

# Моделирование физических процессов в пакете Geant4

Алексей Жемчугов  
ОИЯИ  
E-mail: [zhemchugov@jinr.ru](mailto:zhemchugov@jinr.ru)

Иркутский государственный университет

28-29 апреля 2015

# Содержание

- Лекция 1

Цели моделирования. Метод Монте-Карло. Доступные инструменты. Как проходит моделирование эксперимента от космической частицы до отклика детектора

- Лекция 2

Как устроен пакет Geant4. Самая простая программа моделирования. Как запустить ее на сервере ИГУ?

- Лекция 3

Более сложная программа моделирования. Geant4 и ROOT.

Лекции и примеры программ доступны на <http://geant4.jinr.ru>

# Что такое моделирование?

Имитация физических процессов и отклика детекторов в проводимом или планируемом эксперименте

- Вопрос:

*Если мы умеем моделировать на компьютере все физические процессы в установке, можем ли мы совсем обойтись без проведения физического эксперимента?*

- Еще вопрос:

*Если у нас уже есть результаты реальных измерений, зачем их моделировать?*

# Что такое моделирование?

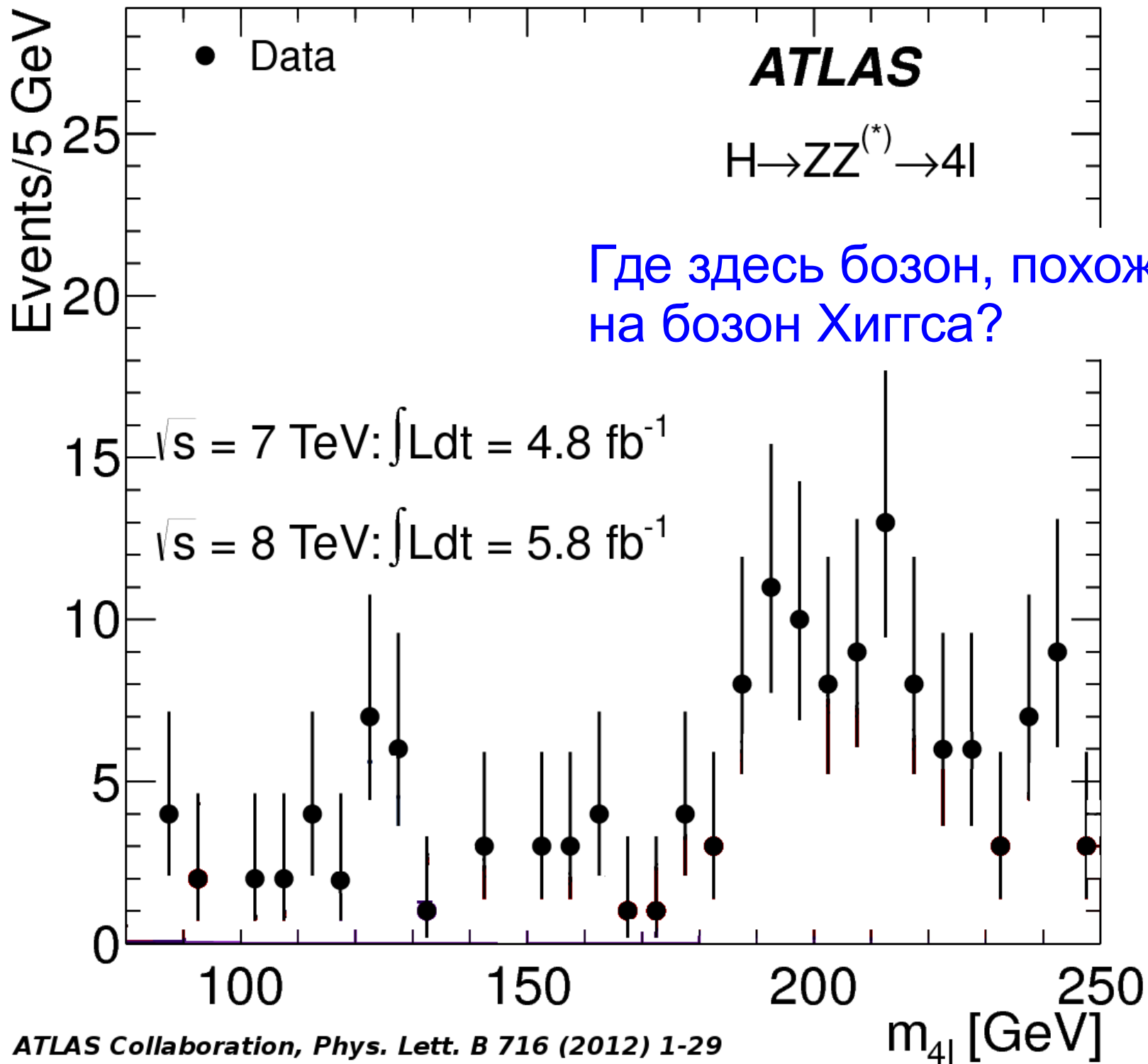
Имитация физических процессов и отклика детекторов в проводимом или планируемом эксперименте

