ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta Jaderná a Fyzikálně Inženýrská

Studium metodiky detekce lokalizovaných domén se zvýšenou četností nabitých částic metodami s proměnným rozlišením (Výzkumný úkol)

Autor: Radek Šmakal
Vedoucí úkolu: doc. Vojtěch Petráček CSc.
Akademický rok: 2006/2007

Studium metodiky detekce lokalizovaných domén se zvýšenou četností nabitých částic metodami s proměnným rozlišením
(abstrakt)

Tato práce se věnuje problematice jetů - jejich popisu, různým metodám jejich hledání, zvláště metodami multiškálové analýzy. Práce nabízí náhled na MRA a waveletovou analýzu a popis programu vytvořeného pro získání zdrojových dat pro testování MRA.
Klíčová slova: MRA, MSA, jet

(summary)

This work is dedicated to jets and MRA. It describes jets, describes different methods of finding jets, mainly multiresolution methods. This text provide insight into wavelet analysis and into program for generating testing data for new algorithm.
Key words: MRA, MSA, jet

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti použití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 26.8.2007 Radek Šmakal

Poděkování:

Chtěl bych poděkovat doc. Vojtěchu Petráčkovi CSc. za jeho profesionální pomoc, rady a cenné poznámky.

Obsah

Kapitola 1

Úvod

Při srážkách těžkých iontů dochází k produkci velkého množství nabitých částic. Mohou se přitom vyskytnout oblasti se zvýšenou četností nabitých částic a tyto by mohly být detekovatelné pomocí metod s proměnným rozlišením. Ty by mohly být při malých velikostech těchto oblastí účinnější než jiné dostupné metody. Multiškálová analýza (multi-scale analysis - MSA, nebo také multi-resolution analysis - MRA) se dá aplikovat v mnoha různých problémech. Například k pátrání po jetech, právě tomuto je věnována převážná část této práce. Jinou možností je hledání disorientovaného chirálního kondenzátu (tzv. pionový laser), další možností třeba studium dropletů (droplety jsou kapičky kvark-gluonového plazmatu, které by mohly existovat při rozpadu fireballu během úvodních fází hadronizace).

Kapitola 2

MRA

Analýza s proměnným rozlišením [15] (MultiResolution Analysis - MRA), nebo také multiškálová aproximace (MultiScale Approximation - MSA) je skupina metod využívaných při zpracování signálů, kompresi obrazových dat či analýze funkcí. Nejčastěji se spojuje s pojmem waveletové analýzy, o které se zmíním v následující kapitole (budu používat anglický název wavelet, který je nejrozšířenější, někdy v literatuře používaný překlad vlnková analýza se mi nezdá vhodný. Původní název je francouzský - ondulette transformation). MRA se vyvinula z mikrolokální analýzy (používané v teorii diferenciálních rovnic) a z pyramidálních algoritmů.
Mějme prostor $L^2(R)$ - prostor kvadraticky integrovatelných funkcí. Definujeme sekvenci rozlišení, označených celým číslem tak, že všechny detaily signálu (zkoumané funkce) na škálách menších než 2^{-j} jsou potlačeny na rozlišení j. Podprostor funkcí, které obsahují informace o signálu až do škály 2^{-j}, označíme jako V_j. MRA zahrnuje rozklad funkce do systému podprostorů V_j.

Prvním požadavkem je, aby V_j byl obsažen ve všech vyšších podprostorech, tedy
$... \subset V_0 \subset V_1 \subset ... \subset V_n \subset V_{+1} \subset ... \subset L_2(R)$
V literatuře psané anglickým jazykem se tento systém nazývá ïnested subspaces:.
Označme aproximaci funkce $f(t)$ na úrovni j jako $f_j(t)$. Potom zřejmě $f_j(t) \in V_j$.
Rozdíl mezi $f_{j+1}(t)$ a $f_j(t)$ je informace o detailech na úrovni $2^{-(j+1)}$, označme $d_j(t)$.
Potom $f_{j+1}(t) = f_j(t) + d_j(t)$

Podobně můžeme rozločit náš podprostor a dostaneme:
$V_{j+1} = V_j \oplus W_j$,
kde W_j je nazván detailovým prostorem při úrovni rozlišení j a je ortogonální k V_j. Opětovnými rozklady prostoru V dostaneme:
$V_{j+1} = W_j \oplus V_j = W_j \oplus W_{j-1} \oplus V_{j-1} = ... = W_j \oplus W_{j-1} \oplus W_{j-2} \oplus ... \oplus W_{j-J} \oplus V_{j-J}$
Za zmínku stojí, že jakékoli dva detailové prostory s odlišným rozlišením jsou ortogonální a také to, že detailový prostor W_j je ortogonální k prostoru V_k pouze pokud $k < j$.

Druhým požadavkem MRA je to, že všechny kvadraticky integrabilní funkce jsou obsaženy v nejjemnějším rozlišení a pouze nulová funkce je v nejhrubší úrovni.
Při stále hrubším a hrubším rozlišení je stále více detailů odstraněno a v limitě $j \to -\infty$ zůstane jen konstantní funkce (musí být nulová k zajištění kvadratické integrability). V druhém extrému je přídáváno stále více detailů až k dosažení nekonečného rozlišení, až pokryjeme celý prostor kvadraticky integrabilních funkcí.

Třetí požadavek je zajištění dilatační invariance. Dá se vyjádřit jako
$f(t) \in V_j \Longleftrightarrow f(2t) \in V_{j+1}$

Čtvrtý je zajištění translační invariance (jestliže $f(t)$ je z prostoru V_0, potom $f(t-k)$ také, $k \in Z$)

Pátá podmínka je zajištění existence funkce ϕ takové, že její translace jsou ortonormální bází pro V_0. Tato funkce se nazývá škálovací funkcí.

Nyní shrnu všechny podmínky do formální definice MRA

MRA prostoru $L^2(R)$ je řada do sebe vnořených prostorů $\{V_j\}_{j \in Z}$ takových, že:
1)$... \subset V_{-1} \subset V_0 \subset V_1 \subset ... \subset L^2(R)$
2)$\cap V_j = 0, \overline{\cup_j V_j} = L^2(R)$
3)$f(t) \in V_j \Longleftrightarrow f(2t) \in V_{j+1}$
4)$f(t) \in V_0 \Longrightarrow f(t-k) \in V_0$
5) $\exists$ funkce $\phi(t)$, zvaná škálovací funkce, taková, že $\phi(t-k)$ je ortonormální bází V_0

