ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ԿԻՐԱՌԱԿԱՆ ՄԱԹԵՄԱՏԻԿԱՅԻ ԵՎ
ԻՆՖՈՐՄԱՏԻԿԱՅԻ ՖԱԿՈՒԼՏԵՏ

ԾՐԱԳՐԱՎՈՐՄԱՆ ԵՎ ԻՆՖՈՐՄԱՑԻՈՆ
ՏԵԽՆՈԼՈԳԻԱՆԵՐԻ ԱՄԲԻՈՆ

ԱՎԱՐՏԱԿԱՆ ԱՇԽԱՏԱՆՔ

ԹԵՄԱ՝ ՆՈՒԿԼՈՆՆԵՐԻ ԼԻՑՔԱՓՈԽԱՆԱԿՄԱՆ ՌԵԱԿՑԻԱՆԵՐԻ ՏՎՅԱԼՆԵՐԻ ՊԱՀՈՑԻ ՍՏԵՂԾՈՒՄԸ ՝ ՏՎՅԱԼՆԵՐԻ ՈՐՈՆՄԱՆ, ՆԵՐԿԱՑԱՑՄԱՆ ԵՎ ՎԵՐԼՈՒԾՈՒԹՅԱՆ ՀԱՄԱՐ

ՈՒՍԱՆՈՂՈՒՀԻ՝ ՊԱՊԻԿՅԱՆ ՎԱՐԴԱՆՈՒՇ
ՂԵԿԱՎԱՐ՝ ՆԱՋԱՐՅԱՆ ՋԱՎԵՆ

ԵՐԵՎԱՆ 2010

1 Ներածություն: Նուկլոնի լիցքափոխանակման ռեակցիաները: Գիտափորձի և տեսության անհամաձայնությունները

Ուժեղ փոխազդեցությունների մեխանիզմների և նուկլոնների ներքին կառուցվածքի հետազոտություններում կարևոր դեր են կատարում նուկլոնների լիցքափոխանակման (nucleon charge exchange, NCE) ինքլուզիվ ռեակցիաները, այսինքն պրոտոն-պրոտոն բախումներում նեյտրոնների ինքլուզիվ ծնումը պրոտոնների ֆրագմենտացման կինեմատիկական տիրույթում

$$pp \to nX \tag{1}$$

ինչպես նաև π^+- նեյտրոն և պրոտոն-նեյտրոն բախումներում պրոտոնների ինքլուզիվ առաջացումը նեյտրոնների ֆրագմենտացման տիրույթում

$$\pi^+ n \to pX \tag{2}$$

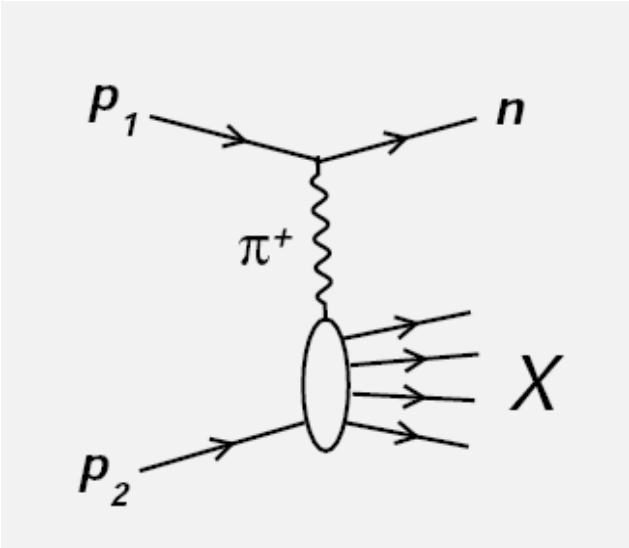
$$pn \to p_n X \tag{3}$$

Վերոհիշյալ ռեակցիաները շարունակաբար հանդիսացել են մանրակրկիտ փորձարարական և տեսական ուսումնասիրությունների առարկա:

(1)-(3) ռեակցիաների փորձարարական հետազոտությունները փնջերի էներգիայի լայն տիրույթում (սկսած 12 գիգաէլեկտրոնվոլտ (ԳԷՎ) փնջի էներգիայից ֆիկսված թիրախի գիտափորձերում մինչև 62.7 ԳԷՎ զանգվածների կենտրոնական համակարգի էներգիան կոլայդերային գիտափրրձերում) իրականացվել են դեռևս 70-ական և 80-ական թվականներին՝ Ֆերմիի Ազգային Լաբորատորիայում (ԱՄՆ) և Միջուկային Հետազոտությունների Եվրոպական Կենտրոնի (CERN, Շվեյցարիա) PS(Proton Synchrotron) ու ISR(Intersecting Storage Rings) արագացուցիչների վրա [1-10]:

Այդ գիտափորձերի մանրամասն տեսական ուսումնասիրությունը կատարվել է [11-15] աշխատանքներում: Տեսական մոտեցման հիմքում ընկած է մեկ պի-մեզոնի փոխանակման (one-pion exchange, OPE) գերակշռության պատկերը՝ սկզբնականից վերջնական նուկլոնին փոխանցված իմպուլսի փոքր արժեքների դեպքում: Այս գերակշռությունը պայմանավորված է պի-մեզոնի փոքր զանգվածով և նուկլոնների հետ դրա ուժեղ կապով:

OPE Ֆեյնմանի դիագրամը պրոցես (1)-ի համար պատկերված է նկար 1-ում: Ընկնող փնջի պրոտոնը առաքում է դրական պի-մեզոն և վեր է ածվում նեյտրոնի: Առաքված պի-մեզոնը փոխազդում է թիրախ հանդիսացող պրոտոնի հետ՝ ստեղծելով ինկլուզիվ X համակարգ:



Նկար 1 Ռեակցիա (1)-ում պրոտոնների ծնման հիմնական մեխանիզմը պատկերող Ֆեյնմանի դիագրամը:

[11-15]-ի վերլուծությունը ցույց է տվել, որ (1)-(3) ռեակցիաների տվյալների մեծամասնությունը կարող է լավ նկարագրվել OPE մեխանիզմի հիման վրա: Սակայն ISR-ի մեծ էներգիաների տվյալները ռեակցիա (1)-ի վերաբերյալ բացահայտում են հակասական իրավիճակ: Փոխանցված լայնակի իմպուլսի ոչ զրոյական արժեքների դեպքում չափված կտրվածքի արժեքները զգալիորեն ցածր են OPE-ի կանխատեսումներից, մինչդեռ այդ իմպուլսի զրոյական արժեքին համապատասխանող փորձարարական տվյալները լավ համաձայնության մեջ են գտնվում տեսության հետ:

Վերջերս երեք գիտափորձեր ներկայացրեցին նեյտրոնների ինկլուզիվ սպեկտրների իրենց ուսումնասիրությունները: Դրանցից երկուսում չափված է ռեակցիա (1)-ը: CERN-ի NA49 գիտափորձում փնջի էներգիան կազմել է 158 ԳէՎ [16,17], մինչդեռ PHENIX (Բրուքհեյվընի Ազգային Լաբորատորիայի Ռելյատիվիստիկ Ծանր Իոնների Արագացուցչի (ԱՄՆ)) գիտափորձը կատարվել է զանգվածների կենտրոնական համակարգի 200 ԳէՎ էներգիայի համար [18]: NA49 գիտափորձի արդյունքները չեն համընկնում ստանդարտ OPE մեխանիզմի կանխագուշակման և

ISR [3] –գիտափորձի տվյալների հետ, մինչդեռ PHENIX-ի սակավաթիվ տվյալները (միայն երեք կետ) մոտավորապես համընկնում են [3] –ի արդյունքների հետ:

Էլեկտրաձնման պրոցեսներում նեյտրոնների առաջացումը պրոտոնների ֆրագմենտացման տիրույթում՝ վիրտուալ ֆոտոնի և պրոտոնի փոխազդեցության հետևանքով

$$\gamma^* p \to Xn \qquad (4)$$

հետազոտվել է ZEUS և H1 գիտափորձերում DESY –ի կենտրոնի (Գերմանիա) HERA արագացուցչի վրա [19-22]: Ռեակցիա (4)-ը ուսումնասիրվել է սկզբնական էներգիայի, γ^* ֆոտոնի վիրտուալության, նեյտրոնի իմպուլսի տարբեր արժեքների դեպքում: ZEUS գիտափորձում դիտարկված նեյտրոնների սպեկտրի տեսքը վերարտադրում է OPE կանխատեսումները մեծ երկայնակի և փոքր լայնակի իմպուլսների տիրույթում, սակայն կտրվածքի բացարձակ արժեքները շատ ավելի ցածր են, քան տեսական սպասումները:

Մի քանի տեսական աշխատանքներում փորձ է արվել բացատրել HERA-ում դիտարկված նեյտրոնների սպեկտրը: [23]-ի հեղինակները դիտարկել են այսպես կոչված վերացրման մեխանիզմը, որի հետևանքով նեյտրոնները մեծ իմպուլսների տիրույթից անցնում են փոքր իմպուլսների տիրույթ: Սակայն, կասկած է հարուցում վերացրման մեխանիզմի կիրառումը նուկլոնի լիցքափոխանակման պրոցեսներում հադրոնային բախումների դեպքում (ռեակցիաներ (1)-(3)):

Այսպիսով, հանելուկային իրավիճակ է դիտարկվում նուկլոնի լիցքափոխանակման պրոցեսները OPE մեխանիզմի հիման վրա նկարագրելու տեսասկետից: Ի տարբերություն բազմաթիվ ռեակցիաների, որտեղ տեղի ունի պի-մեզոնի փոխանակման հստակ ունիվերսալություն, փորձում չափված նուկլոնի լիցքափոխանակման պրոցեսները բացահայտում են զգալի շեղում այդ ունիվերսալությունից: Սա նշանակում է, որ անհրաժեշտ են այս պրոցեսների մանրակրկիտ հետազոտություններ, նպատակ ունենալով ադեկվատ տեսական պատկերի մշակումը:

Երևանի Ֆիզիկայի Ինստիտուտի ALICE խմբի մասնագետները նախաձեռնել են այդ հետազոտությունները: Ներկայացվող ավարտական աշխատանքը կազմում է վերոհիշյալ հետազոտությունների բաղկացուցիչ մասը, որը ենթադրում է նուկլոնի լիցքափոխանակման պրոցեսների բոլոր կուտակված տվյալների բազայի ստեղծում:

2 Փորձարարական տվյալների պահոցի ստեղծման արդիականությունը: Պահանջները պահոցին: ROOT տեխնոլոգիաների ընտրությունը:

(1) - (4) ռեակցիաներում նուկլոնների սպեկտրները բացատրելու կարևոր փուլ է հանդիսանում գոյություն ունեցող բոլոր փորձարարական տվյալների պատշաճ պահոցի ստեղծումը: Այսպիսի պահոցը մեծ պահանջարկ ունի ֆիզիկոսների շրջանում: Այն պետք է բավարարի հետևյալ պահանջներին՝

1. Փորձարարական տվյալների ցանկացած բազմության արագ ընտրում
2. Տվյալների գրաֆիկական ներկայացման հնարավորություն
3. Տեսական մոդելները փորձարարական տվյալների հետ համեմատման հնարավորություն

ROOT համակարգը [24] բավարարում է վերը նշված պահանջներին:

3 ROOT համակարգի համառոտ ներկայացումը

ROOT-ը օբյեկտ-օրիենտացված աշխատանքային միջավայր է, որը հիմնականում նպատակաուղղված է ապահովելու բարձր էներգիաների ֆիզիկայի տվյալների վերլուծումը: ROOT-ը ստեղծվել է CERN-ում (Միջուկային Հետազոտությունների Եվրոպական Կենտրոն) [25] ֆիզիկոսների և ծրագրավորողների մեծ խմբի ընդգրկմամբ: Ներկայումս ROOT-ը կիրառվում է աշխարհով մեկ սփռված հետազոտական հաստատություններում աշխատող մասնագետների կողմից: Այն ապահովում է տվյալների կուտակում և պահեստավորում, մշակում և վերլուծում, այդ տվյալների 1D, 2D և 3D գրաֆիկական ներկայացում, Monte Carlo-ի մեթոդով մոդելավորում, մինիմիզացում և այլն: ROOT-ն ունի նաև հարուստ մաթեմատիկական գրադարան, որը թույլ է տալիս կատարել բարդ տեսական հաշվարկներ: Նախատեսված է, որ CERN-ի LHC-ի (Մեծ հադրոնային կոլայդեր) վրա կատարվող փորձերը տարեկան կտան մի քանի

տասնյակ Պետաբայթ (10^{15} բայթ) ծավալի տվյալներ և այդ տվյալները կմշակվեն ROOT-ի տեխնոլոգիաների կիրառմամբ:

3.1 CINT հրամանային տողի ինտերպրետատորը

CINT-ը ROOT-ի հրամանային տողի C/C++ ինտերպրետատոր է, որը ներդրված է ROOT համակարգում: CINT հրամանները միշտ սկսվում են կետով: Օրինակ ROOT-ից դուրս գալու համար տպում ենք **.q** :

Հրամանային տողում կարող ենք տպել նաև C++-ով գրված արտահայտություններ: Օրինակ՝

```
root [0] double value = 4

root [1] sqrt(value)

(const double)2.00000000000000000e+00
```

Հրամանային տողը կարող է նաև օգտագործվել բազմատող հրամաններ աշխատեցնելու համար: Բազմատող հրամանը սկսելուց առաջ պետք է տպել { (ձախ ձևավոր փակագիծը), իսկ հրամանն ավարտելուց հետո տպել՝ } (աջ ձևավոր փակագիծը): Ներքևում բերված է 1-ից մինչև 10-ը բնական թվերի գումարի հաշվարկման օրինակը.

```
root [0] {

end with '}', '@':abort > float n, sum = 0;

end with '}', '@':abort > cout << "Input n ";

end with '}', '@':abort > cin >> n;

end with '}', '@':abort > sum = ((1 + n) / 2) * n;

end with '}', '@':abort > cout << "sum =  " << sum << endl;

end with '}', '@':abort > }

Input n 10

sum =  55

(class ostream)2146336
```

Հրամանները հրամանային տողում տպելու փոխարեն դրանք սովորաբար գրվում են կոդային ֆայլում, հայտնի որպես մակրո: Մակրոները կարող են լինել անվանված կամ անանուն:

Անանուն մակրոները պարունակում են ձևավոր փակագծերով պարփակված C++ և CINT հրամանների հաջորդականություններ:

Անանուն մակրոները բեռնվում և աշխատացվում են .x CINT հրամանի միջոցով՝

.x filename

Անվանված մակրոները պարունակում են սահմանված մեկ կամ մի քանի ֆունկցիաներ: Դրանք չպետք է պարփակված լինեն ձևավոր փակագծերով: Անվանված մակրոն աշխատեցնելու համար սկզբից բեռնում ենք մակրոն **.L** հրամանով և հետո տպում ենք մակրոյի ներսում պարունակվող ֆունկցիայի անունը: Օրինակ՝

root [0] .L filename

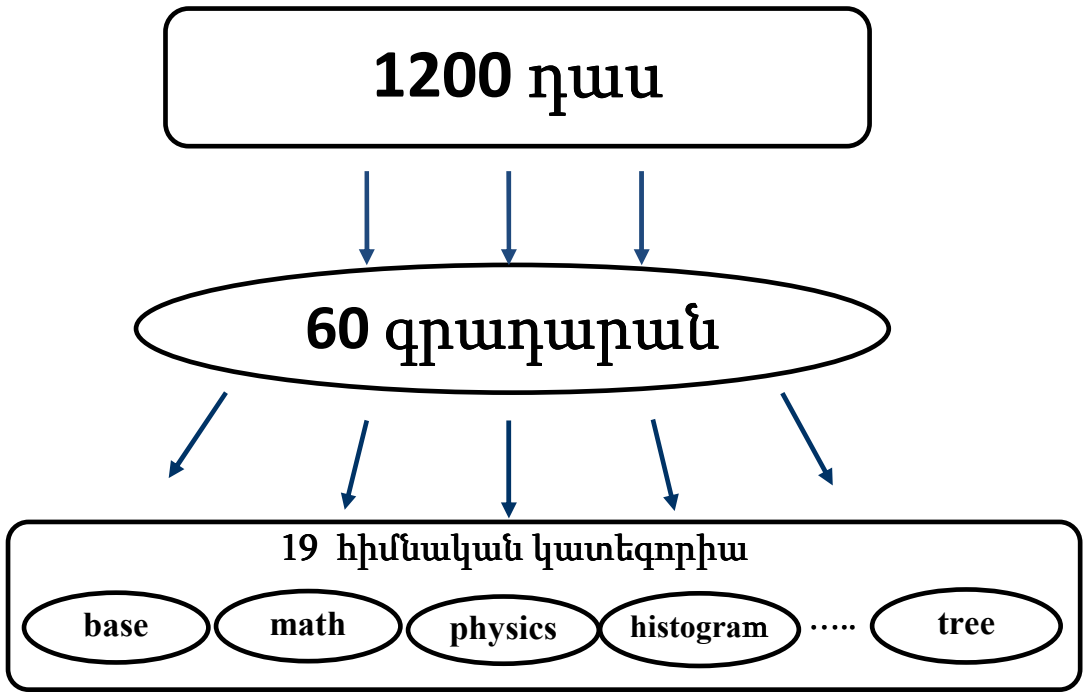
root [1] main()

3.2 ROOT -ում կոդավորման պայմանների առանձնահատկությունները

Դասերի անվանումները սկսվում են T-ով՝	**TObject**
Ոչ դասային տիպերի անվանումները վերջանում են _t –ով՝	**Int_t**
Անդամ տվիալների անվանումները սկսում են f-ով՝	**fTree**
Անդամ ֆունկցիաների անվանումները սկսում են մեծատառով՝	**Loop()**
Հաստատունների անվանումները սկսում են k-ով՝	**kInitialSize**
Գլոբալ փոփոխականների անվանումները սկսում են g-ով՝	**gSystem**
Դասի ստատիկ անդամ տվյալների անվանումները սկսում են fg-ով՝	**fgTokenClient**
Թվարկվող տիպերի անվանումները սկսում են E-ով՝	**EColorLevel**
Լոկալ փոփոխականների և պարամետրերի անվանումները սկսում են փոքրատառերով՝	**nbytes**
Արժեքները ստացող և դնող ֆունկցիաների անվանումները սկսվում են համապատասխանաբար Get-ով և Set-ով՝	**GetFirst()**, **SetLast()**

3.3 ROOT –ի կառուցվածքը

ROOT-ի կառուցվածքը հիմնված է դասերի հիերարխիայի վրա (տես՝նկար 2): ROOT-ը բաղկացած է շուրջ 1200 դասերից, որոնք խմբավորված են 60 գրադարանների: Վերջիններս խմբավորված են 19 հիմնական կատեգորիաների (մոդուլների):



Նկար 2. ROOT-ի դասերի հիերարխիայի սխեմատիկ պատկերը

3.4 ROOT-ի որոշ հիմնական կատեգորիաները

Բազային կատեգորիա. Այս կատեգորիան բաղկացած է ամբողջ ROOT-ի ֆունկցիոնալությունը ապահովող դասերից: Օրինակ, **TObject** դասը (բոլոր ROOT օբյեկտների մայրը) ROOT-ի բոլոր օբյեկտների լռելյայն վարքը (default behaviour), հետևաբար դասերի մեծամասնությունը ժառանգված են **TObject** –ից: Այն ապահովում է արձանագրային մեթոդ I/O օբյեկտի համար, սխալների մեկնաբանում, տեսակավորում, ստուգում, տպում, նկարում և այլն: Մյուս, **TFile,** դասը ապահովում է օբյեկտների պահեստավորումը և դրանց հասանելիությունը օգտագործման համար: **TBrowser** դասը օգտագործելով մենք կարող ենք որոնել ROOT-ի ցանկացած

դասի օբյեկտ, ROOT ֆայլեր (տես գլուխ 4.2) և դրանց պարունակությունը զննել գրաֆիկական պատուհանի միջոցով:

Կոնտեյներ կատեգորիա. Այս կատեգորիայի դասերն ապահովում են տվյալների ընդհանուր բնույթի կառուցվածքներ, ինչպիսիք են զանգվածները, ցուցակները, քարտեզները և այլն:

Մատրիցների կատեգորիա. Պարունակում է դասեր մատրիցների և վեկտորների համար:

Մաթեմատիկական կատեգորիա. Հնարավորություն է տալիս օգտվելու CERN-ի հարուստ մաթեմատիկական գրադարանից:

Հիստոգրամների կատեգորիա. Ապահովում է դասեր վիճակագրական տվյալների վերլուծությունների համար 1D, 2D և 3D հիստոգրամավորման հիման վրա: Թույլ են տրվում short, long, float կամ double արժեքների փոփոխականներ: Այս կատեգորիան հնարավորություն ե տալիս կառուցել ֆիքսված կամ փոփոխվող չափսերի բիններ, պրոֆիլային հիստոգրամներ, ինչպես նաև կատարել բանաձևերի հաշվարկներ:

Ծառերի կատեգորիա. Այս կատեգորիայի դասերից մեկն է մեծ թվով և ծավալներով օբյեկտների/տվյալների/ պահեստավորումը կազմակերպող **TTree** դասը: Այն կարող է պահել բոլոր տեսակի օբյեկտներ, զանգվածներ և պարզ տիպեր: **TTree** դասը օպտիմալացված է այնպես, որ տվյալները քիչ տարածք զբաղեցնեն սկավառակի վրա և ապահովվի դրանց արագ հասանելիությունը: Օգտագործելով ROOT-ի հարուստ գրաֆիկական միջավայրը՝ ծառի օբյեկտների մեջ գրված տվյալները կարող են ներկայացվել տարբեր տիպի հիստոգրամներով և գրաֆիկներով:

Ծառերի կատեգորիան օգտագործվել է սույն դիպլոմային աշխատանքում՝ NCE ռեակցիաների փորձարարական տվյլաները պահեստավորելու և որոնելու համար:

4 Տվյալների հետ աշխատելու ROOT-ի ծառերի տեխնոլոգիան

4.1 TTree և TBranch դասերը

TTree դասի օբյեկտը բաղկացած է ճյուղերից: Վերջիններիս կազմումը թույլ է տալիս օպտիմալացնել տվյալները՝ դրանց հարմար օգտագործելու համար: Ճյուղերի դասը, որը ապահովում է ճյուղերի հետ աշխատանքը, կոչվում է **TBranch**:

Ճյուղ օբյեկտը պարունակում է տերևներ, որոնց մեջ գրվում են փոփոխականների արժեքները՝ մեկ փոփոխականի արժեքներ մեկ տերևում: Երբ փոփոխականները չեն օգտագործվում միասին, դրանք սովորաբար տեղադրվում են առանձին ճյուղերի վրա: Եթե փոփոխականները օգտագործվում են մեկը մյուսի հետ կապված (ինչպիսիք են կետի կոորդինատները) ապա ամենաէֆեկտիվ ձևն է դրանց արժեքները մեկ ճյուղի տերևների մեջ պահելը:

Ճյուղի վրա կարող են պահվել լրիվ օբյեկտներ և դրանց զանգվածները, պարզ փոփոխականներ, ֆոլդերների և **TList**-ի դասի օբյեկտների պարունակությունը, և այլն:

4.1.1 Ծառի ճյուղավորումը

Ծառի ճյուղավորման մակարդակը կարող է ընդունել 1-ից մինչև 99 արժեք:

Եթե ճյուղավորման մակարդակը 1 է, ապա ամբողջ օբյեկտը գրվում է մեկ ճյուղի վրա: Այս դեպքում **TTree**-ի օբյեկտը կունենա ձախ կողմում պատկերված տեսքը՝ մեկ ճյուղով և մեկ տերևով: Եթե **TTree**-ի օբյեկտն ունի մեկից ավելի ճյուղ և ամեն մի ճյուղի վրա մեկ տերև, ապա այն կունենա աջ նկարում պատկերված տեսքը (5 ճյուղանոց օբյեկտ):

Ճյուղավորված ծառը արագ է կարդացվում, որովհետև նույն տիպի փոփոխականները հաջորդաբար են պահվում, հետևաբար փոփոխականի տիպը ամեն անգամ չպետք է կարդացվի:

4.2 ROOT ֆայլը

ROOT ֆայլը սեղմված բինար ֆայլ է, որում կարող ենք պահել տարբեր տիպի օբյեկտներ: Տվյալները կարող են պահվել մեկ կամ մի քանի ROOT ֆայլերում : Մի քանի ֆայլերով տարածված ROOT-ի ծառերը կարող են շղթայվել և հասանելի դառնալ որպես ամբողջ օբյեկտներ՝ ինչը թույլ է տալիս ցիկլերի միջոցով կարդալ տվյալների հսկայական ծավալներ:

ROOT ֆայլը կարող է կառուցվել օպերացիոն համակարգի կողմից ապահովվող ֆայլերի համակարգի նման, այսինքն այն կարող է բաշխվել տարբեր ֆոլդերների (կամ "*դիրեկտորիաների*"), որոնք իրենց հերթին կարող են պարունակել տվյալներ և ֆոլդերներ: ROOT ֆայլով կարելի է “նավարկել” օգտագործելով **TBrowser**-ի գրաֆիկական հնարավորությունները:

5 Պահոցի կառուցման ալգորիթմի ընտրությունը

NCE ռեակցիաներն ուսումնասիրվել են կոլայդեր տիպի (երբ երկու արագացված փնջեր բախվում են իրար) և ֆիքսված թիրախի տիպի (երբ արագացված փունջն ընկնում է անշարժ թիրախի վրա) արագացուցիչների վրա: Ռեակցիա (1)-ի տվյալները ստացվել են ինչպես ֆիքսված թիրախի, այնպես էլ կոլայդերային փորձերում: Ֆիքսված թիրախի փորձերում փնջի իմպուլսի արժեքը կազմել է 12 և 24 ԳԷՎ/c, մինչդեռ բախման էներգիայի արժեքները ISR կոլայդերի փորձերում ընտրվել են 22.5, 30.6, 44.6 և 53.0 ԳԷՎ: Վերոհիշյալ գիտափորձերում չափվել է նեյտրոնների սպեկտրի կախվածությունը Ֆեյնմանի x մասշտաբային փոփոխականից՝ լայնակի իմպուլսի տարբեր ֆիքսված արժեքների դեպքում (0.0-ից մինչև 1.0 ԳԷՎ/c):

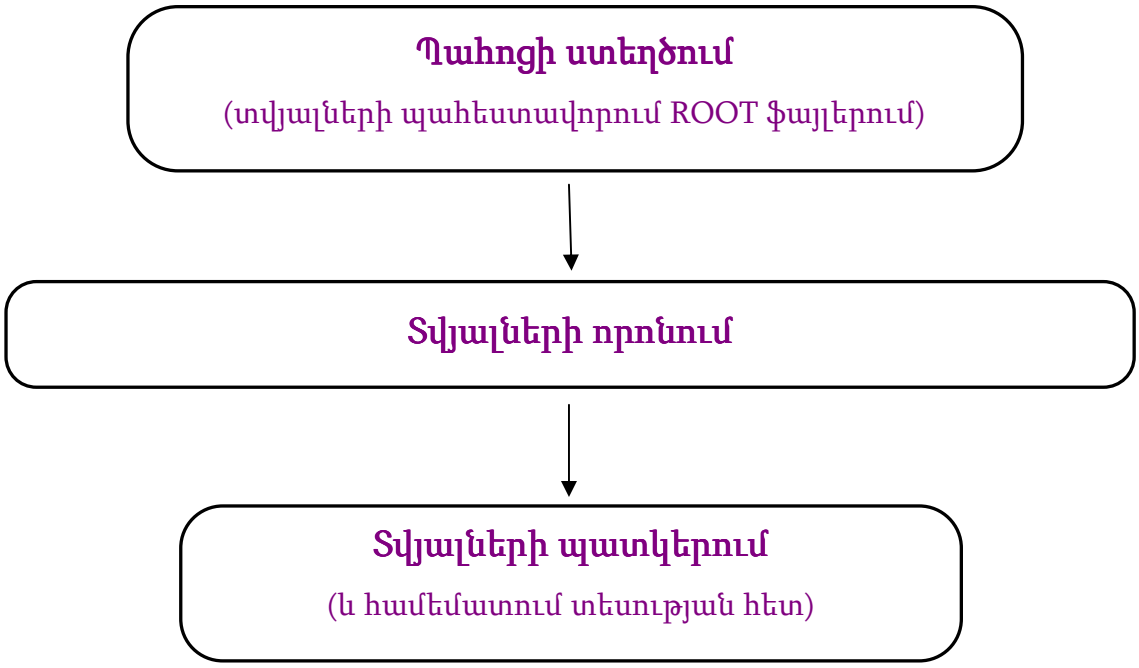
(2) և (3) ռեակցիաները կարող են ուսումնասիրվել միայն ֆիքսված թիրախի գիտափորձերում: Պրոտոնների բաշխումն ըստ x-ի և փոխանցված իմպուլսի քառակուսու ուսումնասիրվել է ռեակցիա (2)-ում, օգտագործելով 100 ԳԷՎ/c իմպուլս ունեցող դրական լիցքի պի-մեզոնների փնջեր և ռեակցիա (3)-ում, օգտագործելով 400 ԳԷՎ/c իմպուլս ունեցող պրոտոններ:

Ռեակցիա (4)-ը հետազոտվել է էլեկտրոն-պրոտոն և պոզիտրոն-պրոտոն փոխազդեցություններում՝ 318 ԳէՎ զանգվածների կենտրոնական համակարգի էներգիայի դեպքում: Ստացվել են նեյտրոննեի ծնման վերաբերյալ տվյալներ՝ γ^*p զանգվածների կենտրոնական համակարգի էներգիայի, ֆոտոնի վերտուալության, ինչպես նաև նեյտրոնների x-ի և լայնակի իմպուլսի արժեքների համար:

Վերը ներկայացված արագացուցիչների և գիտափորձերի տիպերը, ինչպես նաև այդ գիտափորձերում ստացված տվյալների բնույթը, որոշում են մեր պահոցի կառուցվածքը և համապատասխան որոնման ալգորիթմը (տես՝ մաս 6.2):