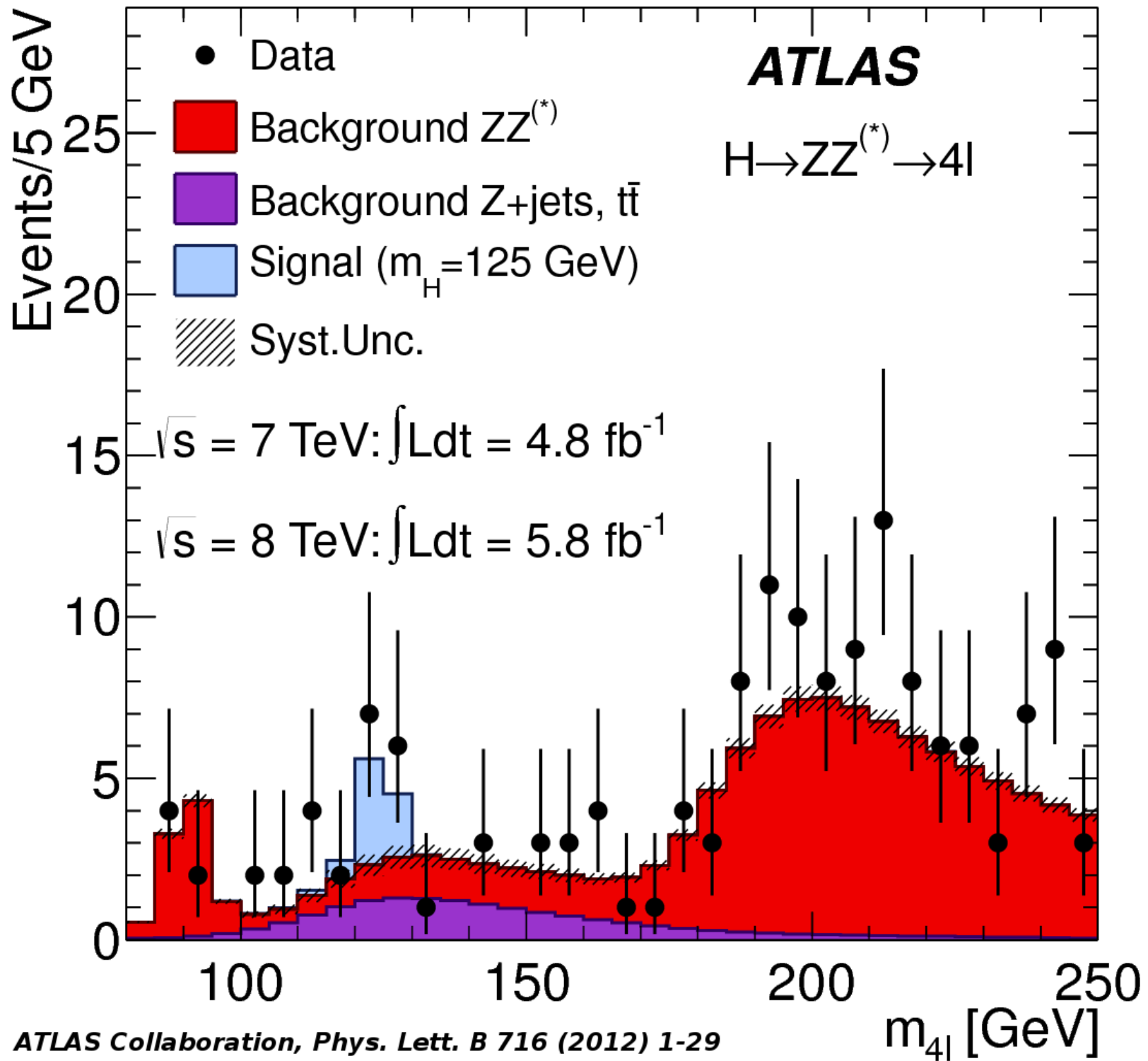
- Вопрос:

*Если мы умеем моделировать на компьютере все физические процессы в установке, можем ли мы совсем обойтись без проведения физического эксперимента?*

- Еще вопрос:

*Если у нас уже есть результаты реальных измерений, зачем их моделировать?*





# Цели моделирования

- При планировании эксперимента
  - Оптимизация конструкции детектора
  - Отладка алгоритмов реконструкции событий
  - Расчет ожидаемых значений сигнала и фоновых процессов.  
Оценка ожидаемой точности измерений
- При анализе данных
  - оптимизация процедуры анализа данных
  - определение аксептанса установки
  - определение вклада фоновых процессов
  - оценка систематических погрешностей
  - сравнение результатов анализа с теоретическими предсказаниями

# Метод Монте-Карло

*Метод Монте-Карло – это численный метод решения прикладных математических задач при помощи моделирования случайных величин и статистической оценки их характеристик.*





# JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

*Number 247*

SEPTEMBER 1949

*Volume 44*

## THE MONTE CARLO METHOD

NICHOLAS METROPOLIS AND S. ULAM

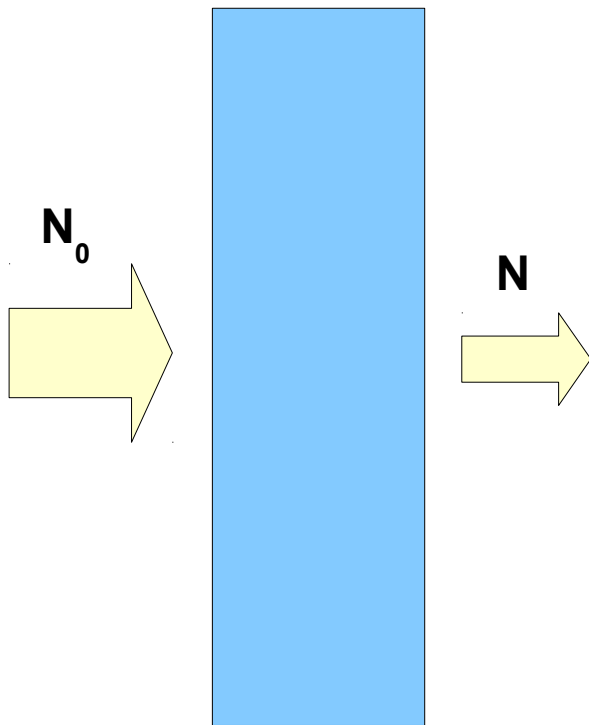
*Los Alamos Laboratory*

We shall present here the motivation and a general description of a method dealing with a class of problems in mathematical physics. The method is, essentially, a statistical approach to the study of differential equations, or more generally, of integro-differential equations that occur in various branches of the natural sciences.

Метод появился при работе над Манхэттенским проектом:

- S.M. Ulam, J. von Neumann, “On combination of stochastic and deterministic processes”. *Bull. Amer. Math. Soc.* 53 1120 (1947)
- S.M. Ulam, N. Metropolis, “The Monte-Carlo method”, *J. Amer. Statist. Assoc.* 1949 , 44 Vol 247, 335-341

- Математический смысл: эффективный способ вычисления многомерных интегралов со сложными пределами интегрирования
- Пример:



**Поглощение излучения в веществе:**

$$dN = -\mu N dx$$

$$N = N_0 \exp(-\mu x)$$

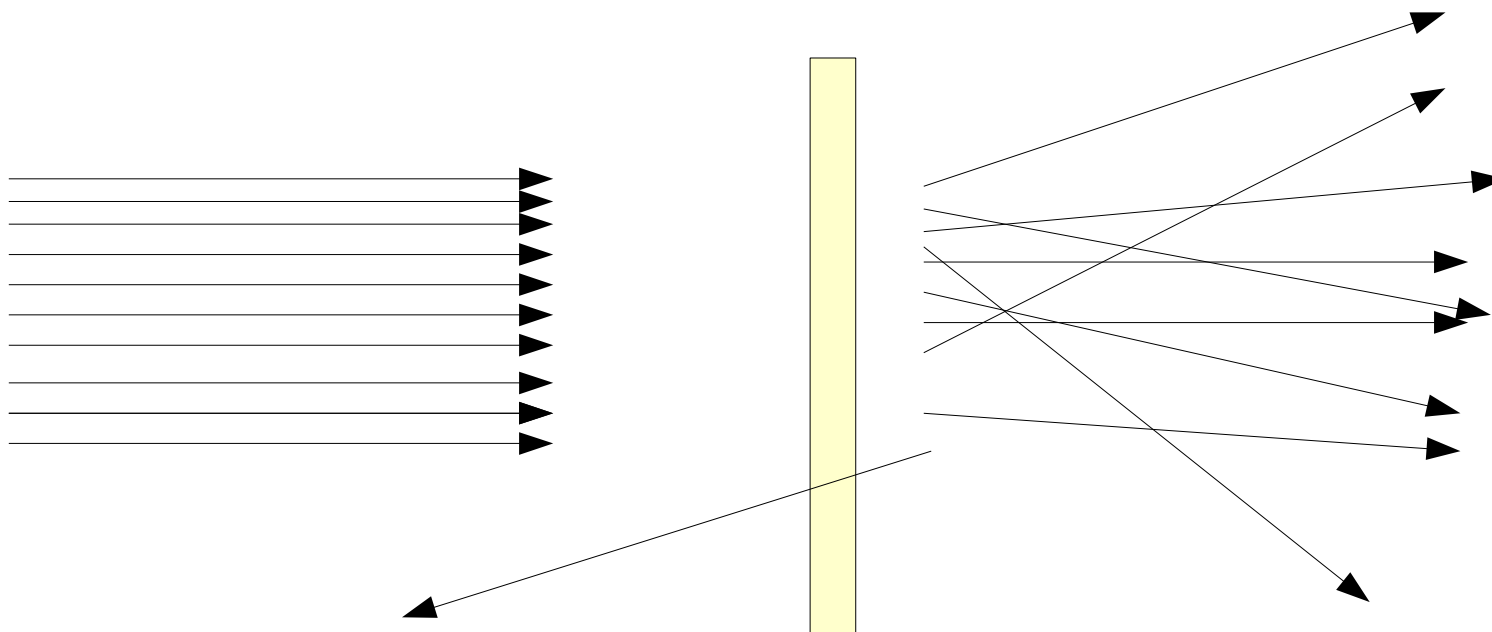
Если  $\mu$  зависит от  $(x, y, z, E)$       ??