Kapitola 3

Waveletová analýza

Wavelet [1] je matematická funkce, užívaná k rozkladu dané funkce (obdoba Fourierovy transformace) do komponent odpovídajících detailovým funkcím $d_j(t)$, zavedených v kapitole věnované MRA. Waveletová transformace je vyjádření funkce pomocí waveletů. Wavelety jsou vytvořeny dilatací a translací nějaké rychle ubývající (nebo konečné) funkce, zvané mateřský wavelet.
Danou funkci $f(t)$ tedy můžeme rozložit v řadu postupných aproximací pomocí škálovací funkce ϕ (father wavelet), nebo v řadu detailových funkcí pomocí funkce ψ (mother wavelet) - druhý případ je nazván waveletovou analýzou.
Waveletová analýza se rozděluje na diskrétní waveletovou transformaci (Discrete Wavelet Transform - DWT) a spojitou waveletovou transformaci (Continuous Wavelet Transform - CWT)

Spojitá waveletová transformace:
$W_f(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \psi(\frac{t-b}{a}) f(t) dt$
kde $a \in R^+$, $b \in R$ a $\psi(\frac{t-b}{a})$ je waveletová báze.

zpětná transformace pak probíhá následovně: $f(t) = \frac{1}{C} \int_{-\infty}^{\infty} \int_{0}^{\infty} \frac{W_f(a,b)\psi(\frac{t-b}{a})}{a^2} da db$

kde $C = \int_0^{\infty} \frac{|\Psi(\Omega)|^2}{\Omega} d\Omega$
$\Psi(\Omega)$ je Fourierův obraz funkce $\psi(t)$

Diskrétní waveletová transformace:
$k(m,n) = a_0^{-m/2} < \psi(a_0^{-m}t - nb_0), f(t) >= \frac{1}{a_0^{m/2}} \int f(t)\psi(a_0^{-m}t - nb_0) dt$
za podmínky, že báze $\psi(a_0^{-m}t - nb_0)$ je úplná. Zpětná transformace je pak:
$f(t) = \sum_m a_0^{-m/2} \sum_n k(m,n)\psi(a_0^{-m}t - nb_0)$

Kapitola 4

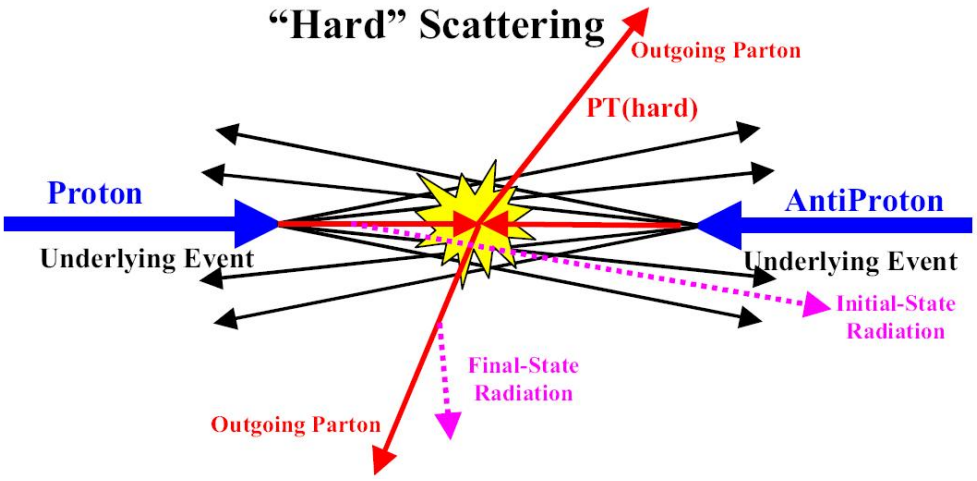
Studium fluktuací metodami s proměnným rozlišením

Vojtěch Petráček [5] vyvinul metodu lokální analýzy s proměnným rozlišením, schopné detekovat lokalizované domény (oblasti s odlišnou distribucí nabitých a neutrálních částic). Tato metoda se ovšem jeví jako perspektivní i v jiných oblastech, ať už při analýze dropletů (přechodné fáze při hadronizaci kvark-gluonového plazmatu), nebo při hledání jetů. Na poslední zmíněnou možnost využití se zaměřuje právě tato práce. Detekční metoda pro studium lokalizovaných domén je založena na použití dvojrozměrné Cauchy-Lorentzovy distribuční funkce $L(\varphi, \eta, \epsilon_\varphi, \epsilon_\eta) = \sum_{k=1}^{N_{cl}} \frac{1}{\pi^2} \frac{(\epsilon_\varphi + \epsilon_{\varphi 0k})(\epsilon_\eta + \epsilon_{\eta 0k}) Q_k e}{\{(\varphi - \varphi_k)^2 + (\epsilon_\varphi + \epsilon_{\varphi 0k})^2\}\{(\eta - \eta_k)^2 + (\epsilon_\eta + \epsilon_{\eta 0k})^2\}}$
kde η, φ jsou pseudorapiditní a azimutální koordinátory místa, ve kterém vyhodnocujeme funkci. ϵ_φ, ϵ_η jsou odpovídající rozlišení, $\epsilon_{\varphi 0k}$, $\epsilon_{\eta 0k}$ jsou polohově závislá intrinsická rozlišení křemíkového driftového detektoru použitého k měření rozdělení nabitých částic, φ_k, η_k jsou polohy průletu jednotlivých částic detektorem. e je efektivita rekonstrukce a Q_k je náboj zanechaný v detektoru k-tou částicí. Zavedeme funkci:
$F(\varphi, \eta, \epsilon 1_\varphi, \epsilon 1_\eta, \epsilon 2_\varphi, \epsilon 2_\eta) = L(\varphi, \eta, \epsilon 2_\varphi, \epsilon 2_\eta) - L(\varphi, \eta, \epsilon 1_\varphi, \epsilon 1_\eta)$
kde $\epsilon 1_\varphi < \epsilon 2_\varphi$ a $\epsilon 1_\eta < \epsilon 2_\eta$.
Můžeme potom napat:
$L(\varphi, \eta, \epsilon 1_\varphi, \epsilon 1_\eta) = L(\varphi, \eta, \epsilon 2_\varphi, \epsilon 2_\eta) + F(\varphi, \eta, \epsilon 1_\varphi, \epsilon 1_\eta, \epsilon 2_\varphi, \epsilon 2_\eta)$
Což je nápadně podobné vztahům v kapitoleo MRA (rozklad funkce pomocí detailových funkcí). Pokud amplituda funkce F v určité oblasti překročí limit šumu (který se dá parametrizovat lineární funkcí celkové četnosti částic), můžeme stanovit velikost oblasti a počet čístic které v ní jsou.