6 Մակրոյի նկարագրությունը

Ստեղծված մակրոն բաղկացած է երեք գլխավոր բլոկերից (նկար 3): 6.1-6.3 մասերում նկարագրված է այդ բլոկների ֆունկցիոնալությունը:



Նկար 3. Մակրոյի կառուցվածքի սխեմատիկ պատկերը

6.1 Պահոցի ստեղծումը

Նախնական փուլում տվյալները գրվում են ASCII ֆայլերի տեսքով: Ներքևում բերված է այդպիսի ֆայլի օրինակ: Այն պարունակում է ISR կոլայդերի վրա կատարված *pp –> nX* ռեակցիայի բնորոշիչների չափումների արդյունքները: Պրոտոնների բախման էներգիան եղել է 52.8 ԳԷՎ, իսկ ծնված նեյտրոնների լայնակի իմպուլսի արժեքը ֆիքսվել է 0.0 ԳԷՎ/c-ին հավասար : Չափվել է ռեակցիայի լայնակի կտրվածքի կախվածությունը Ֆեյնմանի *x* փոփոխականից:

0.15	**0.0**	**10.9**	**1.4**
0.20	**0.0**	**10.0**	**1.3**
0.25	**0.0**	**9.9**	**1.2**
0.30	**0.0**	**10.0**	**1.2**
0.35	**0.0**	**10.4**	**1.1**
0.4	**0.0**	**10.9**	**1.1**
0.45	**0.0**	**11.4**	**1.0**
0.5	**0.0**	**12.2**	**1.0**
0.55	**0.0**	**13.3**	**1.1**
0.6	**0.0**	**14.7**	**1.1**
0.65	**0.0**	**17.0**	**1.3**
0.7	**0.0**	**19.8**	**1.4**
0.75	**0.0**	**23.6**	**1.6**
0.775	**0.0**	**25.5**	**1.6**
0.8	**0.0**	**26.7**	**1.6**
0.825	**0.0**	**26.7**	**1.6**
0.85	**0.0**	**25.7**	**1.7**
0.875	**0.0**	**23.8**	**1.8**
0.9	**0.0**	**19.3**	**2.6**
0.925	**0.0**	**13.8**	**2.6**
0.95	**0.0**	**8.3**	**2.6**
0.975	**0.0**	**2.7**	**2.6**

Հետևաբար, ASCII ֆալը իրենից ներկայացնում է երեք փոփոխականների արժեքների գրառում: Առաջին, երկրորդ և երրորդ սյունակներում գրառված են համապատասխանաբար *x*-ի (x–փոփոխական), լայնակի իմպուլսի (pt–փոփոխական) և լայնակի կտրվածքի (f–փոփոխական) արժեքները: Չորրորդ սյունակում գրառված են լայնակի կտրվածքի սխալի (deltaf - փոփոխական) արժեքները: Լայնակի կտրվածքը և դրա սխալը բերված են միլիբարն (millibarn) միավորներով (1 միլիբարնը = 10^{-27} սմ^2)

Պահոցի ստեղծմանը վերաբերող մակրոյի մասը բաղկացած է երեք ֆունկցիայից՝

CreateExpDataTree, **CreateBranch** և **FillData**:

CreateExpDataTree ֆունկցիայում կատարվող քայլերը հետևյալն են՝

- Ստեղծվում է ExpData.root ROOT ֆայլը՝

  TFile *f = new TFile(Form("%sExpData.root",gDir.Data()),"RECREATE");

  "RECREATE" օպցիան ստեղծում է ֆայլը, իսկ եթե այն արդեն գոյություն ունի, վրայից գրում է հրամանով ստեղծվածը:

- Կառուցվում է ծառը՝

  TTree *tree = new TTree("T","Experimental data for different experiments");

- Կանչվում է **CreateBranch** ֆունկցիան և կառուցվում են ծառի ճյուղերը, ամեն փորձի համար՝ մեկ ճյուղ, օրինակ՝

  branchName = "Experiment_for_ppnX_ISR_30GeV_pt02";
  fileName = "ppnX_ISR_pt0.2_Engler_et_al30GeV.txt";
  CreateBranch (tree,branchName, fileName);

CreateBranch ֆունկցիայում կատարվող քայլերը հետևյալն են՝

- x, pt, f, և deltaf փոփոխականների արժեքները գրանցելու համար ստեղծվում է ճյուղ x, pt, f, և deltaf տերևներով՝

  tree->Branch(branchName,&ExpData.x,"x/F:pt:f:deltaf");

- Բացվում է տվյալ ճյուղին համապատասխան ASCII ֆայլը՝

  in.open (gDir + fileName);

- ASCII ֆայլը կարդացվում է **FillData** ֆունկցիայով (ifstream տիպի in օբյեկտի օգնությամբ):

FillData ֆունկցիան վերադարձնում է integer տիպի numEntries թիվը, որը հավասար է տվյալ փորձում չափված փոփոխականների արժեքների (ASCII ֆայլում տողերի) թվին:

numEntries = FillData(in,&ExpData,tree);

CreateBranch-ում ստեղծվում է օժանդակ՝ մեկ տերևով ճյուղ (Num) : Տերևում պարունակվում է numEntries փոփոխականի արժեքը՝

tree->Branch(Form("%s_Num", branchName,&numEntries,"numEntries/I");

Վերջին քայլում, **CreateExpDataTree**-ում ստեղծված ծառի օբյեկտը գրվում է

ExpData.root ֆայլում և ֆայլը փակվում է:

```
tree->Write();
f->Close();
```

6.2 Տվյալների որոնումը

Տվյալների որոնման կազմակերպմանը վերաբերոզ մակրոյի մասը հետևյալն է՝

Անհրաժեշտ փորձարարական տվյալները (ճյուղերը) որոնվում են **SearchTBranch** ֆունկցիայի միջոցով: Այս ֆունկցիայում հայտարարվում են ցուցիչներ չորս (a, b, c և d) զանգվածների վրա:
“a” զանգվածում պահվում են արագացուցիչների տիպերը՝

```
Char_t* a[ 2 ] = { "collider", "fixed target" };
```

“b զանգվածում պահվում են ռեակցիաների տիպերը՝

```
Char_t* b[ 4 ] = { "pp->nX", "pn->pX", "pin->pX", "gammap->nX" };
```

“c” զանգվածում պահվում են էներգիաների արժեքները՝

```
Char_t* c [ 4 ] = { "23", "30", "44", "53" }:
```

“d” զանգվածում պահվում են լայնակի իմպուլսի արժեքները՝

```
Char_t* d [ 5 ] = { "0", "0.2", "0.4", "0.6", "0.8"}:
```

Ընտրած փորձարարական տվյալների որոնումը կատարվում է հետևալ կերպով՝

- Բեռնվում է մակրոն՝

  ```
  root [ ] .L  macro.C
  ```

- Աշխատեցվում է SearchTBranch ֆունկցիան՝

  ```
  root [ ] SearchTBranch( )
  ```

- Էկրանի վրա արտածվում են հետևյալ տողերը՝

  ```
  Choose accelerator type
  1 collider
  2 fixed target
  ```

Կոլայդեր տիպի արագացուցչի տվյալները ստանալու համար պետք է մուտքագրել “1”, ֆիքսված թիրախի տիպի – “2”: Մուտքագրված թիվը պահվում է ֆունկցիայում սահմանված չորս միավոր երկարությամբ integer տիպի “z” զանգվածում:

- Այնուհետև արտածվում են հետևյալ տողերը՝

  Choose type of reaction

  1 pp->nX

  2 pn->pX

  3 pip->pX

  4 gammap->nX

Նմանապես ընտրվում են փորձի էներգիայի և լայնական իմպուլսի արժեքները և մուտքագրված թվերը հերթականությամբ պահվում են “z” զանգվածում:

Մեկ ուրիշ՝ “f” երկչափ զանգվածում, պահված են ճյուղերի անվանումները: “z”-ի արժեքները օգտագործելով որպես “f”-ի ինդեքս, ստանում ենք որոնվող ճյուղի անվանումը: Վերջինս էլ փոխանցվում է **ReadData** ֆունկցիային՝ տվյալները պատկերելու համար:

6.3 Տվյալների պատկերումը

Տվյալների պատկերման կազմակերպմանը վերաբերոզ մակրոյի մասը հետևյալն է՝

ReadData ֆունկցիայում՝

- բացվում է ROOT ֆայլը՝

  TFile* f = new TFile(gDir + "ExpData.root");

- Ստանում ենք ROOT ֆայլում պահված ծառը՝

  TTree* tree = (TTree*)f->Get("T");

- Կարդացվում է (ֆունկցիային փոխանցված արգումենտի արժեքին համապատասխանող) ճյուղը և դրան համապատասխանող Num օժանդակ ճյուղը:

```
ExpData_t Exp1Data;
Int_t num
TBranch* b = tree->GetBranch(Form("%s_Num", branchName));
b->SetAddress(&num);
```

```
b->GetEntry(0);
b = tree->GetBranch(branchName);
b->SetAddress(&Exp1Data);
```

- Տվյալները իրենց չափման սխալների հետ միասին պատկերելու համար օգտագործվում է TGraphError դասը