Если  $N$  зависит от  $(E, \bar{r})$       ???

Если учесть рассеяние      ?????

Если  $\mu$  нельзя задать аналитически      ??????

# Пример: опыт Резерфорда



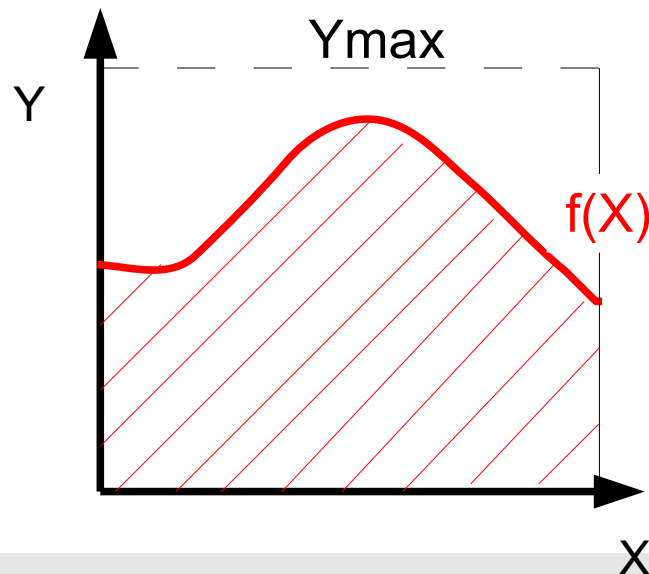
$$\frac{d\sigma}{d\Omega} = \left( \frac{Z_1 Z_2 e^2}{2mv^2} \right)^2 \frac{1}{\sin^4 \frac{\Theta}{2}} \sim \text{вероятность рассеяния на угол } \Theta$$

## Шаги моделирования:

1. Генерируется пучок из N частиц
2. Для каждой частицы вычисляется случайный угол рассеяния  $\Theta$  в соответствии с известным распределением и заданной скоростью частиц
3. Для каждой частицы вычисляется случайный азимутальный угол  $\varphi$ , который разыгрывается равномерно в интервале  $[0, 2\pi]$
4. Рассеяние частиц смоделировано

# Как смоделировать произвольное распределение?

- Равномерные распределения случайных чисел получаются при помощи программ - генераторов псевдослучайных чисел (алгоритмы L'Ecuyer, Marsaglia-Zaman, Mersenne Twister и др.)
- Метод выбраковки ("Hit-and-miss") - наиболее простой способ моделирования произвольного распределения  $f(x)$  при помощи равномерно распределенных случайных чисел



1. Выбирается случайное  $X$  в интервале  $[0, X_{\max}]$
2. Выбирается случайное  $Y$  в интервале  $[0, Y_{\max}]$
3.  $X$  принимается при условии  $Y < f(X)$

# Литература

И.М.Соболь Численные методы Монте-Карло. Наука, М. 1973

Е.Бюклинг, К.Каянти Кинематика элементарных частиц, Мир, М. 1975

С.М.Ермаков, Метод Монте-Карло и смежные вопросы, Наука, М., 1975

Г.И.Копылов, Основы кинематики резонансов, Наука, М., 1970

В электронном виде доступны на сайте <http://geant4.jinr.ru>  
(Библиотека)

# Программы для моделирования

Существует большое количество уже написанных программ и программных пакетов для моделирования физических процессов с участием элементарных частиц с помощью метода Монте-Карло:

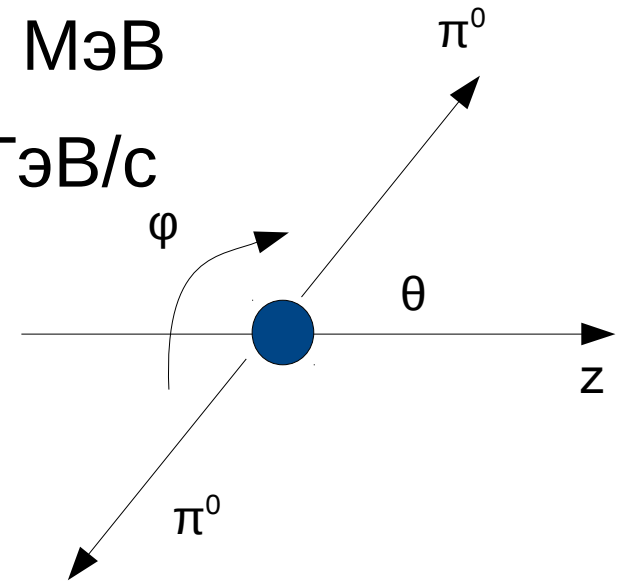
- Генераторы событий
- Специализированные программы
- Универсальные программы

# Генераторы событий

- Позволяют моделировать наблюдаемые величины (импульс, энергия, точка рождения ...) на основе теоретических предсказаний.
- Искажение наблюдаемых величин, связанное с прохождением частиц через вещество, не учитывается
- Генератор событий - это «мост» между теоретиками и экспериментаторами.
- Генераторов существует великое множество. Только на LHC применяются несколько десятков генераторов. Самый известный, пожалуй, PYTHIA.
- Давайте напишем свой генератор?

