částice detektorem proletí, a ten vzhledem k povaze detektoru a rychlosti částic stejně nejsme schopni změřit.

Výhoda takového modelu je v tom, že simulátor založený na takovém modelu můžeme spustit v libovolném místě v detektoru a v libovolném místě detektoru můžeme simulaci

zase zastavit. Tento fakt nám mimo jiné umožní provádět simulace z bodu a do bodu b, což je výhodné pro potřeby rekonstrukčního algoritmu, který nyní může provádět simulace od jedné detektorové hladiny k druhé, ale i pro obecný Kalmanův filtr, jehož predikční matice nyní není závislá na časovém intervalu Δt, který se mezi hladinami detektoru může lišit, ale na místním intervalu Δz, který je v experimentu CBM konstantní (hladiny detektoru jsou rozestavěné po intervalech deseti centimetrů).

Tento přístup má svá vlastní omezení, konkrétně zmíníme dvě:

- vzhledem k tomu, že používáme souřadnici z jako časovou složku, některé možné dráhy částic nelze tímto způsobem simulovat. Konkrétně se jedná o částice, které z detektoru vyletí pod pravým úhlem k ose z, a o částice, které mají dostatečně nízkou energii na to, aby se v detektoru „točily“ (viz částice na obrázku 4.5). Intuitivně existují tato omezení z toho důvodu, že v rámci simulace se nemůžeme „vracet v čase“, z matematického hlediska je to pak z toho důvodu, že hodnoty stavových proměnných t_x a t_y mají v případě pravého úhlu k ose z funkční hodnotu limitně v $\pm\infty$. Pravděpodobnost výskytu natolik pomalé částice je každopádně velmi nízká. Navíc, v důsledku mnohonásobného rozptylu uvnitř detektorových hladin, jehož vliv je silnější pro částice s nižší energií, je prakticky nemožné, aby byl detektor schopen takovou částici zrekonstruovat.
- protože neznáme rychlost částice, vliv speciální relativity na částici je skryt v proměnné p/q. To sice pro potřeby filtrace není problém, v případě, že chceme simulovat dráhu reálné částice, výpočet vlivu relativity na ni je netriviální. Nicméně díky faktu, že model nebere v úvahu čas, není toto omezení nijak závažné.

Simulátor byl implementován na velmi podobné bázi, jako první verze simulátoru pro nerelativistické částice, a díky jeho povaze bylo možné upustit od kombinované simulace a simulovat místo toho každý segment trajektorie zvlášť (vzhledem k tomu, že derivujeme podle osy z, můžeme částici „odstartovat“ v terči a jako z_{max} nastavit první hladinu detektoru ve 30cm, vyčíst hodnoty jeho stavového vektoru v $z_{max} = 30$ a inicializovat simulaci znovu, s hodnotami shodnými, jako na konci simulace, počátečním z v první hladině detektoru a z_{max} nastavíme na druhou hladinu detektoru ve 40 cm atd.), což je nejen výhodné pro vlastní rekonstrukční algoritmus, ale i výpočetně efektivní pro vlastní simulaci částic.

4.8 Problematika paměťové náročnosti rekonstrukčního algoritmu

V současné době existuje návrh řešení akcelerace rekonstrukčního algoritmu pro CBM. Stejně jako algoritmus zvolený pro tuto práci je postavený na konceptech celulárního automatu a rozšířeného Kalmanova filtru. Toto řešení je navrženo pro platformu *Intel Xeon Phi* a využívá paralelního zpracování pomocí SIMD[3] instrukcí. Tento postup má oproti akceleraci na FPGA velkou výhodu, a tou je jeho **operační paměť**: Největší zřejmá slabina hradlových polí v této situaci je velmi malé možnosti operační paměti.