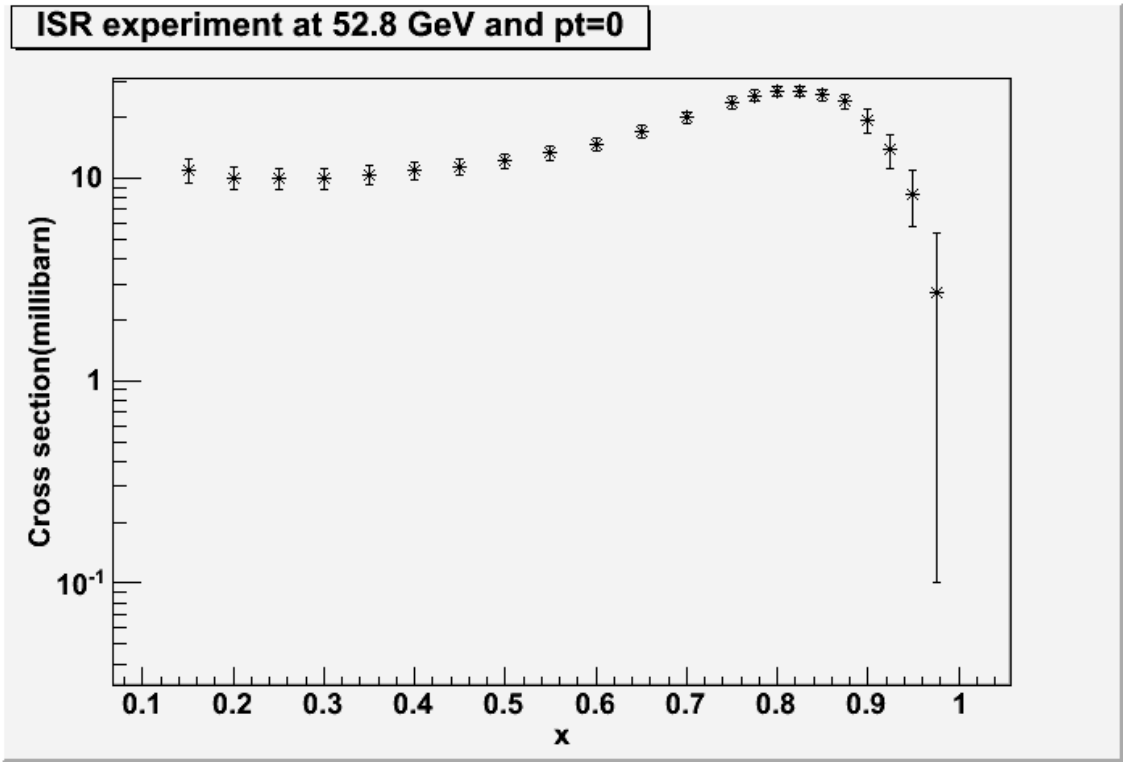
```
TGraphErrors* graph1 = new TGraphErrors(num,x1,f1,deltax1,deltaf1);
```

- Վերջնական գրաֆիկական պատկերը ստանալու համար կիրառվում է Draw մեթոդը

```
graph1->Draw("A*");
```

X (Y)-առանցքի վրա դրված են x (f) փոփոխականի արժեքները:

Նկար 4-ում պատկերված է մակրոյի կիրառման օրինակ: Վորոնվել են ISR կոլայդերի վրա չափված *pp –> nX* ռեակցիայի տվյալները որոնք ստացվել են 52.8 ԳԷՎ էներգիայով բախումներում՝ ֆիքսելով նեյտրոնների լայնակի իմպուլսի արժեքը 0.0 ԳԷՎ –ին հավասար:



Նկար 4 *pp –> nX* ռեակցիայում նեյտրոնների ծնման կտրվածքի կախումը Ֆեյնմանի *x*-փոփոխականից: Տվյալները ստացվել են 52.8 ԳեՎ էներգիայի բախումներում՝ զրո անկյան տակ դիտարկված նեյտրոնների համար:

6.4 Տեսական մոդելին, ինչպես նաև տեսություն-փորձ

համեմատությանը վերաբերող մակրոյի մասը

Թեզի այս գլխում նկարագրվում է տեսական մոդելը և մակրոյի համապատասխան մասը, նվիրված մոդելի կանխագուշակումների համեմատությանը ISR արագացուցչի օգնությամբ ստացած աշխատանք [3] և [1]-ի տվյալների հետ:

6.4.1 Տեսական մոդելը

Պրոցես (1)-ի ինվարիանտ կտրվածքը ունի հետևյալ տեսքը աշխատանք [14]-ի OPE մոդելում

$$f = E_n \frac{d^3\sigma}{d^3 p_n} = \frac{G_{\pi NN}^2 \mid t \mid}{(2\pi)^3 p_{in}\sqrt{s}} \mid F(s, M_X^2, t) \mid^2 \; p_\pi \sqrt{M_X^2}\, \sigma_{tot}^{\pi^+ p}(M_X^2) \qquad (5)$$

Բանաձև (5)-ի մեջ մտնող մեծությունները հետևյալն են'

$$s = (p_1 + p_2)^2 \text{-ը} \qquad (6)$$

s -կանալի ինվարիանտն է, իսկ p_1-ը և p_2-ը բախվող պրոտոնների քառաչափ իմպուլսներն են (տես նկար 1);

$$t = (p_n - p_1)^2 \text{-ը} \qquad (7)$$

t-կանալի ինվարիանտ փոփոխականն է, որն իրենից ներկայացնում է նեյտրոնից պրոտոնին փոխանցված քառաչափ իմպուլսի քառակուսին: p_n-ը նեյտրոնի քառաչափ իմպուլսն է, իսկ E_n-ը դրա ենթերգիան ռեակցիա (1)-ի զանգվածների կենտրոնական համագարգում:

$$M_X^2 = (p_1 + p_2 - p_n)^2 \text{-ը} \qquad (8)$$

X-համակարգի ինվարանտ զանգվածի քառակուսին է:

$G_{\pi NN}$ -ը նուկլոնների հետ պիոնի փոխազդեցաության հաստատունն է

$$G^2_{\pi NN} / 4\pi = 14.6 \qquad (9)$$

Բանաձև (5)-ում p_{in}–ը սկզբնական վիճակի իմպուլսն է զանգվածների կենտրոնական ընդհանուր համակարգում

$$p_{in} = \sqrt{(s - 4M_N^2)} / 2 \qquad (10)$$

Այստեղ M_N=0.94 ԳէՎ մեծությունը նուկլոնի զանգվածն է:

Բանաձև (5)-ում p_π-ն վիրտուալ պիոնի իմպուլսն է պիոնի և երկրորդ պրոտոնի զանգվածների կենտրոնական համազարգում

$$p_\pi = \sqrt{(M_X^2 - M_N^2 - M_\pi^2)^2 - 4M_N^2 M_\pi^2} / 2M_X \qquad (11)$$

M_π = 0.14 ԳէՎ մեծությունը պիոնի զանգվածն է:

$\sigma^{\pi^+ p}_{tot}(M_X^2)$ -ը իրենից ներկայացնում է $\pi^+ p$ փոխազդեցության լրիվ կտրվածքը:

Բանաձև (5)-ի մեջ մտնող F մեծությունը իրենից ներկայացնում է պի-մեզոնի Գրինի ֆունկցիան: Աշխատանք [14]–ի OPE-մոդելի համաձայն, այն պարամետրացվում է հետևյալ տեսքով

$$F(s, M_X^2, t) = \frac{\pi\alpha'}{2\sin(\pi\alpha'(t - M_\pi^2)/2)} \exp\{\Lambda(t - M_\pi^2)\} \qquad |t|<|t_0|\text{-ի համար} \qquad (12)$$

$$F(s, M_X^2, t) = \frac{\pi\alpha'}{2\sin(\pi\alpha'(t_o - M_\pi^2)/2)} \exp\{\Lambda(t - M_\pi^2) + R_2^2(t - t_o)\}$$

$$|t|>|t_0|\text{-ի համար} \qquad (13).$$

(12) և (13) բանաձևերի մեջ մտնող մեծությունների արժեքները հետևյալն են

$\Lambda = R_1^2 + \alpha' \ln(s / s_o)$, R_1^2 = 0.3 ԳէՎ$^{-2}$, R_2^2 = 0.74 ԳէՎ$^{-2}$, α'=1 ԳէՎ$^{-2}$, s_0=1 ԳէՎ^2, t_0= −0.7 ԳէՎ^2:

OPE մոդելի կանխագուշակումները ISR գիտափորձի տվյալների հետ համեմատելու համար անհրաժեշտ է արտահայտել f ինքլուզիվ կտրվածքը նեյտրոնի

Ֆեյնմանի սքեյլինգային x–փոփոխականի և լայնակի p_n^T իմպուլսի միջոցով: Ֆեյնմանի սքեյլինգային x–փոփոխականը սահմանվում է հետևյալ կերպ

$$x = p_n^L / p_{in}\text{, այսինքն} \quad p_n^L = x p_{in} \tag{14}$$

որտեղ p_n^L -ը նեյտրոնի եռաչափ իմպուլսի երկայնակի բաղադրիչն է:

Բանաձև (5)-ի մեջ մտնող մեծություններից հետևյալ փոփոխականներն են կախված x-ից և p_n^T -ից՝ E_n -ը, p_π -ն, t -ն և M_X^2 -ն: E_n-ի կախվածությունը x-ից և p_n^T -ից հետևյալն է

$$E_n = \sqrt{(x\, p_{in})^2 + (p_n^T)^2 + M_N^2} \tag{15}$$

Այնուհետև, t-ն և M_X^2 -ն արտահայտվում են p_n^L-ի, p_n^T -ի և E_n -ի միջոցով

$$t = (E_n - E_{in})^2 - (p_n^L - p_{in})^2 - (p_n^T)^2 \tag{16}$$

$$M_X^2 = s - 2\sqrt{s}\, E_n + M_N^2 \tag{17}$$

p_π-ի համապատասխան կախվածությունը տրվում է (11) բանաձևով :

(11), (16) and (17) արտահայտությունները սահմանում են բանաձև (5)-ի մեջ մտնող բոլոր փոփոխականները որպես x-ի և p_n^T -ի ֆունքցիա:

ISR-ի բախման էներգիաների դեպքում, M_X փոփոխականի արժեքները մեծ են 20 ԳեՎ-ից դիտակվող կինեմատիկական տիրույթում ($1.0>x>0.6$, $p_n^T < 1.0$ ԳեՎ): $\pi^+ p$ փոխազդեցության կտրվածքը այդ արժեքների տիրույթում հաստատուն մեծություն է՝ 22 միլիբարն:

6.4.2 Տեսական մասի ծրագրավորումը

OPE մոդելում $pp->nX$ ռեակցիայի լայնակի կտրվածքի կախվածությունը x սքեյլինգային փոփոխականից նկարագրման մակրոյի մասը իրենից ներկայացնում է f ֆունկցիայի (տես (5)) կառուցում՝ x −ից և p_n^T-ից կախում ունեցող բոլոր փոփոխականների հաջորդաբար կառուցման միջոցով: Ներքևում տրվում է E_n −ի կառուցման մասը

```
Defining function E (Energy of created particle)------------
Double_t E(Double_t xf, Double_t pt)
{
return (sqrt((xf*PA)*(xf*PA)+pt*pt+MN2))
}
```