# Простой генератор: распад каона $K^0 \rightarrow 2\pi^0$

- Масса каона 498 МэВ, масса пиона 135 МэВ
- Для определенности, импульс каона 5 ГэВ/с
- Перейдем в СЦМ:
  - $E_\pi = M_K/2$
  - $P_\pi = \text{sqrt}(E_\pi^2 - M_\pi^2)$
- Разыграем направление одного из пионов (изотропно)
  - $\cos(\theta) = \text{RNDM}[-1,1]$
  - $\varphi = \text{RNDM} [0, 2\pi]$
- Зададим направление второго пиона противоположно первому
- Перейдем в лабораторную систему при помощи преобразования Лоренца





# Как это выглядит на C++

```
{  
  
double mkaon = 498; // MeV  
double mpion = 135; // MeV  
double pkaon = 5000; // MeV/c  
double beta = pkaon/sqrt(pkaon*pkaon+mkaon*mkaon);  
  
int i;  
for (i=0; i < 100000 ; i++) // 100000 events  
{  
    // calculate kinematics  
    double epion = mkaon/2.;  
    double ppion = sqrt(epion*epion - mpion*mpion);  
    double costheta = 2*gRandom->Rndm() - 1; // [-1,1]  
    double sintheta = sqrt(1-costheta*costheta);  
    double phi = gRandom->Rndm()*2*3.14159; // [0,2pi]  
  
    // calculate momentum of the 1st pion  
    double px1 = ppion*sintheta*cos(phi);  
    double py1 = ppion*sintheta*sin(phi);  
    double pz1 = ppion*costheta;  
  
    // calculate momentum of the 2nd pion  
    double px2 = -px1;  
    double py2 = -py1;  
    double pz2 = -pz1;  
  
    // make 4-vectors  
    TLorentzVector pion1 (px1, py1, pz1, epion);  
    TLorentzVector pion2 (px2, py2, pz2, epion);  
    // Lorentz boost  
    pion1.Boost(0., 0., beta);  
    pion2.Boost(0., 0., beta);  
}  
}
```

Пример kaon.C на <http://geant4.jinr.ru>

# Добавим сохранение в дерево ROOT

```
{  
double mkaon = 498; // MeV  
double mpion = 135; // MeV  
double pkaon = 5000; // MeV/c  
double beta = pkaon/sqrt(pkaon*pkaon+mkaon*mkaon);  
int i;  
for (i=0; i < 100000 ; i++) // 100000 events  
{  
    // calculate kinematics  
    double epion = mkaon/2.;  
    double ppion = sqrt(epion*epion - mpion*mpion);  
    double costheta = 2*gRandom->Rndm() - 1; // [-1,1]  
    double sintheta = sqrt(1-costheta*costheta);  
    double phi = gRandom->Rndm()*2*3.14159; // [0,2pi]  
  
    // calculate momentum of the 1st pion  
    double px1 = ppion*sintheta*cos(phi);  
    double py1 = ppion*sintheta*sin(phi);  
    double pz1 = ppion*costheta;  
  
    // calculate momentum of the 2nd pion  
    double px2 = -px1;  
    double py2 = -py1;  
    double pz2 = -pz1;  
  
    // make 4-vectors  
    TLorentzVector pion1 (px1, py1, pz1, epion);  
    TLorentzVector pion2 (px2, py2, pz2, epion);  
    // Lorentz boost  
    pion1.Boost(0., 0., beta);  
    pion2.Boost(0., 0., beta);  
}  
}
```

```
// open ROOT file  
TFile fout("kaon.root","RECREATE");  
TTree* tree = new TTree("kaon","kaon");  
  
// initialize ROOT tree  
double p1[4], p2[4];  
tree->Branch("p1",p1,"p1[4]/D");  
tree->Branch("p2",p2,"p2[4]/D");
```

```
// Fill the tree  
p1[0] = pion1.Px(); p2[0] = pion2.Px();  
p1[1] = pion1.Py(); p2[1] = pion2.Py();  
p1[2] = pion1.Pz(); p2[2] = pion2.Pz();  
p1[3] = pion1.E(); p2[3] = pion2.E();  
tree->Fill();
```

```
// Save file  
tree->Write();  
fout.Close();
```

## Задание на дом

- Смоделировать влияние импульсного разрешения детектора:  
в лабораторной системе для каждого пиона  
*импульс = gRandom->Gaus(импульс, разрешение)*
- Проверить как изменятся распределения импульса пионов и инвариантной массы системы двух пионов
- **Повышенной сложности:** Написать программу - генератор распада  $K^0 \rightarrow 3\pi^0$