Algoritmus na bázi celulárního automatu byl vybrán z toho důvodu, že je lokální a relativně snadno paralelizovatelný, navíc Kalmanův filtr, který je pro fungování celulárního automatu stěžejní, pracuje stále s těmi stejnými maticemi a data, která zpracoval v rámci minulých iterací algoritmu, zahazuje. Velký problém ale vyplývá z náročnosti práce

[3]SIMD – Single Instruction, Multiple Data (jedna instrukce, více dat)

s magnetickým polem. Současné řešení pracuje s rozsáhlou 3D mapou magnetického pole o velikosti okolo 90 MB [1]. Ukládat pro rychlý přístup takové množství dat pro každé FPGA je nepraktické, nabízí se ale možná řešení:

- najít vhodný způsob zjednodušení magnetického pole: V případě, že najdeme způsob, jak magnetické pole vhodně aproximovat, lze teoreticky výrazně snížit nejen nároky na paměť, ale i zvýšit rychlost zpracování (odpadá potřeba pro každou propagaci provádět RK4 simulaci). Jedním z možných řešení je rozdělit magnetické pole na jednotlivé intervaly mezi hladinami detektoru: v rámci každého „úseku“ pak můžeme směr magnetického pole vhodně *linearizovat* v závislosti na poloze v detektorové hladině. Tímto způsobem lze zredukovat množství dat z 3D mapy na 8 2D map a potenciálně tak značně snížit paměťovou náročnost výpočtů.
- neukládat všechna data o magnetickém poli. Toto řešení je intuitivní: Snažíme se rozdělit prostor detektoru tak, aby každé hradlové pole obsluhovalo určitou část prostoru detektoru. Z toho vyplývá, že obsluhuje-li dané pole pouze určitou část detektoru, není potřeba, aby mělo informace o magnetickém poli v celém detektoru.

Oba přístupy řešení lze v ideálním případě kombinovat pro nalezení ideálního kompromisu mezi rychlostí zpracování a přesností modelu.

Kapitola 5

Závěr

Cílem této práce bylo seznámit se s moderními experimenty na půdě částicové a kvantové fyziky, podrobně prostudovat použité technologie a algoritmy využívané pro rekonstrukci trajektorií v detektorech částic, navrhnout a implementovat rekonstrukční algoritmus na hardwarové platformě. Vzhledem ke rozsáhlosti tématu a matematicko-fyzikálnímu aparátu mimo rámec informačních technologií si tato práce dává také za cíl problematiku rekonstrukce částic v moderních fyzikálních experimentech co nejlépe přiblížit.

V rámci práce jsem se podrobně seznámil s jednotlivými fázemi zpracování dat z detektorů částic, různými technologiemi pro získávání dat z nich a algoritmy pro následnou rekonstrukci trajektorií částic, zejména pak pro detektor CBM při výzkumném středisku FAIR v Darmstadtu. Pro podrobnější studium jsem zvolil dvě výpočetně nejnáročnější (a tedy nejvhodnější pro studium možností jejich akcelerace) části zpracování fyzikálních dat z detektorů: rekonstrukce trajektorií částic a odhad (*fitování*) vlastností takto rekonstruovaných trajektorií. Pro akceleraci těchto dvou fází jsem vybral rekonstrukční algoritmus na bázi celulárních automatů a Kalmanových filtrů a po prostudování stávajících řešení pro detektor CBM a detektory mu konceptuálně podobné jsem se rozhodl podrobně zabývat možnostmi akcelerace algoritmu na platformě programovatelných hradlových polí (FPGA).

Vzhledem k neexistenci vhodných testovacích dat (ať už reálných, či simulovaných) bylo třeba vhodná testovací data vytvořit. Z tohoto důvodu byly v průběhu práce pro různé účely postupně navrženy a implementovány čtyři různé verze simulačního modelu chování částic uvnitř detektorů se silným magnetickým polem. Více modelů bylo implementováno z důvodu, že samotný sledovací algoritmus potřebuje používat simulační model pro výpočet predikčního kroku rozšířeného Kalmanova filtru, a bylo tedy potřeba simulační model navrhnout tak, aby potřebám algoritmu co nejlépe odpovídal. Z tohoto důvodu jsem v různých fázích studia rekonstrukčního algoritmu průběžně podobu modelu upravoval tak, aby nejenže vhodně simuloval dráhy částic v detektoru, ale aby umožňoval i „částečnou“ simulaci (tj. aby bylo možné simulovat pouze zadaný úsek dráhy částice).