Kapitola 5

Jety

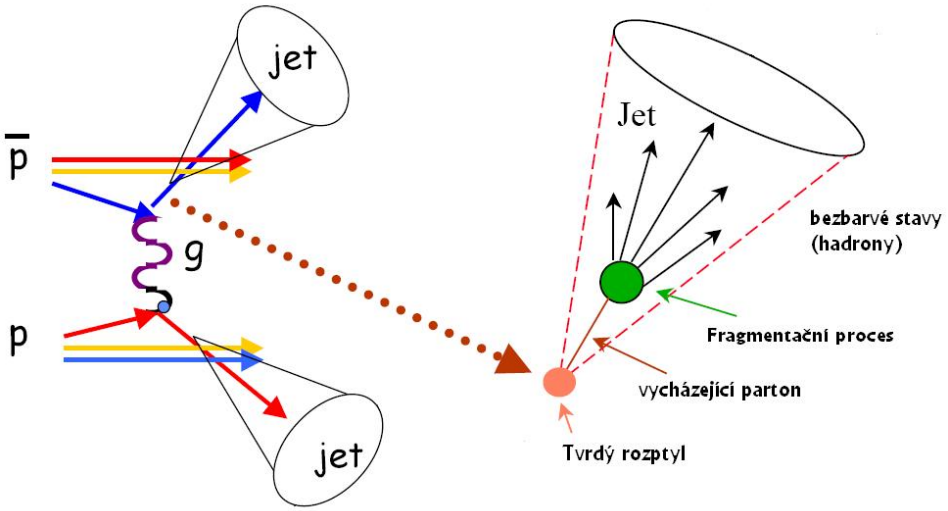
Při kolizi vysokoenergetických hadronů může nastat jeden ze čtyř druhů rozptylové reakce: elastická, difraktivní, měkká neelastická a tvrdá neelastické.
Elastický rozptyl je takový, kde vstupní i výstupní částice jsou stejného typu a energie.Difraktivní rozptyl je takový elastický rozptyl, který se přihodí, když neelastický proces odstraní částici ze svazku.
Neelastický rozptyl je takový, kdy se jeden nebo oba participující hadrony rozpadnou. Měkký neelastický rozptyl způsobuje jen malé přenosy hybností. Je popsán pomocí výměny virtuálních hadronů (Reggeho teorie) a zahrnují největší část totálního účinného průřezu. Tvrdý neelastický rozptyl (to je ten, který nás zajímá) je znázorněn na následujícím obrázku:



Obrázek 5.1: *Znázornění tvrdé kolize dvou hadronů - rozptyl dvou kvarků.*

Při tvrdém neelastické rozptylu reagují partony v hadronech přímo. Hadrony se rozpadají a je vytvořeno velké množství částic. Výchozí partony z tvrdých subprocesů (vyprodukovány v prvotních fázích kolizního procesu, v čase menším než 0,01fm/c) se vyvíjejí pomocí měkké kvarkové a gluonové radiace a následně fragmentují do výtrysku částic, tzv. JETu. Hadrony v jetu mají nízké příčné hybnosti vzhledem ke směru mateřského partonu a součet podélných hybností se přibližně rovná hybnosti rodičovského partonu.
Tvrdý neelastický rozptyl způsobuje velké výměny hybnosti (Q), a poskytují sondu do struktury hadronů na malých vzdálenostech. α_s (QCD coupling constant) je malá díky asymptotické volnosti (menší než 0,3) a proto je možno použít k popisu poruchových metod. Měření

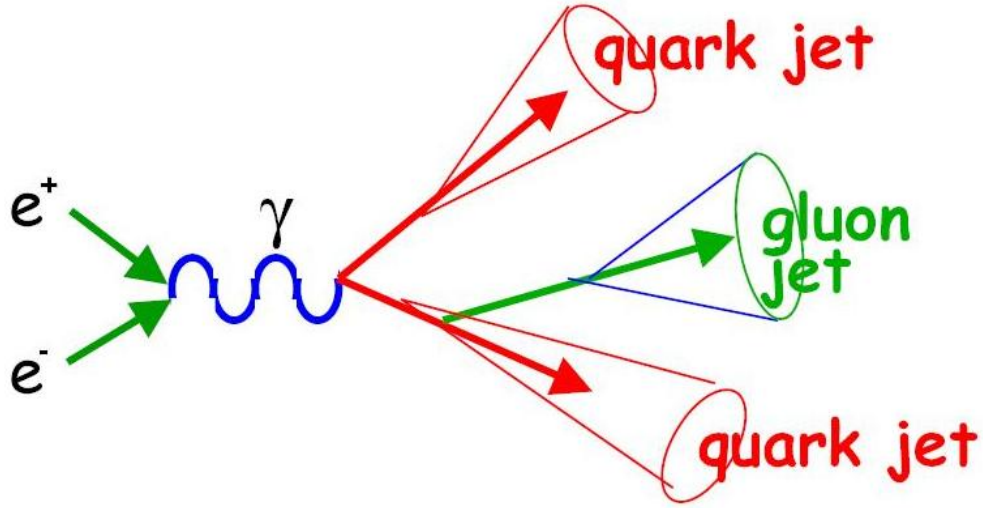
účinných průřezů a dalších vlastností jetů může být použito k testování předpovědí poruchové QCD a také ke zpřesnění s a partonových distribučních funkcí na velkých vzdálenostech.



Obrázek 5.2: *Vznik jetů při tvrdém neelastickém rozptylu.*

Jet silně interaguje s konstituenty média ve kterém vzniká, to je výrazné u srážek těžkých iontů. Jetové partony ztrácejí energii brzdným zářením a přerozptylem (rescattering) partonů s konstituenty média. Intenzita přerozptylu silně roste se zvyšující se teplotou, formace suprahusté horké partonové hmoty ve srážkách těžkých iontů při počáteční teplotě kolem 1GeV může vyústit ve značné energetické ztráty, oproti hadronovému plynu (teplota kolem 0,2GeV), či chladné jaderné hmotě.
Ztráty energie tvrdého partonu v kvark-gluonové plazmě se nazývá JET QUENCHING [13].

Dle QCD se očekává odlišnost mezi kvarkovými a gluonovými jety (výchozím tvrdým partonem byl kvark, respektive gluon). Kvalitativně mají gluonové jety vyšší multiplicitu, měkší fragmentaci a jsou širší co do úhlu [10].