# Специализированные программы

- Оптимизированы под конкретную задачу
- Узкий набор физических моделей
- Как правило, очень ограниченные возможности в описании геометрии установки
- Примеры:
  - Моделирование ШАЛ: CORSIKA, AIRES
  - Моделирование пробега частиц низкой энергии в веществе (ионная имплантация и т.д.): SRIM, MARLOWE
  - Моделирование защиты: MARS, SHIELD, PHITS
  - Моделирование задач радиационной медицины: EGSnrc, BEAMnrc, PEREGRINE

Есть отдельный набор программ для расчета реакторов

# Моделирование ШАЛ

- Количество частиц в ливне может достигать  $10^{10}$ - $10^{15}$
- Развитие ливня зависит от свойств атмосферы, магнитного поля Земли и т.д.
- Требуются модели физических взаимодействий ориентированные на сверхвысокие энергии
- Требуется смоделировать развитие ЭМ и адронного каскадов частиц, черенковское излучение
- Расчет с помощью универсальных программ требует огромных затрат времени ЦПУ

# CORSIKA, AIRES

- Специализированные программы для моделирования ШАЛ
  - оптимизация скорости расчетов за счет введения статистических весов ( моделирование 1000 частиц с близкими свойствами заменяется моделированием одной частицы с весом 1000)
  - встроено описание атмосферы и магнитного поля Земли
  - большой набор адронных моделей для сверхвысоких энергий (DPMJET, SYBILL, QGS). **Непонятно, как проверить их правильность? Одно из решений - посчитать с разными моделями и сравнить ответ.**
  - в результате работы программ получается список частиц на поверхности детектора (без моделирования его отклика)
- CORSIKA <https://web.ikp.kit.edu/corsika/>
- AIRES <http://www2.fisica.unlp.edu.ar/auger/aires/>

# Универсальные программы

- Широкий набор физических моделей для всех диапазонов энергий
- Богатый инструментарий для описания геометрии установки
- Хорошо подходят для моделирования процессов в детекторе и его отклика
- Встроенные генераторы событий довольно простые, поэтому универсальные пакеты обычно применяют в связке с внешними программами-генераторами
- Примеры: MCNP, FLUKA, Geant3, Geant4
  - MCNP, FLUKA - программы, требующие составления конфигурационного файла
  - Geant3, Geant4 - пакеты-конструкторы, предоставляющие набор библиотек и требующие написания своей программы

# FLUKA (FLUktuierende KAskade)

- Первоначально разрабатывалась для расчета защиты в проекте SPS в ЦЕРН (1962-1978)
- В настоящее время универсальная программа расчета взаимодействия частиц с веществом
- Написана на языке Фортран.
- Хорошее моделирование адронных ливней
- Сложность описания геометрии
- Лицензионные ограничения



## MCNP (Monte-Carlo N-particle transport code)

- Разработана в Лос-Аламосе для расчета реакторов
- В настоящее время универсальная программа расчета взаимодействия частиц с веществом
- Написана на языке Фортран.
- Хорошее моделирование нейтронных процессов, и процессов при низких энергиях
- Лицензионные и экспортные ограничения
- После выхода версии 4 разделилась на две ветви: MCNP5 и MCNPX (MCNP+LANE)

# GEANT3 ( GEometry ANd Tracking )

- Первая версия появилась в 1974 году в ЦЕРН
- Описание физических процессов основано на программах EGS (э/м ливни) и GHEISHA (адронные ливни)
- Пакет GEANT3 появился в 1982 году и был использован для моделирования детекторов в экспериментах на LEP
- Написан на языке Фортран
- Основной инструмент моделирования в физике частиц на протяжении 30 лет

# GEANT4

- Объектно-ориентированная программа с функциональностью GEANT3
- Первая версия пакета появилась в 1995 году
- Написана на языке C++
- Первое “боевое” применение – эксперимент ВаВаг
- С 2004 года – основная программа моделирования в экспериментах на LHC (кроме ALICE)
- Широкое и растущее применение в физике частиц, космонавтике (ESA), радиационной медицине.

# Что установлено на сервере ИГУ?

```
hep@S5500BC:~$ ls /opt/hep/
```

aires - AIRES версия 2.8.4a

corsika - CORSIKA версия 7.4005

geant4 - Geant4 версия 4.10.01.p01

root - ROOT версия 6.02.05

setup.sh - скрипт задания переменных окружения:

```
hep@S5500BC:~$ source /opt/hep/setup.sh
```

# Пакет программ GEANT4

## **Geant4 – это набор инструментов**

- GEANT4 представляет собой набор программ для моделирования прохождения частиц через вещество
- Включает в себя инструменты для гибкого описания геометрии
- Содержит множество физических моделей взаимодействия частиц с веществом
  - Электромагнитные процессы
  - Адронные процессы
  - Фотон-адронные и лептон-адронные процессы
  - Процессы с участием оптических фотонов
  - Моделирование распадов
  - Параметризация ливней
  - Методы использования статистических весов
- **Не учитывает гравитацию, акустические сигналы, взаимодействие нейтрино**

# Системные требования

## Основные платформы и компиляторы

- Linux + gcc
- Windows + VisualC++
- MacOSX + gcc

## Внешние пакеты

- CLHEP (начиная с версии 4.9.5 встроен в Geant4)
- cmake
- Xerces-C (опционально, при использовании GDML)
- графические библиотеки OpenGL, Qt и другие (опционально, но желательно)

