Генераторы в PandaRoot

А.В. Лучинский, С.В. Пославский

Институт физики высоких энергий
Протвино, Россия

24.04.2013

Содержание

1. Новый генератор NewGen
2. $p\bar{p} \rightarrow \chi_c + X$
3. Pythia
4. EvtGen
5. Заключение

Новый генератор *NewGen*

Для того, чтобы добавить новый генератор надо:

1) Создать директорию **pgenerators/NewGen** и класс **NewGen**

NewGen.h

```
class NewGen: public FairGenerator {
        …
        virtual Bool_t ReadEvent(FairPrimaryGenerator* primGen);
        ...
}
```

NewGen.cpp

```
Bool_t PndChiGen::ReadEvent(FairPrimaryGenerator* primGen) {
        …
        primGen->AddTrack(pdgID, px, ,py, pz, x, y, z);
        …
        return kTRUE;
}
```

Новый генератор *NewGen*

Для того, чтобы добавить новый генератор надо:

1) Создать директорию **pgenerators/NewGen** и класс **NewGen**

2) Добавить его в список компиляции

CMakeLists.txt

```
...
set(CHIGEN_SRCS
NewGen.cxx
)
...
```

Новый генератор *NewGen*

Для того, чтобы добавить новый генератор надо:

1) Создать директорию **pgenerators/NewGen** и класс **NewGen**

2) Добавить его в список компилирования

3) Добавить его в библиотеки ROOT

NewGenLinkDef.h

```
...
#pragma link C++ class NewGen+;
...
```

NewGen.cxx

```
...
ClassImp(NewGen);
...
```

NewGen.h

```
...
ClassDef(NewGen,1);
...
```

Новый генератор *NewGen*

Для того, чтобы добавить новый генератор надо:

1) Создать директорию **pgenerators/NewGen** и класс **NewGen**

2) Добавить его в список компилирования

3) Добавить его в библиотеки ROOT

4) Откомпилировать

```
build/$  cmake ../
build/$  make
```

Новый генератор *NewGen*

Для того, чтобы добавить новый генератор надо:

1) Создать директорию **pgenerators/NewGen** и класс **NewGen**

2) Добавить его в список компилирования

3) Добавить его в библиотеки ROOT

4) Откомпилировать

5) Написать макро для ROOT

macro/run_sim_sttcombi_NewGen.C

```
...
FairRunSim *fRun = new FairRunSim();
…
FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
fRun->SetGenerator(primGen);
NewGen* newGen = new PndChiGen();
primGen->AddGenerator(chiGen);
...
```

Новый генератор *NewGen*

Для того, чтобы добавить новый генератор надо:

1) Создать директорию **pgenerators/NewGen** и класс **NewGen**

2) Добавить его в список компилирования

3) Добавить его в библиотеки ROOT

4) Откомпилировать

5) Написать макро для ROOT

6) Далее по обычной схеме (sim → digi → reco → pid → analysis)

7) Profit!

Лучше всего брать за основу уже готовый генератор и модифицировать его...

$$p\bar{p} \rightarrow \chi_c X$$

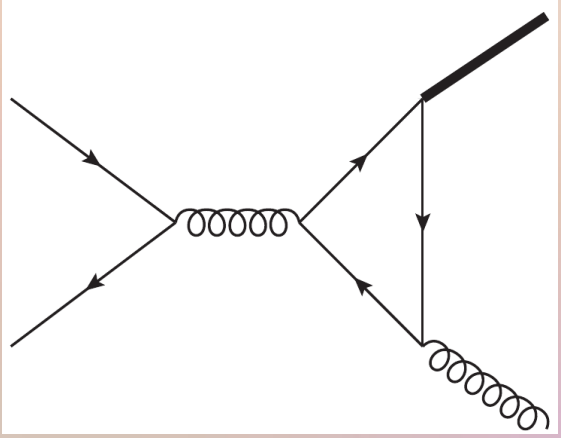
Нашей задачей было написать генератор для рождения и распадов поляризованных χ_c-мезонов.

При низких энергиях доминирует

Кварк-антикварковая аннигиляция

[A.V. Luchinsky, S.V. Poslavsky,

PR D85 (2012) 074016]