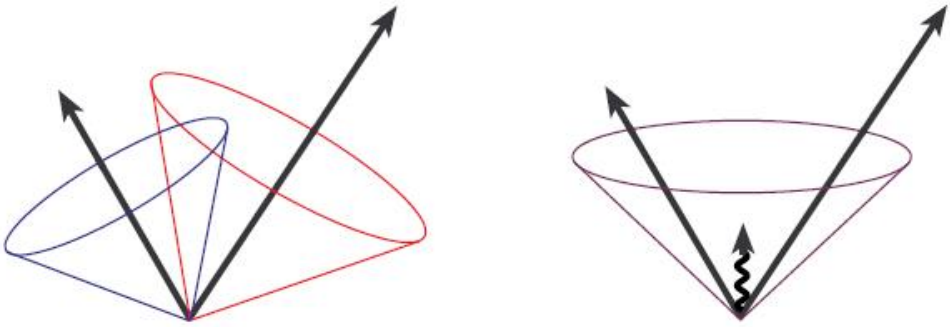


Obrázek 5.3: *Vytvoření dvou kvarkových a jednoho gluonového jetu z elektron-pozitronové anihilace [14]*

Kapitola 6

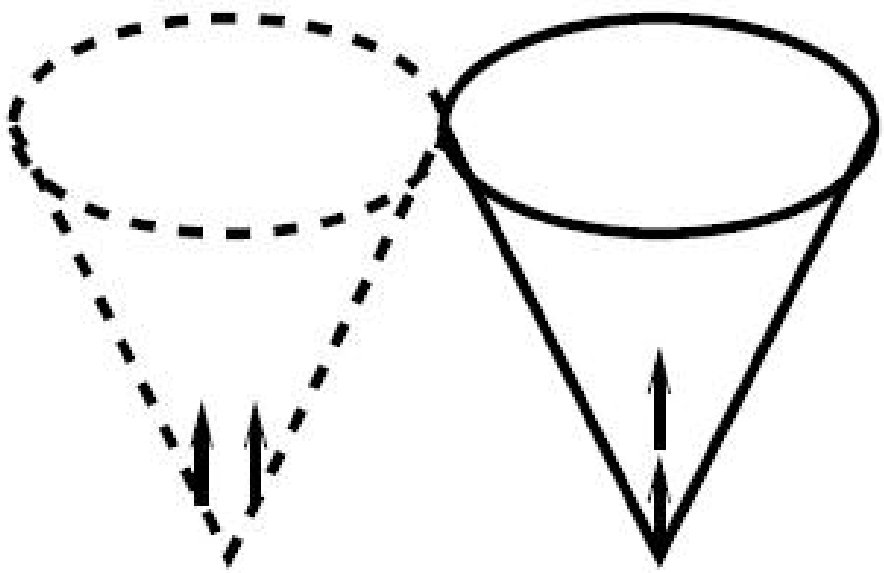
Hledání jetů

Přestože každý pozná jet když ho spatří, samotná definice jetu je obtížná. Bylo vyvinuto několik algoritmů k rozpoznávání jetů. Každý takový algoritmus by měl splňovat několik základních požadavků [9]:
Plná specifikace - proces selekce jetů, kinematické proměnné a nejrůznější korekce by měly být interpretovány jediným způsobem a kompletně definovány.
Správné chování - zajištění infračervené a kolineární bezpečnosti, bez potřeby ad-hoc parametrů
Nezávislost na detektoru - nezávislost na typu detektoru, segmentaci nebo rozměrech
Konzistence - aplikovatelné jak na úrovni teoretické tak i experimentální
První dvě kritéria by měl splňovat každý algoritmus, poslední dvě však nebudou nikdy úplně splněny.



Obrázek 6.1: *Ukázka špatné infračervené bezpečnosti - dva jety mohou být algoritmem rozpoznány jako jediný, což by se nestalo v nepřítomnosti měkkého záření.*

Rozlišují se dvě základní skupiny algoritmů pro hledání jetů - rekombinační (clusterové) a kuželové. Obě jsou založeny na předpokladu, že hadrony náležící jetu budou "blízké" sobě navzájem. Definice kuželových hledačů je založena na blízkosti v reálném prostoru (úhlech), zatímco rekombinační jsou založeny na prostoru hybností (někdy nazývané k_T algoritmy, dle úspěšného algoritmu z této skupiny metod, vyvinutého v roce 1991).



Obrázek 6.2: *Ukázka kolineární citlivosti - v levém případě selhalo nalezení jetu kvůli rozdělení energie do dvou buněk do dvou buněk detektoru a nevytvoří seed (semeno, pokud užijeme doslovný překlad z angličtiny), v pravém případě je vyprodukova seed, protože je energie zanechána v menší oblasti.*

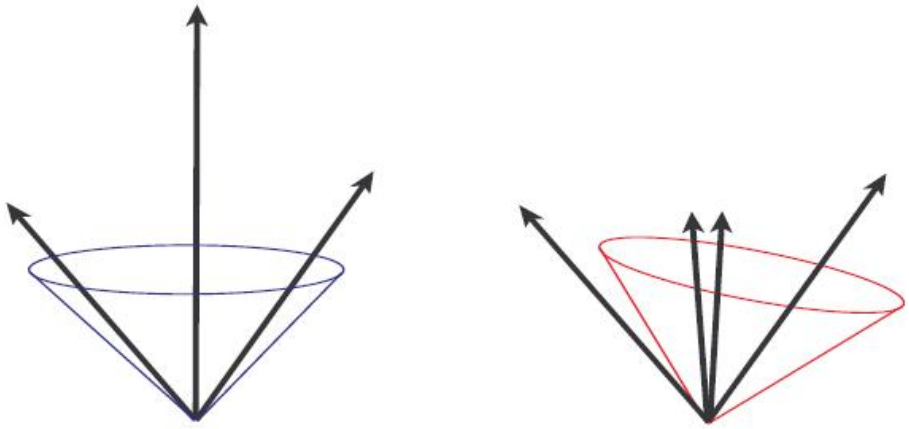
6.1 Rekombinační hledače jetů

Tento typ algoritmů se snaží napodobit hadronizační proces zpětně a spojovat páry částic (vektorů). Typicky obsahují parametr, D, který řídí konec spojování. Díky designu jsou infračerveně a kolineárně bezpečné. Byly původně vyvinuty pro studium jetů z elektron-pozitronových anihilací.
Problematická je aplikace této třídy algoritmů na hadronové kolize. Problémy jsou převážně s odečtením energie ze spektátorových fragmentů a z pozadí mnohonásobné hadron-hadronové interakce. Řešení těchto problémů se začala objevovat až v poslední době.