**Требует около 1.2 Гб дискового пространства (полная установка)**

**Требует минимум 128 Мб оперативной памяти для работы**

# Документация и исходный код

<http://cern.ch/geant4>

## Основные публикации

*S. Agostinelli et al.,*

*Geant4: a simulation toolkit, NIM A 506 (2003) 250-303*

*J. Allison et al.,*

*Geant4 developments and applications*

*IEEE Trans. Nucl. Sci. 53 No. 1 (2006) 270-278*



# Основные типы переменных

- Для достижения переносимости кода в Geant4 переопределены основные типы переменных

***G4int, G4long, G4float, G4double,***

***G4bool, G4complex, G4String***

- Лучше использовать переопределенные типы

## Ввод и вывод в коде Geant4

Можно использовать обычные `printf()` и `cout`

Однако более правильно использовать переопределенные потоки Geant4:

```
G4cout << "test" << G4endl;
```

```
G4cerr << "error" << G4endl;
```

При этом будет корректно обрабатываться ввод-вывод в случае, если интерфейс пользователя отличен от командной строки

# Библиотека CLHEP

## Class Library for High Energy Physics

- Содержит описание стандартных математических объектов, часто используемых в ФВЭ
  - 3-векторы и 4-векторы
  - действия с матрицами
  - геометрические объекты и преобразования
  - генераторы случайных чисел
  - система единиц и основные физические константы
- Широко используется в коде Geant4
- Подробное описание

<http://cern.ch/clhep>

- **G4ThreeVector**

- трехкомпонентный  $(x, y, z)$  вектор и действия с ним

- **G4LorentzVector**

- четырехкомпонентный  $(x, y, z, t)$  вектор

- **G4RotationMatrix**

- матрица  $3 \times 3$ , определяющая вращение 3-вектора

- **G4LorentzRotation**

- матрица  $4 \times 4$ , определяющая вращение 4-вектора

- Геометрические объекты и преобразования

- **G4Plane3D, G4Transform3D, G4Normal3D, G4Point3D, G4Vector3D**

# Система единиц Geant4

- Любое число, имеющее размерность, должно быть умножено на соответствующую единицу для перевода во внутреннюю систему единиц Geant4

$$\mathit{length} = 10.0 * \mathit{cm};$$

$$\mathit{kinetic\_energy} = 5.0 * \mathit{GeV};$$

- Для получения величины в желаемых единицах следует делить число на единицу

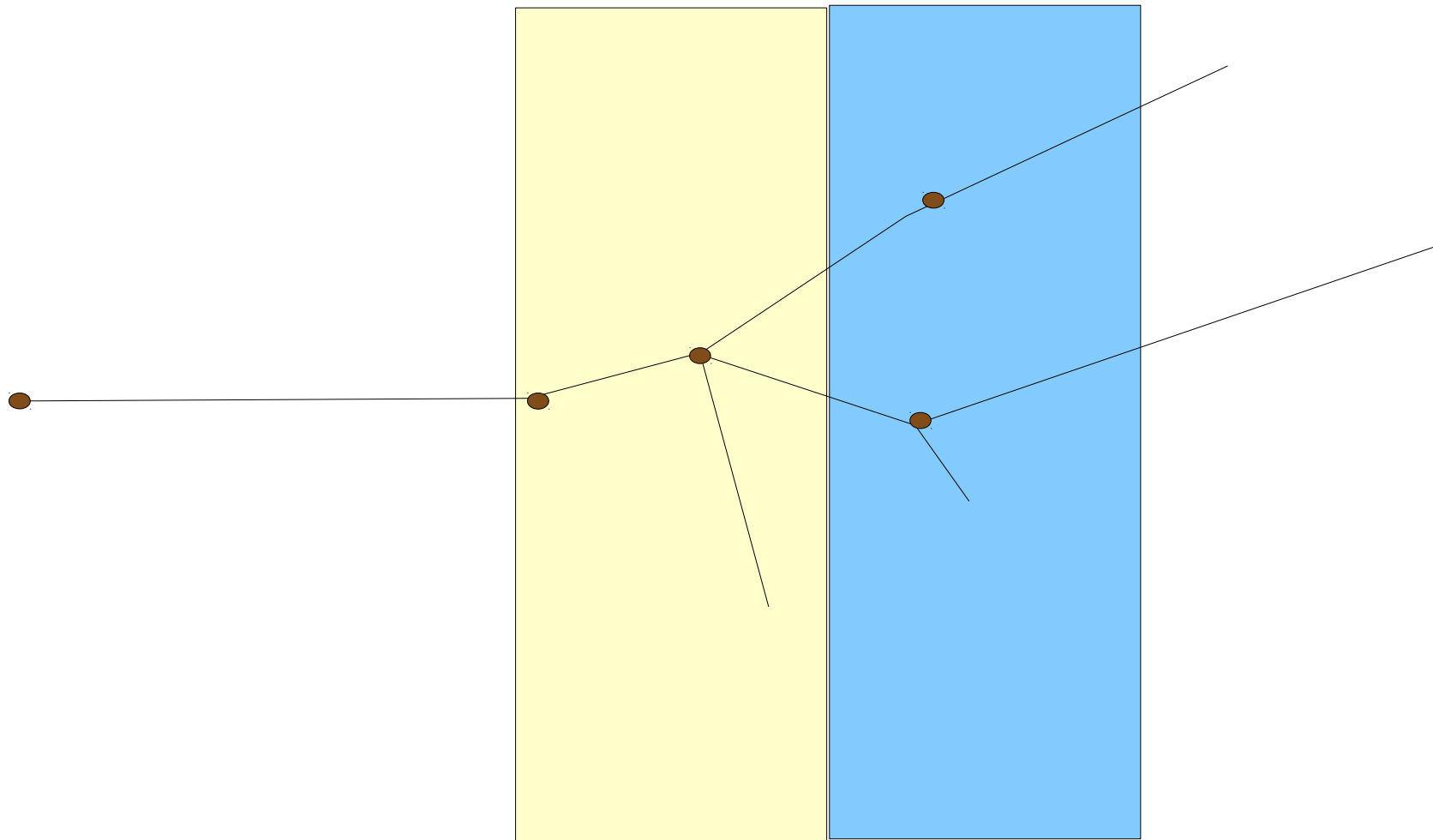
$$\mathit{G4cout} \ll \mathit{eDep} / \mathit{MeV} \ll "[\mathit{MeV}]" \ll \mathit{G4endl};$$