Ačkoliv jsem věnoval značné množství práce studiu rekonstrukčního algoritmu, z důvodu složitosti technologií a matematicko-fyzikálního aparátu, na kterých je rekonstrukční algoritmus postaven, společně s extrémně přísnými nároky detektoru na rychlost rekonstrukčního algoritmu jsem dospěl k rozhodnutí odložit vlastní implementaci algoritmu na navazující diplomovou práci a věnovat se podrobnému studiu rekonstrukčních algoritmů, technologiím, kterých využívá, implementaci simulátoru testovacích částic a přípravě testovacích dat a hledání případných problémů se zvoleným přístupem, vyplývajících ze zvoleného algoritmu a hardwarové platformy a následnému návrhu jejich řešení.

Přestože se zvolený postup ukázal jako vhodný pro paralelní zpracování, jeho největší nevýhoda spočívá ve vysoké paměťové náročnosti, která značně ohrožuje proveditelnost na programovatelných hradlových polích. V reakci na tento fakt jsem v práci navrhl dvě možná řešení v podobě vhodné úpravy modelu magnetického pole v detektoru, které by mělo vhodně snížit paměťové nároky algoritmu a kterému bych se chtěl dále věnovat v rámci implementace vlastního algoritmu v navazující diplomové práci. V rámci dalšího pokračování této práce je také možné prostudovat, jaký vliv by na vlastnosti rekonstrukčního algoritmu měl jiný způsob linearizace Kalmanova Filtru (například použití *Unscented Kalmanova filtru*). Další příležitosti pokračování práce vidím v dalším vývoji simulátoru částic, kupříkladu zavedení nehomogenního magnetického pole nebo mnohonásobného rozptylu. Další možná pokračování jsou například zavedení elektrického pole v okolí středu detektoru nebo radiační ztráty energie.

Literatura

[1] Akishina, V. *Four-dimensional event reconstruction in the CBM experiment*. 2016. Disertační práce. Johann Wolfgang Goethe-Universität, Fachbereich Informatik und Mathematik. Dostupné z: inis.iaea.org/search/search.aspx?orig_q=RN:49093434.

[2] Duncan, A. *The analytic impulse* [online]. Květen 1988 [cit. 2022-7-20]. Dostupné z: www.andrewduncan.net/air/.

[3] Aumann, T. Prospects of nuclear structure at the future GSI accelerators. *Progress in Particle and Nuclear Physics*. 2007, sv. 59, č. 1, s. 3–21. DOI: doi.org/10.1016/j.ppnp.2006.12.018. ISSN 0146-6410. International Workshop on Nuclear Physics 28th Course. Dostupné z: www.sciencedirect.com/science/article/pii/S0146641006000974.

[4] Becker, A. *Covariance Extrapolation Equation: Constructing the process noise matrix Q* [online]. 2022 [cit. 2022-15-07]. Dostupné z: www.kalmanfilter.net/covextrap.html#withQ.

[5] Becker, A. *Multidimensional Kalman Filter: Examples* [online]. 2022 [cit. 2022-05-07]. Dostupné z: www.kalmanfilter.net/multiExamples.html.

[6] Bezanson, J., Edelman, A., Karpinski, S. a Shah, V. B. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*. 2017, sv. 59, č. 1, s. 65–98. DOI: 10.1137/141000671. Dostupné z: doi.org/10.1137/141000671.

[7] *Program for Bisection Method* [online], 06. dubna 2021 [cit. 2022-7-27]. Dostupné z: www.geeksforgeeks.org/program-for-bisection-method.

[8] Curry, E., Guyon, B., Sheridan, C. a Donnellan, B. Developing an Sustainable IT Capability: Lessons From Intel's Journey. *MIS Quarterly Executive*. Červen 2012, sv. 11, s. 61–74.

[9] *Desmos Graphing Calculator* [online]. 2015 [cit. 2022-10-07]. Dostupné z: www.desmos.com.

[10] Duda, R. O. a Hart, P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. jan 1972, sv. 15, č. 1, s. 11–15. DOI: 10.1145/361237.361242. ISSN 0001-0782. Dostupné z: doi.org/10.1145/361237.361242.