Algoritmus je přibližně takovýto[**?**]:
1. výpočet vzdáleností
d_{ij} mezi všemi částicemi i a j
d_{iB} mezi i a svazkem
2. nalezení nejmenšího z d_{ij} a d_{iB}
jestliže d_{ij} je nejmenší, rekombinace i a j (do jedné částice)
pokud d_{iB} je nejmenší, i se nazve jetem
3. návrat ke kroku 1), pokud něco chybí

Dvě varianty (a jeden parametr, R)

k_T hledač jetů (1991)
$D_{ij} = min(k_{ti}^2, k_{tj}^2)\Delta R_{ij}^2$, $d_{iB} = k_{ti}^2 R^2$



Obrázek 6.3: *Další ukázka kolineárního problému. V případě vpravo je k rekonstrukci použit první seed (díky špatně zvolenému algoritmu), stojící výše v seznamu, což vyústilo k vypuštění levé částice z jetu.*

Cambridge/Aachen (1998)
$D_{ij} = \Delta R_{ij}^2$, $d_{iB} = R^2$ $[\Delta R_{ij}^2 = \Delta y_{ij}^2 + \Delta \varphi_{ij}^2]$

Standardní C++ a FORTRANovské k_T-algoritmy jsou složitosti N3 (a 1 Pb-Pb event by zabral hodiny!). To je ovšem nejhorší případ, složitost se dá snížit na NlnN (s využitím Voronoiových diagramů, například algoritmus FastJet)

6.2 Kuželové hledače jetů

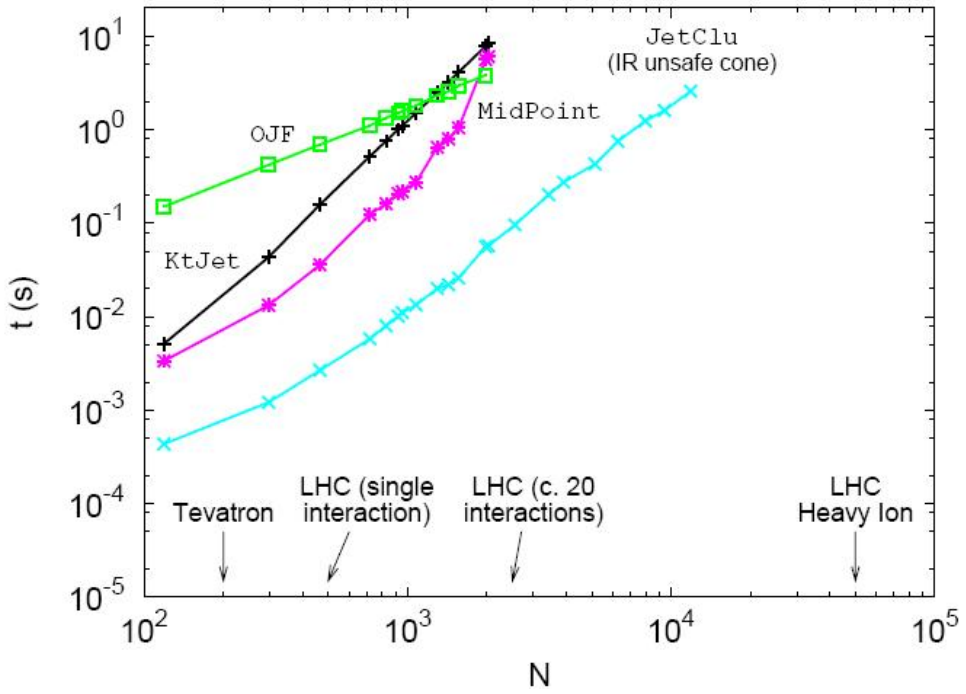
Kuželové algoritmy, původně vyvinuté pro hadron-hadronové kolize, seskupují částice uvnitř kuželu o poloměru R v $\eta \times \varphi$ prostoru do jednoho jetu. R je definováno jako $R = \sqrt{\Delta\eta^2 + \Delta\varphi^2}$, kde $\Delta\eta$ a $\Delta\varphi$ jsou vzdálenosti částic (partonů) v pseudorapiditě a azimutálním úhlu (v radiánech), vzhledem k ose jetu.

Způsob seskupování je takový, že střed kužele je zarovnán v jetovém směru. Typicky algoritmus začíná s několika (vysokoenergetickým) seedy, ale také existují implementace bez seedů. Koncept seedů slouží ke zrychlení výpočtu - místo toho aby pátralo všude po stabilních kuželech, iterační proces začíná pouze v centrech - seedech - buňkách detektoru s velkou deponovanou energií, které překročí jistý minimální energetický cut. Kužely se mohou překrývat, jedna částice může náležet dvěma nebo více kuželům. Proto je třeba definovat proceduru sloužící k rozdělení nebo spojení překrývajících se jetů. Na partonové úrovni výpočty NLO vyžadují přidání ad-hoc separačního parametru Rsep, k regulování shlukování partonů a k simulování role seedů.

Algoritmus je zhruba takovýto:
1. vytvoření seedu (3-vektoru) ze směru každé vstupní částice (je možno implementovat cestu ke snížení počtu seedů a tím ke snížení doby zpracování)

2. Pro každý seed, s, vytvoření kužele v $\eta \times \varphi$ prostoru o poloměru R tak, že častice p s $(\eta_s - \eta_p)^2 + (\varphi_s - \varphi_p)^2 < R^2$ je definována jako uvnitř kužele
3. následná rekombinace každé částice v tomto kuželi do jetu
4. vytvoření nového kužele okolo osy jetu a opakování kroku 3. Jestliže osa nového jetu je kolineární s předešlou osou, potom jet je stabilní a je přidán na list meta-jetů, jinak je proces opakován, dokud se nenajde stabilní jet, nebo dokud není dosaženo maximálního počtu iterací.
5. k zajištění infračervené bezpečnosti, opakování kroků 2-4 s novou sadou seedů mezi každým párem jetů i,j ,nalezenými výše, pokud i a j jsou mezi 1 a 2 poloměry kužele.
Pokud $R^2 < (\eta_i)^2 + (\varphi_i - \varphi_j)^2 < (2R)^2$
Potom $\eta_s = \frac{\eta_i + \eta_j}{2}$, $\varphi_s = \frac{fii + fij}{2}$
6. jakýkoli jet s p_t menším než předdefinovaný parametr epsilon (obvykle řádu 5GeV), je vyřazen ze seznamu.
7. Pro každý jet na seznamu, pokud součet p_t některých částic v jetu, které jsou sdíleny s jetem o vyšším p_t, je větší než nějaký podíl p_t těchto jetů, ovlim, odstranění jetu ze seznamu
8. pro každou částici, která je ve více než jednom jetu, odstraň částici ze všech jetů, kromě toho nejbližšího ke směru částice, např. s jetem o nejmenším $\Delta(\eta)^2 - \Delta(\varphi)^2$
9. Krok 6 je opakován

Z používaných kuželových hledačů uvedu alespoň algoritmus JetClu, který je velmi rychlý, ovšem infračerveně nestabilní na NLO (next-to-leading order), infračerveně stabilnější (infračerveně nestabilní až na NNLO) Midpoint je zase pro změnu stejně pomalý jako k_T.