- Все общеупотребительные единицы описаны в Geant4 (CLHEP). При необходимости можно определить свои единицы

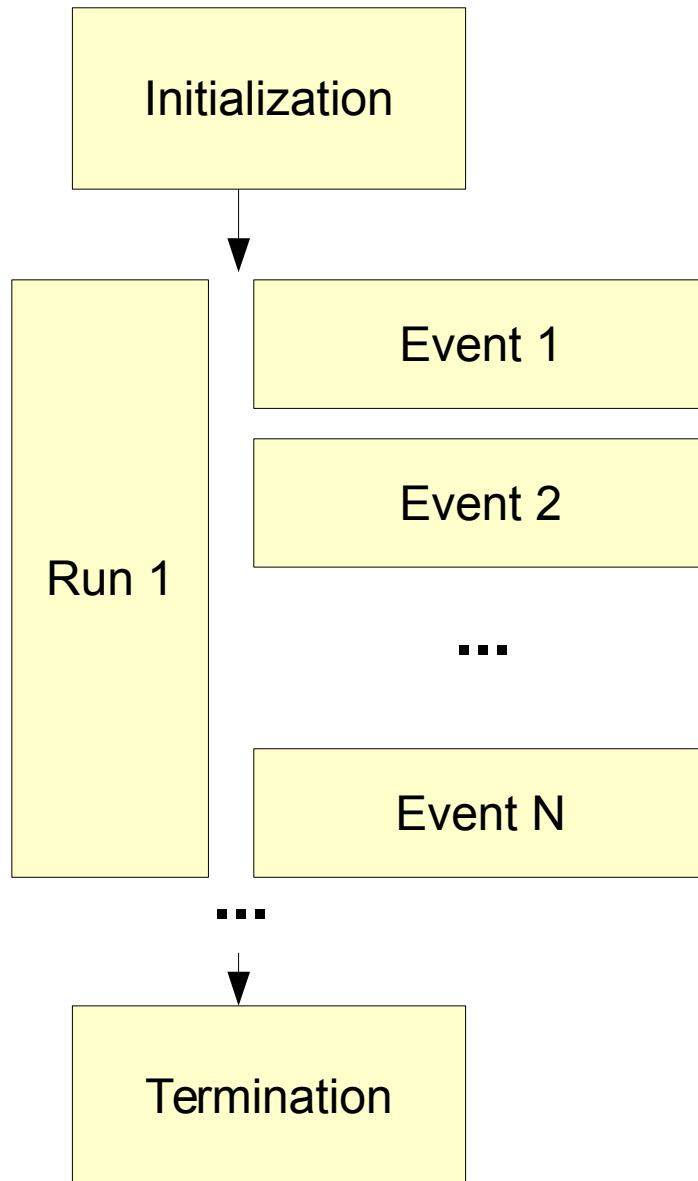
# Трекинг

- Событие (event) - совокупность всех частиц, включая образованные при одном запуске генератора первичной вершины и рожденные ими вторичные частицы
- Каждая частица моделируется до одного из четырех исходов:
  - исчезновение в результате взаимодействия (распад, неупругое рассеяние)
  - достижение самой внешней границы моделируемого объема
  - уменьшение кинетической энергии до нуля
  - остановка по требованию пользователя
- Каждая частица перемещается по геометрии детектора шагами (step). Совокупность шагов от рождения до остановки частицы образует трек.

# Трекинг



# Цикл моделирования



## Необходимо описать:

- Распределение вещества в детекторе и поля
- Генератор события
- Список физических процессов, учитываемых в моделировании

## Кроме того, по желанию:

- Чувствительные элементы и способы моделирования отклика
- Способы визуализации
- Графический интерфейс
- Сохранение результатов
- Пользовательские расширения