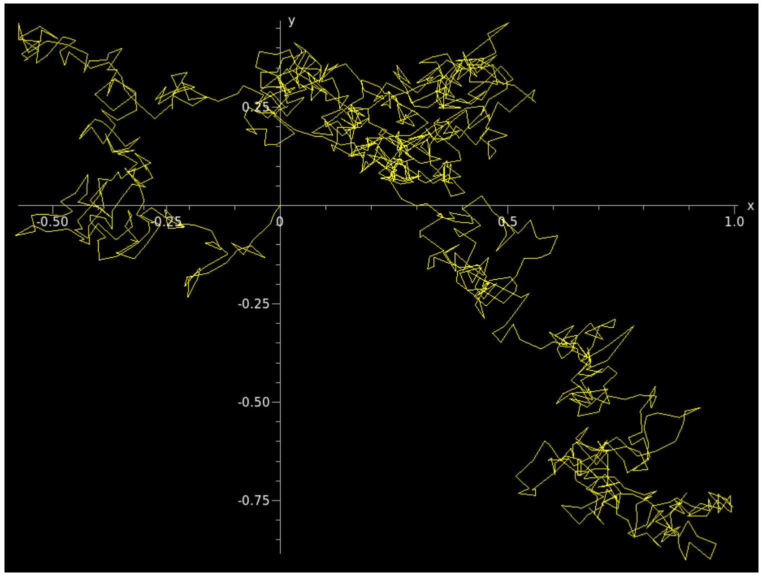
蒙卡方法



李强　北京大学物理学院中楼411

qliphy0@pku.edu.cn，　15210033542

http://www.phy.pku.edu.cn/~qiangli/CP2017.html

蒙卡方法初步

蒙卡方法介绍：

粒子衰变　盒中粒子

蒙卡积分：

中心极限定理，Importance Sampling ,
Vegas

MC模拟：

随机行走
Markov链　Metropolis

高能物理事例产生子：

权重，Unweight，　直方图，Tree，bin，　error
拟合，Chi-2，Likelihood

引言: 粒子衰变

在很多物理问题中，可以直接模拟物理过程，而不用写下比如描述系统的微分方程。唯一的要求是需要知道几率分布函数。

假设在初始时刻t=0，我们有N(0)个X核子， 该种核子会辐射衰变，单位时间衰变几率为ω，则有

$$dN(t) = -\omega N(t)dt,$$

解为 $N(t) = N(0)e^{-\omega t},$ 可定义平均寿命为 $\tau = \frac{1}{\omega}.$

假设 X核子辐射衰变到Y，Y也会衰变，则有

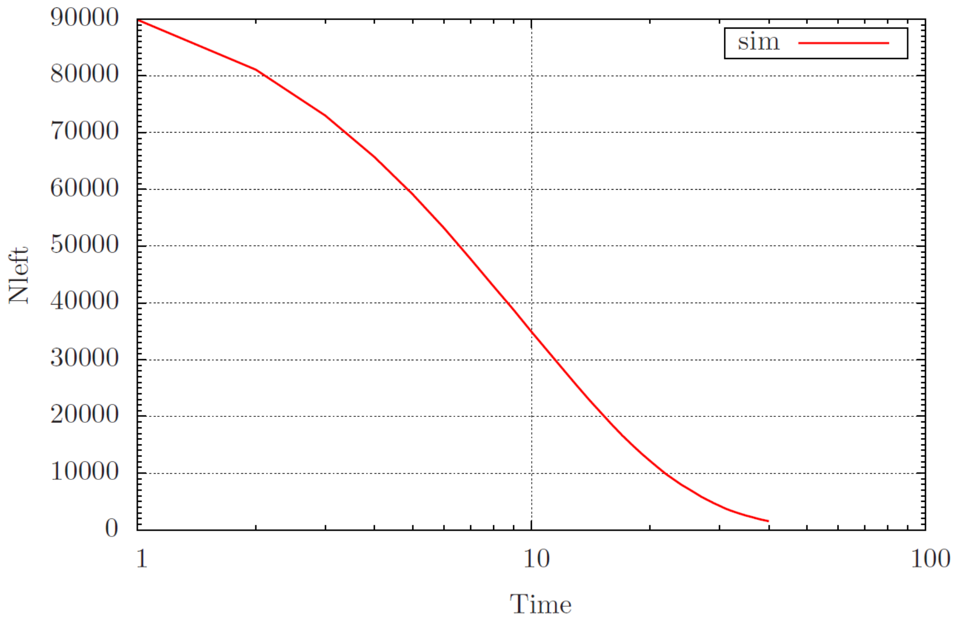
$$\frac{dN_X(t)}{dt} = -\omega_X N_X(t),$$