Obrázek 6.4: *Srovnání rychlosti hledačů jetů, rekombinačních (KtJet) i kuželových (JetClu, MidPoint) a také algoritmu OJF (OptimalJetFinder)*

Kapitola 7

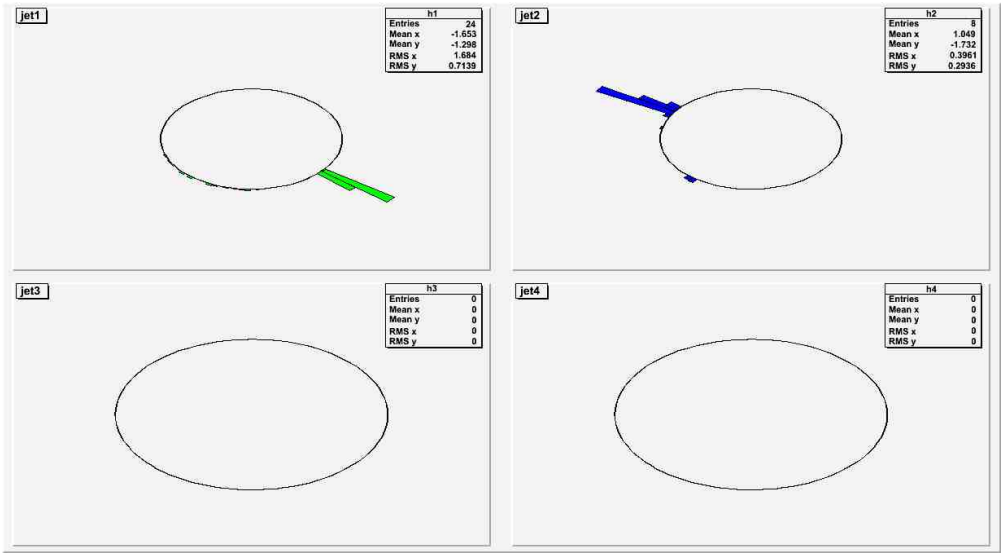
Program pro generování testovacích jetů

Jedním z nejdůležitějších úkolů této práce bylo připravit skript (využívající prostředí ROOT[7] a generátor srážek PYTHIE[6]), který zajistí dostatečné množství rekonstruovaných simulovaných jetů. Zdrojový kód skriptu je uveden v příloze, já pouze okomentuji nejdůležitější části. Používány jsou dvě datové struktury, jedna pro jety a druhá pro částice. U jetu se uchovávají jeho rekonstruované parametry, jako hybnosti a energie (jak v kartézských souřadnicích, tak v η, φ, p_t) a to jak v případě započítání všech částic (i neutrálních), tak pouze pro koncové stabilní nabité částice (piony, kaony, protony, elektrony, a jim příslušné antičástice). Jelikož počet částic se výrazně liší a taktéž počet jetů v jednom eventu (srážce) se může odlišovat, je využita instance TList z prostředí ROOT. Jedná se ve své podstatě o dvousměrný seznam. Výhodnější by bylo použít mnohem jednodušší instanci vector, která je obsažena v knihovnách C++, ovšem problém nastává při kompilaci. Interpreter jazyka C++ v prostředí ROOT (CINT) vyvolal v některých případech úniky paměti (memory leaks) a někdy dokonce došlo až k narušení segmentace (segmentation violation). Tato chyba je zákeřně skryta a projevila se jen když jsem provedl více než šest srážek za sebou a jen pokud měla struktura jet více než jistý počet proměnných typu double. Po korespondenci s vývojáři systému ROOT jsem se proto rozhodl využít dvousměrného seznamu (jinou možností by bylo upravit zdrojový kód a použít interpreter ACLiC). Nyní přejdu k dalším částem skriptu.

```
p->Initialize("cms", "p", "p", 5500);
p->SetCKIN(3,200);
```

První příkaz inicializuje srážku. Jedná se o proton-protonové kolize v CMS systému s energií 5,5TeV. Druhý řádek je horní kinematický cut. Zajímají nás totiž pouze tvrdé procesy. Až v dalších částech práce na projektu se bude přidávat pozadí z měkkých procesů. Po vygenerování srážky následuje cyklus který identifikuje jetové systémy a přidává je do seznamu. Zároveň se provádí rekonstrukce jetu z nabitých i neutrálních částic. Vybírají se pouze jety v jistém, předem nastaveném rozmezí pseudorapidity. Poté následuje nejdůležitější část programu. Prochází se v ní Pylist a u každé koncové částice určeného typu se putuje po odkazech od první mateřské částice stále dál k počátku, až je částice přiřazena některému jetu (ty které jetovému systému nenáleží jsou zahozeny). Informace o příslušnosti k jetu se ukládá do seznamu částic. Chtěl bych upozornit na nepoužití příkazu Pyedit(3). Tento příkaz sice změní Pylist a ukáže jen konečné stabilní nabité částice, které nás zajímají, ale je při tom zničena informace podle které se dá usoudit příslušnost částice k jetovému systému. Další části programu ukládají data do dvou souborů - jeden pouze pro parametry jetů a druhý v

OSCAR1999A formátu (výpis částic v blocích příslušejících jednotlivým jetům a eventům), který bude sloužit jako vstup pro skript provádějící MRA analýzu.



Obrázek 7.1: *Ukázka grafického výstupu z programu - event se dvěma jety při nastavení CKIN(3,100)*