[11] *The accelerator facility of FAIR and GSI* [online]. Facility for Antiproton and Ion Research in Europe [cit. 2022-17-04]. Dostupné z: www.fair-center.eu/overview/accelerator.

[12] *The universe in the lab* [online]. Facility for Antiproton and Ion Research in Europe [cit. 2022-17-04]. Dostupné z: www.fair-center.eu/overview.

[13] Fajmon, B., Hlavičková, I., Novák, M. a Vítovec, J. *Numerická matematika a pravděpodobnost: Informační technologie* [online]. VUT v Brně, 2014 [cit. 2022-17-07]. Dostupné z: matika.umat.feec.vutbr.cz/inovace/texty/INM/CZ/INM_plna_verze_CZ.pdf.

[14] Friman, B., Höhne, C., Knoll, J., Leupold, S., Randrup, J. et al. *The CBM Physics Book: Compressed Baryonic Matter in Laboratory Experiments*. Leden 2011. ISBN 978-3-642-13292-6.

[15] Gardner, M. The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*. 1970, sv. 223, s. 120–123, [cit. 2022-10-07]. Dostupné z: web.stanford.edu/class/sts145/Library/life.pdf.

[16] Gorbunov, S., Kebschull, U., Kisel, I., Lindenstruth, V. a Müller, W. Fast SIMDized Kalman filter based track fit. *Computer Physics Communications*. 2008, sv. 178, č. 5, s. 374–383. DOI: doi.org/10.1016/j.cpc.2007.10.001. ISSN 0010-4655. Dostupné z: www.sciencedirect.com/science/article/pii/S0010465507004262.

[17] *Green IT Cube becomes research and transfer center* [online], 25. dubna 2022 [cit. 2022-2-07]. Dostupné z: www.gsi.de/en/start/news/details?tx_news_pi1%5Baction%5D=detail&tx_news_pi1%5Bcontroller%5D=News&tx_news_pi1%5Bnews%5D=5343&cHash=2b65707d6cffc3d40119b25ea374574e.

[18] *Green IT Cube: supercomputing center for GSI and FAIR* [online], 01. června 2021 [cit. 2022-2-07]. Dostupné z: www.gsi.de/en/researchaccelerators/research_an_overview/green-it-cube.

[19] *GSI - About us* [online]. GSI Helmholtzzentrum für Schwerionenforschung [cit. 2022-17-04]. Dostupné z: www.gsi.de/en/about_us.

[20] Hough, P. V. C. Machine Analysis of Bubble Chamber Pictures. *Conf. Proc. C*. 1959, sv. 590914, s. 554–558. Dostupné z: inspirehep.net/files/53d80b0393096ba4afe34f5b65152090.

[21] Kabel, C. *Großbaustelle in Darmstadt: Rohbau des Fair-Tunnels ist fertig* [online]. Frankfurter Rundshau, 7. června 2021. Aktualizováno 10.6.2021 [cit. 2022-18-04]. Dostupné z: www.fr.de/rhein-main/darmstadt/grossbaustelle-in-darmstadt-rohbau-des-fair-tunnels-ist-fertig-90792726.html.

[22] Kari, J. *Cellular Automata*. University of Turku, 2013. 133 s. Dostupné z: www.cs.tau.ac.il/~nachumd/models/CA.pdf.

[23] Kollegger, T. *Green Cube GSI: FAIR Tier 0* [online], 30. března 2015 [cit. 2022-27-04]. Dostupné z: indico.cern.ch/event/325439/contributions/756690/attachments/631354/868855/20150330-TKollegger-GreenCubeGSI.pdf.

[24] Schmiedel, B. *Runge-Kutta 4th order scheme* [online]. Read the Docs, Inc, 2019 [cit. 2022-07-24]. Dostupné z: lowebms.readthedocs.io/en/latest/code/rk4.html.

[25] Thrun, S., Burgard, W. a Fox, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. Leden 2005. 667 s. ISBN 0262201623.

[26] Toquica Cáceres, H. Hough and Watershed Transforms Algorithms Brief Review. Březen 2017, s. 1–11. DOI: 10.13140/RG.2.2.35718.98889. Dostupné z: www.researchgate.net/publication/315495322_Hough_and_Watershed_Transforms_Algorithms_Brief_Review.