(11), (16) և (17) բանաձևերի հիման վրա կառուցելով մնացած փոփոխականները և դրանք տեղադրելով (5) արտահայտության մեջ, ստանում ենք ռեակցիայի ինվարիանտ կտրվածքը՝ x-ի և p_n^T-ի ֆունկցիայի տեսքով

```
//**Invariant Rho-function for pp->nX as function of xf and pt **
Double_t RHOxfpt(Double_t xf, Double_t pt)
{
Double_t RHOxfpt = 1.0/(pow(2.*PI,3)*PA*U)*G2*(-t(xf,
pt))*Greenpi(xf,pt)*Greenpi(xf,pt)*
ppivirt(xf,pt)*sqrt(s1(xf, pt))*stotpiplusn;
return(RHOxfpt);
}
```

Տեսական կանխագուշակումները ֆիքսած p_n^T-ով (օրինակը բերվում է p_n^T =0.0 դեպքի համար) գիտափորձերի տվյալների հետ համեմատելու համար, սկզբից սահմանվում է կտրվածքը p_n^T =0.0 կետում

```
//---Invariant Rho-function for pp->nX as function of xf at pt=0.--
Double_t RHOxfptzero(Double_t xf)
{
Double_t RHOxfptzero = RHOxfpt(xf, 0.0);
return(RHOxfptzero);
}
Int_t main()
{
```

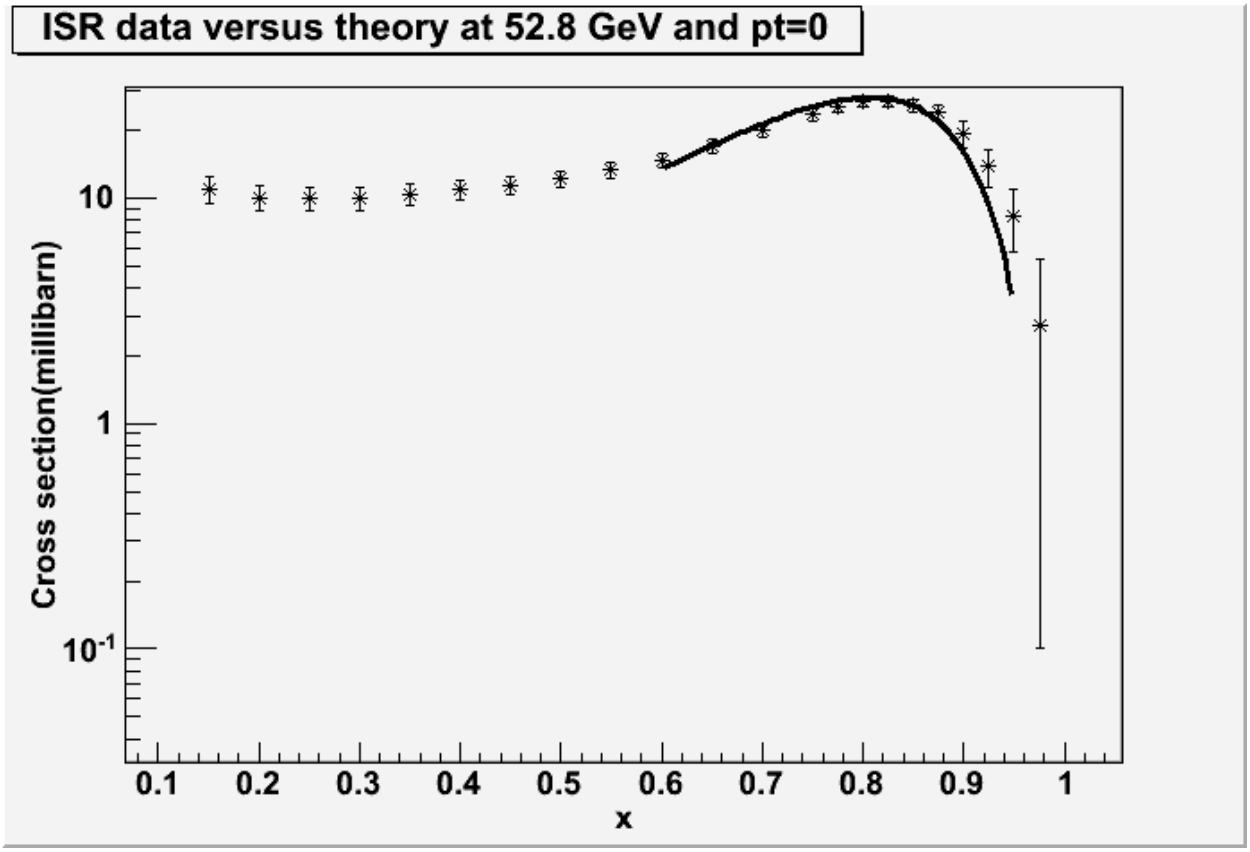
Այնուհետև ստեղծվում է մեկ (x) փոփոխականի ֆունկցիա, օգտագործելով ROOT-ի մեկ արգումենտի ֆունկցիայի կառուցիչը, որը թույլ է տալիս գրաֆիկորեն պատկերել կտրվածքի կախումը x-ից:

```
TF1 * f = new TF1("found", "RHOxfptzero(x)", 0.6, 0.95);
f->Draw();
gPad->SetLogy();
return 0;
}
```

Նմանապես, կարելի է կառուցել տեսական բաշխումներ ըստ x-ի՝ նեյտրոնի լայնակի իմպուլսի ոչ զրոյական արժեքների դեպքում:

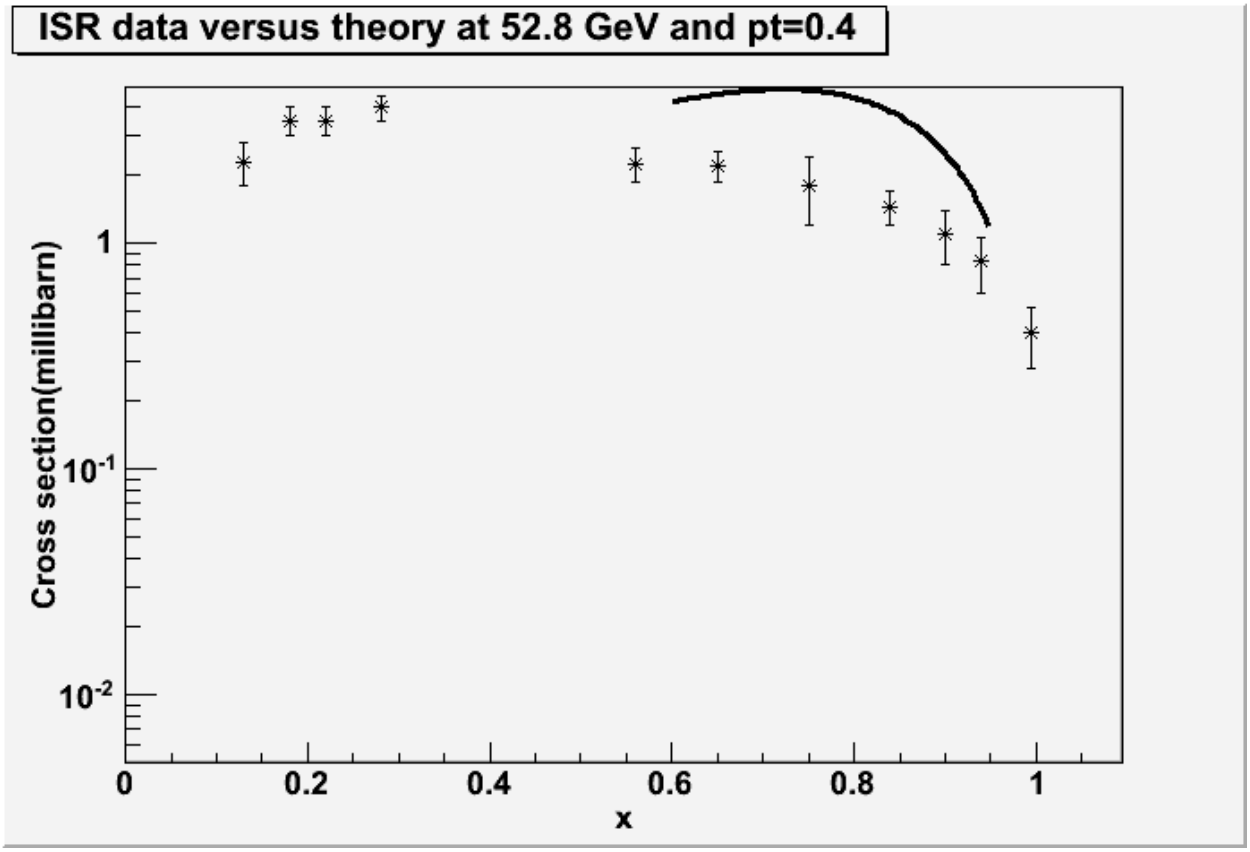
Տեսություն-փորձ համեմատումը

Նկար 5-ում աշխատանք [3]-ի փորձարարական տվյալները համեմատվում են OPE–մոդելի կանխագուշակումների հետ: Համեմատումը ցույց է տալիս, որ OPE–մեխանիզմը լավ է նկարագրում տվյաները մեծ x–երի ($x > 0.6$) արժեքների տիրույթում:



Նկար 5 $pp \to nX$ ինքլուզիվ պրոցեսի կտրվածքի կախվածությունը x-ից՝ 52.8 ԳԷՎ էներգիայի և $p_n^T = 0.0$ ԳԷՎ-ի դեպքում: Աստղանիշերով պատկերված են փորձարարական տվյալները, իսկ հոծ գիծը ներկայացնում է տեսական հաշվարկները:

Նկար 6-ը ներկայացնում է փորձարարական տվյալների և տեսական կանխատեսումների համեմատումը ծնված նեյտրոնների ոչ զրոյական լայնակի իմպուլսի դեպքում (p_n^T = 0.4 ԳէՎ): Տվյաները ստացված են 53.8 ԳէՎ սկզբնական էներգիայի pp- բախումներում [1]: Ինչպես նշված է Ներածությունում, այս դեպքում չափված կտրվածքի արժեքները զգալիորեն (2-3 անգամ) ցածր են OPE-ի կանխատեսումներից $x > 0.6$ տիրույթում:



Նկար 6 pp->nX ինքլուզիվ պրոցեսի կտրվածքի կախվածությունը x-ից' 52.8 ԳէՎ էներգիայի և p_n^T = 0.4 ԳէՎ-ի դեպքում: Աստղանիշերով պատկերված են փորձարարական տվյալները, իսկ հոծ գիծը ներկայացնում է տեսական հաշվարկները:

Գրականություն

1. J. Engler et al. Nucl. Phys. B84 (1975) 70
2. V. Blobel et al.. Nucl. Phys. B135 (1978) 379
3. W.Flauger, J.Monning. Nucl. Phys. B109 (1976) 347
4. R.Robinson et al. Phys. Rev. Lett. 34 (1975) 1475
5. J.Hanlon et al. Phys. Rev. Lett. 37 (1976) 967
6. Y.Eisenberg et al. Nucl. Phys. B135 (1978) 189
7. I.V. Azhinenko et al.. Yad. Fiz. 31(1980) 956
8. P.D.Higgins et al. Phys. Rev. D19, (1979) 731
9. F.T.Dao et al. Phys. Rev. Lett. 37 (1976) 967
10. M.R.Whalley et al., Preprint UM-HE 79149 (1979)
11. A.A. Grigoryan. Preprint ITEP-112 (1975)
12. K.G.Boreskov, A.A.Grigoryan and A.B.Kaidalov, Sov. J. Nucl. Phys. 24 (1976) 411
13. K.G.Boreskov, A.A.Grigoryan, A.B.Kaidalov and I.I.Levintov, Sov. J. Nucl. Phys. 27 (1978) 813
14. G.H.Arakelyan, A.A.Grigoryan. Sov. J. Nucl. Phys. 36 (1982) 211
15. K.G.Boreskov, A.B.Kaidalov and L.A.Ponomarev, Sov. J. Nucl. Phys. 19 (1974) 565
16. D.Varga et al. NA49 collaboration. , Euro. Phys. J. C33, s01, s515 (2004)
17. T.Anticic et al. NA49 collaboration. CERN-PH-EP/2009-006; hep-ex/0904.2708, 2009
18. M.Togawa, for RHIC-PHENIX collaboration. Talk, presented to Diffraction2008; PhD Theses, Kyoto University, 2008.
19. M.Derric et al. ZEUS collaboration, Phys. Lett. B384 (1996) 338
20. S.Chekanov et al. ZEUS collaboration, Phys. Lett. B610 (2005) 199
21. C.Adolff et al. H1 collaboration, Eur. Phys.J. C6, (1999) 587
22. A.Aktas et al. H1 collaboration, Eur. Phys.J. C41, (2005) 273
23. A.B.Kaidalov,. V.A.Khoze, A.D.Martin and M.G.Ryskin. Euro, Phys. J. C 47, 385 (2006); hep-ph/0602215
24. ROOT framework: http://root.cern.ch
25. CERN main page: http://public.web.cern.ch/public/