Kapitola 8

Závěr

Hledání jetů pomocí MRA metod může být alternativou ke hledačům zmíněným v kapitole o jetech. Ovšem vývoj zcela nového způsou hledání jetů je opravdu velký cíl, zcela jistě nad rámec této práce. Seznámil jsem se s problematikou aplikace MRA metod, s problematikou produkce jetů a s nástroji pro simulaci srážek a analýzu dat. Nejbližší kroky, které budou brzy podniknuty, je zajištění většího množství dat pro otestování MRA algoritmu a následně samotné testování, nejdříve čistě s jety a poté po přidání pozadí.

```
/*
  JETS generator by Radek Smakal, version 1.4

  1.1 vectors changed to TLists (CINT interpreter doesn't like vectors)
  1.2 change of iterators and object names, some cleaning in the code
  1.3 pseudorapidity cut inserted, fixed problem when no jet is accepted in the eve
  1.4 graphic view of jets added

*/


//using namespace std;


#define PARTICLESFILENAME "jet_particles.osc"  // file of particles sorted into jet
#define JETSFILENAME      "jet_attributes.dat" // file of some jet properties

#define ETA1          -4 // pseudorapidity cut
#define ETA2           4

#define NEVENTS        1 // number of events accepted (with one or more accepted je

Int_t jets()
{

struct jet : public TObject
{
  int jnumber,     // jet number in current event
      n_particles, // number of final stable charged particles in jet (<anti> proto
      startline;   // start line of the jet system in Pylist
  double pjetx,    // momenta from all particles (include photons, zero charged par
         pjety,
         pjetz,
         ptjet,
         etajet,
         fijet,
         pjetx2, // momenta from final stable charged particles
         pjety2,
         pjetz2,
         ptjet2,
         etajet2,
         fijet2;
};

struct particle : public TObject
{
  int pline, // line of particle in Pylist
      jline; // jet identification
```

```
};

int acc_events = 0;

double tanthetajet;
const double theta1 = 2*atan(exp(-(ETA1))), // pseudorapidity to angle
             theta2 = 2*atan(exp(-(ETA2)));

//cout<<tan(theta1)<<endl;
//cout<<tan(theta2)<<endl;

TList jetList,
      particleList;

particle *paccess;
jet *jaccess;

TH2F *h = new TH2F("h","jet histogram view",100,-3.1416,3.1416,100,ETA1,ETA2);

TH2F *h1 = new TH2F("h1","jet1",100,-3.1416,3.1416,100,ETA1,ETA2);
TH2F *h2 = new TH2F("h2","jet2",100,-3.1416,3.1416,100,ETA1,ETA2);
TH2F *h3 = new TH2F("h3","jet3",100,-3.1416,3.1416,100,ETA1,ETA2);
TH2F *h4 = new TH2F("h4","jet4",100,-3.1416,3.1416,100,ETA1,ETA2);
TH2F *h5 = new TH2F("h5","jet5",100,-3.1416,3.1416,100,ETA1,ETA2);

TCanvas *c1 = new TCanvas("c1","blabla",13,49,1400,800);
c1->Range(0,0,1,1);
c1->SetBorderSize(2);
c1->SetTheta(90);
c1->SetFrameFillColor(0);

TPythia6 *p = new TPythia6();
p->Initialize("cms", "p", "p", 5500);

p->SetCKIN(3,200); // set the min Pt to 2 GeV >, this eliminates soft production //
//p->SetMSTP(61,0);//no initial-state showers
//p->SetMSTP(71,0);//no final-state showers
//p->SetMSTP(81,0);//no multiple interactions
//p->SetMSTP(111,0);//no hadronization



 Int_t ks = 0,
       prevks = 0;

 //ofstream particlesout(PARTICLESFILENAME, ios::app);
 //ofstream jetsout(JETSFILENAME, ios::app);
 /*
```

```
particlesout<<"# OSC1999A"<<endl;
particlesout<<"# final_id_p_x "<<endl;
particlesout<<"# Pythia6"<<endl;
particlesout<<"# "<<endl;
particlesout<<"# p+p @ 5,5TeV in CMS"<<endl;
particlesout<<"# 2GeV Pt cut,final stable charged particles, no spectators"<<endl;
particlesout<<"# ??One test particle per each physical particle"<<endl;

jetsout<<"# jetnumber n_final_particles... "<<endl;
 */
double pt,
       eta,
       fi;


for(int event_number = 1; acc_events < NEVENTS; event_number++)
   {
     int accepted = 0; // counter of accepted jets in one event
     p->GenerateEvent(); // make single event
     Int_t nPart = p->GetN(); // number of involved particles in single event

     //p->Pylist(1);
     //cout<<"=================jet list========================="<<endl;
     //p->Pyedit(3);

     if(event_number == NEVENTS)
       {
 // p->Pylist(1);
       }
     //p->Pylist(2);

     int c = 0;

     int m = 0;

     for(Int_t iPart = 2; iPart <= nPart; iPart++) // iPart<=nPart instead ks!=1 be
       {
 ks = p->GetK(iPart,1);
 prevks = p->GetK(iPart-1,1);

 //cout<<"ks = "<<ks<<" prevks = "<<prevks<<endl;

 if((prevks != 12) && (ks == 12)) // first line of the jet system
   {

     jaccess = new jet();
     jaccess->n_particles = 0;
     jaccess->pjetx2 = 0;
```

```
    jaccess->pjety2 = 0;
    jaccess->pjetz2 = 0;
    jaccess->startline = iPart;
    jaccess->pjetx = p->GetP(iPart,1);
    jaccess->pjety = p->GetP(iPart,2);
    jaccess->pjetz = p->GetP(iPart,3);
  }

if((prevks == 12) && (ks == 12)) // line of the jet system
  {
    jaccess->pjetx += p->GetP(iPart,1);
    jaccess->pjety += p->GetP(iPart,2);
    jaccess->pjetz += p->GetP(iPart,3);
  }

if((prevks == 12) && (ks == 11)) // last line of the jet system
  {
    jaccess->pjetx += p->GetP(iPart,1);
    jaccess->pjety += p->GetP(iPart,2);
    jaccess->pjetz += p->GetP(iPart,3);
    jaccess->ptjet = sqrt(jaccess->pjetx*jaccess->pjetx+jaccess->pjety*jaccess->pj

    tanthetajet = jaccess->ptjet/jaccess->pjetz;
    //cout<<tanthetajet<<endl;

    if(((tanthetajet < tan(theta1)) || (tanthetajet > tan(theta2))) && (jaccess->p
      {
jaccess->etajet = (log((sqrt(jaccess->ptjet*jaccess->ptjet+jaccess->pjetz*jaccess-
jaccess->fijet = acos(jaccess->pjetx/jaccess->ptjet);
if((jaccess->pjety) < 0)
  {
    jaccess->fijet = -(jaccess->fijet);
  }
m++;
jaccess->jnumber = m;

jetList.Add(jaccess);
accepted++;
cout<<"Pt = "<<jaccess->ptjet<<" fi = "<<jaccess->fijet<<endl;


// h->Fill(jaccess->fijet,jaccess->etajet,jaccess->ptjet);

      }
    c = iPart;
  }
      }
    // cout<<accepted<<endl;
```

```
    if(accepted != 0) // if no jet is accepted,next event can be started
      {
acc_events++;


    //jetList.Print();
/*
    jet *object;
    TIter nnn(&jetList);
    while(object = (jet*)nnn())
      {
cout<<object->jnumber<<" "<<object->startline<<" "<<object->pjetx<<endl;
      }

*/

    for(Int_t line = (c+1); line <= nPart; line++)
      {
if((p->GetK(line,1) == 1) && ((p->GetK(line,2) == 211)||(p->GetK(line,2) == -211)|
          {
    //cout<<p->GetK(line,1)<<" "<<abs(p->GetK(line,2))<<endl;
    paccess = new particle();
    paccess->pline = line;
    paccess->jline = p->GetK(line,3);
    //cout<<"paccess->pline = "<<paccess->pline<<" paccess->jline = "<<paccess->jl
    while(1)
      {
//jaccess2 = new jet();
//jetList.Print();
jet *jaccess1;
TIter next1(&jetList);
while(jaccess1 = (jet*)next1())
  {

    //cout<<"jaccess1->startline = "<<jaccess1->startline<<" c = "<<c<<endl;
    if(((jaccess1->startline) == (paccess->jline)) || (paccess->jline < c))
      // statement with 'c' is because of primary products which not belong to jet
      {
goto hop;
      }
  }
paccess->jline = p->GetK(paccess->jline,3);
//cout<<"muuuuu"<<endl;
      }
  hop:
    //cout<<"hop"<<endl;
    /*
      boolean zastav = true;
```

```
       while(!zastav) {
       while (!zastav && neco) {
       if (neco) {
       zastav = true;
       }
       }
       }
     */

     jet *jaccess2;
     TIter next2(&jetList);
     while(jaccess2 = (jet*)next2()) // filter out particles from primary products
       {
 //cout<<" huf "<<endl;
 if((jaccess2->startline) == (paccess->jline))
   {

     //cout<<"muf"<<endl;
     jaccess2->n_particles++;
     jaccess2->pjetx2 += p->GetP(paccess->pline,1);
     jaccess2->pjety2 += p->GetP(paccess->pline,2);
     jaccess2->pjetz2 += p->GetP(paccess->pline,3);

     pt = sqrt(p->GetP(paccess->pline,1)*p->GetP(paccess->pline,1)+p->GetP(paccess-
     eta = (log((sqrt(pt*pt+p->GetP(paccess->pline,3)*p->GetP(paccess->pline,3))+p-
     fi = acos(p->GetP(paccess->pline,1)/pt);
     if(p->GetP(paccess->pline,2) < 0)
       {
 fi = -fi;
       }

      h->Fill(fi,eta,pt);
      switch(jaccess2->jnumber)
{
case 1: h1->Fill(fi,eta,pt);
  break;
case 2: h2->Fill(fi,eta,pt);
  break;
case 3: h3->Fill(fi,eta,pt);
  break;
case 4: h4->Fill(fi,eta,pt);
  break;
case 5: h5->Fill(fi,eta,pt);
  break;
}
```

```
      particleList.Add(paccess);
      break;
    }
        }
    }
        }



      /*
        particle *paccess1;
        TIter next3(&particleList);
        while(paccess1 = (particle*)next3())
        {
          cout<<"jline ="<<paccess1->jline<<" pline ="<<paccess1->pline<<endl;
        }
      */
      /*
      //output to OSCAR 1999A file format
      jet *jaccess3;
      TIter next4(&jetList);
      while(jaccess3 = (jet*)next4())
        {
 jaccess3->ptjet = sqrt(jaccess3->pjetx*jaccess3->pjetx+jaccess3->pjety*jaccess3->p
 jaccess3->etajet = (log((sqrt(jaccess3->ptjet*jaccess3->ptjet+jaccess3->pjetz*jacc
 jaccess3->fijet = acos(jaccess3->pjetx/jaccess3->ptjet);

 jetsout<<event_number<<' '<<jaccess3->jnumber<<' '<<jaccess3->pjetx2<<' '<<jaccess
 particlesout<<jaccess3->n_particles<<" 0 ["<<event_number<<' '<<jaccess3->jnumber<
 particle *paccess2;
 TIter next5(&particleList);
 while(paccess2 = (particle*)next5())
   {
     if((paccess2->jline) == (jaccess3->startline))
       {
 particlesout<<paccess2->pline<<' '<<p->GetK(paccess2->pline,2)<<' '<<'0'<<' '<<p->
       }
   }
        }

particlesout<<"0 0"<<endl;
     */
if(accepted != 0)
  {
    //c1->Divide(2,2);
    p->Pylist(1);
   TPad *c1_1 = new TPad("c1_1", "c1_1",0.01,0.51,0.49,0.99);
```

```
  c1_1->Draw();
  c1_1->cd();
  c1_1->SetTheta(90.1);
  c1_1->SetPhi(3.2);
  /*
   //  h->Draw("LEGO2, CYL");
   h->GetXaxis()->SetTitle("angle");
   h->GetYaxis()->SetTitle("pseudorapidity");
   cout<<acc_events<<" accepted jets in this event = "<<accepted<<endl;
   // p->Pylist(1);

   // c1->cd(2);
   //  c1.cd(1);
   */

h1->SetFillColor(kGreen);
h1->Draw("LEGO1, CYL");

c1->cd();

c1_2 = new TPad("c1_2", "c1_2",0.51,0.51,0.99,0.99);
  c1_2->Draw();
  c1_2->cd();
  c1_2->SetTheta(90.1);
  c1_2->SetPhi(3.2);

h2->SetFillColor(kBlue);
h2->Draw("LEGO1, CYL");
// h2->SetTheta(90.0)

c1->cd();

 c1_3 = new TPad("c1_3", "c1_3",0.01,0.01,0.49,0.49);
  c1_3->Draw();
  c1_3->cd();
  c1_3->SetTheta(90.1);
  c1_3->SetPhi(3.2);

h3->Draw("LEGO1, CYL");
h3->SetFillColor(kRed);
//h3->SetTheta(90.0);

c1->cd();

c1_4 = new TPad("c1_4", "c1_4",0.51,0.01,0.99,0.49);
  c1_4->Draw();
  c1_4->cd();
  c1_4->SetTheta(90.1);
```

```
   c1_4->SetPhi(3.2);

 h4->SetFillColor(kYellow);
 h4->Draw("LEGO1, CYL");

 // h5->Draw("LEGO1, CYL, SAME");

  }

jetList.Clear();
}
particleList.Clear();
jaccess->jnumber = 0;
}



//particlesout.close();
//jetsout.close();
}
```