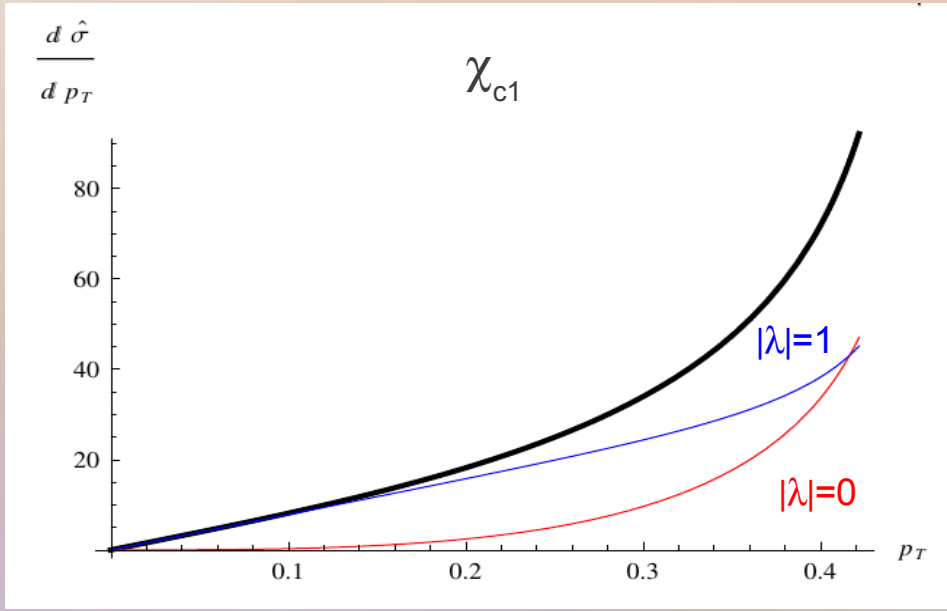
Для этого необходимо:

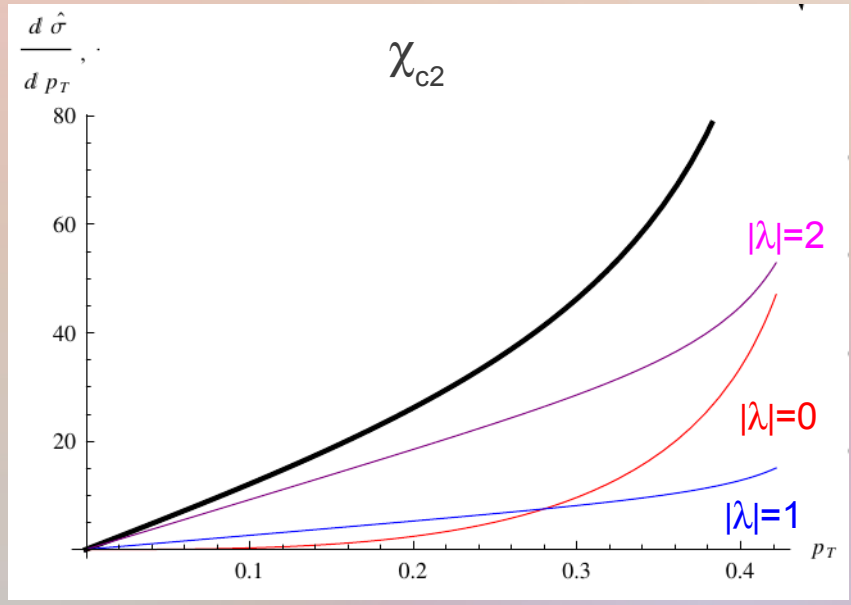
1) Их родить: **Pythia6, Pythia8**

2) Развалить: **EvtGen**

Pythia 6,8

Необходимо создать новые частицы ($\chi_{clL,T}$ и т. д.), которые рождаются в соответствии с известными моделями.





Pythia 6,8

К сожалению, Pythia была написана для высоких энергий, маленькие по умолчанию не принимает

Pythia6

```
ECM=5.5D0
PARP(2)=5D0
CALL PYINIT('USER',' ',' ',0D0)
```

Pythia8

Параметры читаются из служебных файлов, которые можно указать при инициализации

```
...
<parm name="Beams:eCM" default="14000." min="1.">
...
```

Pythia 6,8

В терминах цветовых потоков нельзя написать процессы $0_c \rightarrow 3_c\ 3_c\ \bar{3}_c$ ($p \rightarrow uud$), поэтому в конечном состоянии всегда остаются дикварки $ud(\bar{3}_c)$, а значит и барионы. Но в 50% случаев они должны аннигилировать.

Написан генератор адронных пучков, который силой валит дикварки.

Барионы пропали :)

```
-------- PYTHIA Event Listing  (complete event)  -----

    no      id   name          ...        m
     0      90   (system)      ...     5.500
     1    9998   c1L           ...     3.510
     2      21   (g)           ...     0.000
     3       2   (u)           ...     0.000   }
     4       1   (d)           ...     0.000   }  (ud)
     5      -2   (ubar)        ...     0.000   }
     6      -1   (dbar)        ...     0.000   }  (ūd̄)
...
    14     211   pi+           ....    0.140
    15    -211   pi-           ....    0.140
...
          Charge sum:  0.000 ...    5.500

--------  End PYTHIA Event Listing  ----
```

EvtGen

Родившиеся поляризованные частицы необходимо развалить ($\chi_c \rightarrow J/\psi\gamma$) с учетом поляризаций. Это проще всего сделать с помощью EvtGen, для чего

1) Добавить в их таблицу частиц

evt.pdl

```
..
add  p Meson      c2L   445  3.55618   0.00208 0.0060  0      4      0 214
...
```

2) Написать модель

EvtC2L.h

```
class EvtC2L:public  EvtDecayAmp  {
…
  void decay(EvtParticle *p);
…
}
```

3) Написать файл распада

Заключение