Հավելված. Լրիվ մակրոն

// main.C

```
#include <iostream>
#include "RHOxfpt04_53GeV_07.04.2010.C"
TString gDir;
void CreateBranch (TTree*, Char_t*, Char_t*, TString );
void CreateExpDataTree();
void ReadData(Char_t*);
Int_t FillData(std::ifstream&,struct ExpData_t*, TTree*);
void SearchTBranch ();
void DrawMyTGraph();

struct  ExpData_t {
        Float_t         x;
        Float_t         pt;
        Float_t         f;
        Float_t         deltaf;

        };

    int main() {
    CreateExpDataTree();
    return 0;
    }
// Create a tree with experimental data.
void CreateExpDataTree() {

    gDir = gSystem->UnixPathName(gInterpreter->GetCurrentMacroName());
    gDir.ReplaceAll("main10_05.C","");
    gDir.ReplaceAll("/./","/");

    // Create a new ROOT file
    TFile *f = new TFile(Form("%sExpData.root",gDir.Data()),"RECREATE");

    // Create a tree
    TTree *tree = new TTree("T","Experimental data for different
experiment");
    Char_t* branchName;
    Char_t* fileName;

    branchName = "Experiment_for_ppnX_ISR_22GeV_pt02";
    fileName = "ppnX_ISR_pt0.2_Engler_et_al22GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_30GeV_pt02";
           fileName = "ppnX_ISR_pt0.2_Engler_et_al30GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_44GeV_pt02";
           fileName = "ppnX_ISR_pt0.2_Engler_et_al44GeV.txt";
    CreateBranch (tree,branchName, fileName);


    branchName = "Experiment_for_ppnX_ISR_53GeV_pt02";
           fileName = "ppnX_ISR_pt0.2_Engler_et_al53GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_30GeV_pt04";
           fileName = "ppnX_ISR_pt0.4_Engler_et_al30GeV.txt";
    CreateBranch (tree,branchName, fileName);
```

```
    branchName = "Experiment_for_ppnX_ISR_44GeV_pt04";
           fileName = "ppnX_ISR_pt0.4_Engler_et_al44GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_53GeV_pt04";
           fileName = "ppnX_ISR_pt0.4_Engler_et_al53GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_30GeV_pt06";
           fileName = "ppnX_ISR_pt0.6_Engler_et_al30GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_44GeV_pt06";
           fileName = "ppnX_ISR_pt0.6_Engler_et_al44GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_53GeV_pt06";
           fileName = "ppnX_ISR_pt0.6_Engler_et_al53GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_30GeV_pt08";
           fileName = "ppnX_ISR_pt0.8_Engler_et_al30GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_44GeV_pt08";
           fileName = "ppnX_ISR_pt0.8_Engler_et_al44GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_53GeV_pt08";
           fileName = "ppnX_ISR_pt0.8_Engler_et_al53GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_30GeV_pt0";
           fileName = "ppnX_ISR_pt0_30GeV.txt";
    CreateBranch (tree,branchName, fileName);

    branchName = "Experiment_for_ppnX_ISR_53GeV_pt0";
           fileName = "ppnX_ISR_pt0_53GeV.txt";
    CreateBranch (tree,branchName, fileName);

       // Write the tree to the file
    tree->Write();
    f->Close();

}
    // Create a branch
void CreateBranch (TTree* tree, Char_t* branchName, Char_t* fileName) {

    //cout<<branchName <<endl<<fileName<<endl;

    ExpData_t ExpData;
    tree->Branch(branchName,&ExpData.x,"x/F:pt:f:deltaf");
    ifstream in;
    //cout<<dir+fileName<<endl;
    in.open (gDir + fileName);
    Int_t numEntries;
    numEntries = FillData(in,&ExpData,tree);
    //cout<<numEntries<<endl;
    in.close();
    tree->Branch(Form("%s_Num", branchName),&numEntries,"numEntries/I");
           tree->Fill();

    }
```

```
// read the ASCII file, put the data into the tree and return number of
entries

Int_t FillData(std::ifstream& in,struct ExpData_t* ExpData, TTree* tree)
{
Int_t numEntries=0;

    while (1) {

        // read x
        in >> ExpData->x;
        if (in.eof()) break;

        // read pt
        in >> ExpData->pt;

        // read f
        in >> ExpData->f;

        // read feltaf
        in >> ExpData->deltaf;

        //cout << ExpData->x << " " << ExpData->pt << " " << ExpData->f << "
" << ExpData->deltaf << endl;

                // Count the number of entries read
        ++numEntries;

        // Put the read data to the tree
        tree->Fill();
    }

    return numEntries;
}

// Search necessary branch of tree
void SearchTBranch () {

    Int_t n = 2, m = 4, r = 5;
    Char_t* a[ 2 ]  = { "collider", "fixed target" };
    //cout<<b[0]<<endl<<b[1]<<endl;

    Char_t* b[ 4 ] = { "pp->nX", "pn->pX", "pip->nX", "gammap->nX" };
    //cout<< a[0]<<endl<<a[1]<<endl<<a[2]<<endl<<a[3]<<endl;

    Char_t* c [ 4 ] = {  "23","30", "44", "53" };
     //cout<<c[0]<<endl<<c[1]<<endl;

    Char_t* d [ 5 ] = { "0", "0.2", "0.4", "0.6", "0.8"};


    Int_t z[ 4 ] ;
    Int_t j;

    cout << "Choose accelerator type " << endl << "1 " << a[0] << endl << "2
" << a[1] << endl;
        cin >> j;
    //cout<<"j= "<<j<<endl;

    while(j != 1 && j != 2) {
          cout << "You entered wrong data. You can choose 1 or 2. Try again.
" << endl;
        cout << "Choose accelerator type " << endl << "1 " << a[0] << endl <<
"2 " << a[1] << endl;
```

```
        cin >> j;
    }

    if( j == 1 ) {
        cout<< "There are not data for collider yet\n";
        exit(1);
        }
        else z[0] = j;


    cout << "Choose type of reaction " << endl << "1 " << b[0] << endl <<"2 "
<< b[1] << endl << "3 " << b[2] << endl << "4 " << b[3] << endl;
    cin >> j;

        while( j < 1 || j > 4) {
          cout << "You entered wrong data. You can choose 1,2,3 or 4. Try
again. " << endl;
        cout << "Choose type of reaction " << endl << "1 " << b[0] << endl
<<"2 " << b[1] << endl << "3 " << b[2] << endl << "4 " << b[3] << endl;
        cin >> j;
    }

     z[ 1 ] = j ;


    if ( z[1] == 2 || z[1] == 3 || z[1] == 4 ) {
        cout<< "There are not data  for " << b[1] << " yet\n";
        exit(1);
     }

      cout << "Choose energy " << endl << "1 " << c[0] << endl << "2 " <<
c[1] << endl << "3 " << c[2] << endl << "4 " << c[3] <<endl;
      cin >> j;


        while(j < 1 || j > 4) {
          cout << "You entered wrong data. You can choose 1,2,3 or 4. Try
again. " << endl;
        cout << "Choose energy " <<  endl << "1 " << c[0] << endl << "2 " <<
c[1] << endl << "3 " << c[2] << endl << "4 " << c[3] <<endl;
        cin >> j;
    }
        z[2] = j;

    cout << "Choose transverse momentum" << endl << "1 " << d[0] << endl <<
"2 " << d[1] << endl << "3 " << d[2] << endl << "4 " << d[3]
    <<endl << "5 " << d[4] << endl;
       cin >> j;

        while(j < 1 || j > 5) {
          cout << "You entered wrong data. You can choose 1,2,3,4 or 5. Try
again. " << endl;
        cout << "Choose transverse momentum " <<  endl << "1 " << d[0] <<
endl << "2 " << d[1] << endl << "3 " << d[2] << endl << "4 " << d[3]
    <<endl << "5 " << d[4] << endl;
        cin >> j;
    }
        z[3] = j;

    if (z[3] == 1) {
    if( z[2] == 2 )
        ReadData("Experiment_for_ppnX_ISR_30GeV_pt0");

    if( z[2] == 4 )
```

```
        ReadData("Experiment_for_ppnX_ISR_53GeV_pt0");

        cout << "There are no data for transverse momentum equal to 0 " <<
endl;
        exit(1);
        }


     Char_t* f[4][4];

     f[0][0] = "Experiment_for_ppnX_ISR_22GeV_pt02";
     f[1][0] = "Experiment_for_ppnX_ISR_30GeV_pt02";
     f[2][0] = "Experiment_for_ppnX_ISR_44GeV_pt02";
     f[3][0] = "Experiment_for_ppnX_ISR_53GeV_pt02";
     f[1][1] = "Experiment_for_ppnX_ISR_30GeV_pt04";
     f[2][1] = "Experiment_for_ppnX_ISR_44GeV_pt04";
     f[3][1] = "Experiment_for_ppnX_ISR_53GeV_pt04";
     f[1][2] = "Experiment_for_ppnX_ISR_30GeV_pt06";
     f[2][2] = "Experiment_for_ppnX_ISR_44GeV_pt06";
     f[3][2] = "Experiment_for_ppnX_ISR_53GeV_pt06";
     f[1][3] = "Experiment_for_ppnX_ISR_30GeV_pt08";
     f[2][3] = "Experiment_for_ppnX_ISR_44GeV_pt08";
     f[3][3] = "Experiment_for_ppnX_ISR_53GeV_pt08";

         if(z[3] > 2 && z[2] == 1 ) {
        cout << "There are no data for energy equal to 23 " << endl;
        exit(1);
        }

     ReadData(f[z[2]-1][z[3]-2]);

}

//read data from ExpData.root

 void ReadData(Char_t* branchName) {

     ExpData_t Exp1Data;
    //cout<<branchName<<endl;
    // Opne ROOT file
    TFile* f = new TFile(gDir + "ExpData.root");

     TTree* tree = (TTree*)f->Get("T");
    Int_t num;
    //cout << Form("%s_Num", branchName);
    TBranch* b = tree->GetBranch(Form("%s_Num", branchName));
    b->SetAddress(&num);
    b->GetEntry(0);
    b = tree->GetBranch(branchName);
    b->SetAddress(&Exp1Data);
    //cout << num << endl;
    /*for(Int_t i = 0;i<num;++i)
    {
        b->GetEntry(i);
        //cout << i <<": " << Exp1Data.x << " " << Exp1Data.pt << " " <<
Exp1Data.f << " " << Exp1Data.deltaf << endl;
    }*/

    Double_t* x1 = new Double_t[num];
    Double_t* f1 = new Double_t[num];
    Double_t* deltax1 = new Double_t[num];
    Double_t* deltaf1 = new Double_t[num];
```

```
    for(Int_t i = 0;i < num;++i)
    {
        b->GetEntry(i);
        x1[i] = Exp1Data.x ;
        //cout<<x1[i]<<endl;
        f1[i] = Exp1Data.f;
        //cout<<f1[i]<<endl;
        deltaf1[i] = Exp1Data.deltaf;
        //cout<<deltaf1[i]<<endl;
        deltax1[i]=0;
        //cout<< deltax1[i] << endl;
        }


    TCanvas *c = new TCanvas("c", "c",14,45,700,500);
    TGraphErrors* graph1 = new TGraphErrors(num,x1,f1,deltax1,deltaf1);
    graph1->Draw("A*");
    c->SetLogy();

    }
void DrawMyTGraph() {

Char_t* branchName = "Experiment_for_ppnX_ISR_53GeV_pt04";
    TFile* f = new TFile(gDir + "ExpData.root");
     TTree* tree = (TTree*)f->Get("T");

    Int_t numEntries;
    //TBranch* b = tree->GetBranch("Experiment_for_ppnX_ISR_53GeV_pt0_Num");
    TBranch* b = tree->GetBranch(Form("%s_Num", branchName));
    b->SetAddress(&numEntries);
    b->GetEntry(0);

    ExpData_t Exp1Data;
    b = tree->GetBranch(branchName);
    b->SetAddress(&Exp1Data);

    Double_t* x1 = new Double_t[numEntries];
    Double_t* f1 = new Double_t[numEntries];
    Double_t* deltax1 = new Double_t[numEntries];
    Double_t* deltaf1 = new Double_t[numEntries];

    Int_t i;

    for(i = 0;i<numEntries;++i)
    {
        b->GetEntry(i);
        x1[i] = Exp1Data.x ;
        //cout<<x1[i]<<endl;
        f1[i] = Exp1Data.f;
        //cout<<f1[i]<<endl;
        deltaf1[i] = Exp1Data.deltaf;
        //cout<<deltaf1[i]<<endl;
        deltax1[i]=0;
        //cout<< deltax1[i] << endl;
        }


    TCanvas *c = new TCanvas("c", "c",14,45,700,500);
    TGraphErrors* graph1 = new
TGraphErrors(numEntries,x1,f1,deltax1,deltaf1);

    TF1 *MyFunc2  = new TF1("found", "RHOxfpt04(x)", 0.6, 0.95);
    TGraph* graph2 = new TGraph(MyFunc2);
```

```
    graph1->Draw("*A");
    graph2->Draw("same");
    c->SetLogy();
}

////////////////////////////////////////////////////////////////////////
// RHOxfpt04_53GeV_07.04.2010.C
#include <iostream.h>
#include "TF1.h"
//
//----------Constants--------------------
Double_t PI=3.14159;
Double_t G2=183.4;
Double_t MN=0.94;
Double_t MPI=0.14;
Double_t MN2=MN*MN;
Double_t MPI2=MPI*MPI;
Double_t stotpiplusp=22.;
//
//-------Parameters-----------------------
Double_t T0=-0.7;
Double_t R12=0.3;
Double_t R22=0.74;
//
//----------Initial state-----------------
Double_t U;
U=52.8;
Double_t s;
s=U*U;
Double_t EA;
EA=U/2;
Double_t PA;
PA=sqrt(EA*EA-MN2);
//
//-----Defining function Q(x,y,z)(c.m.s momentum)------------
Double_t Q(Double_t x, Double_t y, Double_t z)
{
Double_t Q=sqrt((x-y-z)*(x-y-z)-4.*y*z)/(2.*sqrt(x));
return (Q);
}
//
//========Kinematics of final state================================
//
//---- Defining function E (Energy of created particle)------------
Double_t E(Double_t xf, Double_t pt)
{
return (sqrt((xf*PA)*(xf*PA)+pt*pt+MN2));
}
//
//---- Defining function t (Square of 4-momentum transfer)---------
Double_t t(Double_t xf, Double_t pt)
{
return ((EA-E(xf,pt))*(EA-E(xf,pt))-(PA-xf*PA)*(PA-xf*PA)-pt*pt);
}
//
//----Defining function s1 (Invariant missing mass)-----------------
Double_t s1(Double_t xf,Double_t pt)
{
return (s+MN2-2.*U*E(xf,pt));
}
//
//---------------Defining c.m.s momenta of virtual pion-------------
Double_t ppivirt(Double_t xf, Double_t pt)
{
```

```
Double_t ppivirt=Q(s1(xf,pt),MN2,MPI2);
return(ppivirt);
}
//=====================================================================

//###############Constructing OPER cross section####################

//_________________Building pion Green function_____________________

//--------Pion signature factor-------------------------------------

Double_t Signfactpi(Double_t xf, Double_t pt)
{
return (PI/2./sin(PI/2.*(t(xf,pt)-MPI2)));
}
//----------------Pion Green finction---------------------------------
//************ t >= T0 ******************************************

Double_t Greenpi(Double_t xf, Double_t pt)
{
if (t(xf,pt) >= T0)
  Double_t Greenpi=exp((R12+log(s/s1(xf,pt)))*(t(xf,pt)-
MPI2))*Signfactpi(xf,pt);
else
//************** t < T0 ***************************************
Double_t Greenpi=exp((R12+log(s/s1(xf,pt)))*(t(xf,pt)-
T0))*(PI/2./sin(PI/2.*(T0-MPI2)))*exp(R22*(t(xf,pt)-T0));
return(Greenpi);
}
//Double_t Greenpi1(Double_t xf, Double_t pt)
//{
//Double_t Greenpi1=exp((R12+log(s/s1(xf,pt)))*(t(xf,pt)-MPI2))
//*Signfactpi(xf,pt);
//return(Greenpi1);
//}
//-------Virtual pion-nucleon scattering-------------------------

//*******Invariant Rho-function for pp->nX as function of xf and pt **
Double_t RHOxfpt(Double_t xf, Double_t pt)
{
Double_t RHOxfpt = 1.0/(pow(2.*PI,3)*PA*U)*G2*(-t(xf,
pt))*Greenpi(xf,pt)*Greenpi(xf,pt)*
ppivirt(xf,pt)*sqrt(s1(xf, pt))*stotpiplusp;

return(RHOxfpt);
}
//--------Invariant Rho-function for pp->nX as function of xf at pt=0.4

Double_t RHOxfpt04(Double_t xf)
{
Double_t RHOxfpt04 = RHOxfpt(xf, 0.4);
return(RHOxfpt04);
}
```

Բովանդակություն