Literatura

[1] A. Akansu, M. Smith, Subband and Wavelet Transforms, Kluwer Academic Publishers, USA 1996.

[2] A. Rosenfeld et al., Multiresolution Image Processing and Analysis, Springer-Verlag, Berlín 1984.

[3] J. Starck, F. Murtagh, A. Bijaoui, Image Processing and Data Analysis, Cambridge University Press, Cambridge 1998.

[4] J. Jan, Číslicová filtrace, analýza a restaurace signálů, VUTIUM, Brno 2002.

[5] V. Petráček, Habilitační práce, České vysoké učení technické v Praze, Praha 2006.

[6] Pythia 6.4 manual, hep-ph/0603175

[7] ROOT User's Guide 5.16, http://root.cern.ch

[8] OSCAR File Format 1999A, http://www-cunuke.phys.columbia.edu/OSCAR/

[9] G. Blazey et al., Run II Jet Physics, arXiv:hep-ex/0005012v2

[10] D. Ward et al., Properties of Quark and Gluon Jets, Nuclear Physics B (Proc. Suppl.) 39B,C (1995) 134-136.

[11] R. Field et al., The Underlaying Event in Hard Scattering Processes, arXiv:hep-ph/0201192v1

[12] G. Salam, Jet-finding for LHC, www.lpthe.jussieu.fr/ salam/repository/talks/ 2006-RobiIs60.pdf

[13] C. Loizides, Phd. thesis - Jet physics in ALICE, Johann Wolfgang von Goethe-Universität, Frankfurt am Main 2005.

[14] N. Varelas, Lectures on Jet Physics, CTEQ Summer School 2000.

[15] http://documents.wolfram.com/applications/wavelet/FundamentalsofWavelets/ 1.2.1.html