$$\frac{dN_Y(t)}{dt} = -\omega_Y N_Y(t) + \omega_X N_X(t).$$

可以取大量样本，每个样本包含多个粒子，模拟这些粒子的演进：
产生随机数 与 衰变几率 比较

引言: 粒子衰变

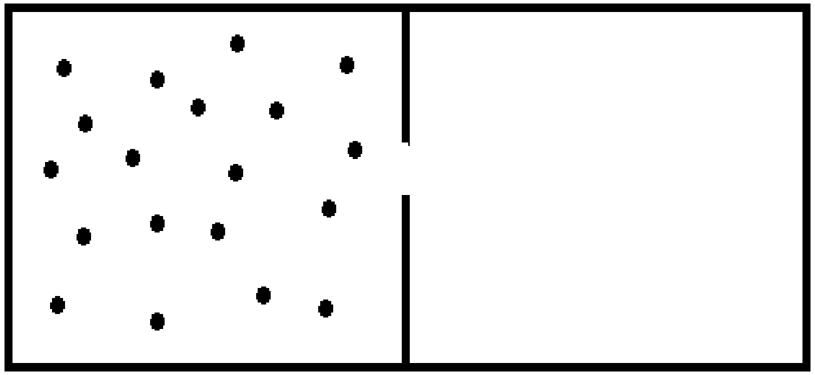
```
void mc_sampling(int initial_n_particles, int
max_time,
        double decay_probability, int *ncumulative)
{
  int time, np, n_unstable, particle_limit;
  n_unstable = initial_n_particles;
// accumulate the number of particles per time
step per trial
  ncumulative[0] = initial_n_particles;
// loop over each time step
  for (time=1; time <= max_time; time++){
    particle_limit = n_unstable;
    for ( np = 1; np <= particle_limit; np++) {
      if( double(rand())/RAND_MAX <=
decay_probability) {
          n_unstable=n_unstable-1;
        }
    } // end of loop over particles
    ncumulative[time] = n_unstable;
  } // end of loop over time steps
} // end mc_sampling function
```



引言: 盒中粒子

在很多物理问题中，可以直接模拟物理过程，而不用写下比如描述系统的微分方程。唯一的要求是需要知道几率分布函数。

t=0



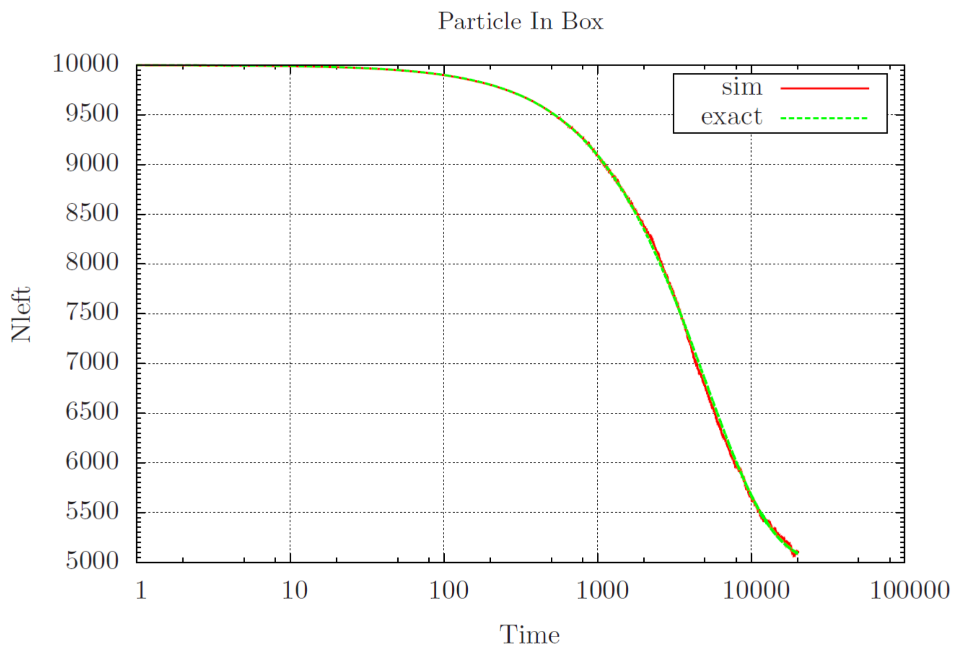
单位时间内，左边粒子移动到右边的几率为

$$n_l / N$$

- Choose the number of particles N.
- Make a loop over time, where the maximum time (or maximum number of steps) should be larger than the number of particles N.
- For every time step Δt there is a probability n_l/N for a move to the right. Compare this probability with a random number x.
- If $x \leq n_l/N$, decrease the number of particles in the left half by one, i.e., $n_l = n_l - 1$. Else, move a particle from the right half to the left, i.e., $n_l = n_l + 1$.
- Increase the time by one unit (the external loop).

引言: 盒中粒子

```
int main(int argc, char* argv[])
{ srand(unsigned(time(0)));  // seed the randomizer
  int initial_n_particles, max_time;
  int time, random_n, nleft;
  ofstream ofile("PinBox.dat");
  cout << "Initial number of particles = " << endl ;
  cin >> initial_n_particles;
  nleft = initial_n_particles;
  max_time = 2*initial_n_particles;
  for( time=0; time <= max_time; time++){
     random_n = (int)(initial_n_particles
*double(rand())/RAND_MAX);
      if ( random_n <= nleft){
       nleft -= 1;
     }
     else {
       nleft += 1;
     }
   ofile << setiosflags(ios::showpoint | ios::uppercase);
   ofile << setw(15) << time;
   ofile << setw(15) << nleft;
  }
  return 0;
} // end main function
```



MC积分

MC积分

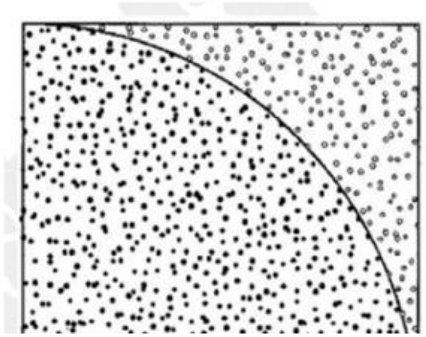
- 蒙特卡洛方法最重要的应用在于对高维定积分的计算。
- 采用蒙卡模拟的方法来求解这种确定性的数学物理问题时，必须先选择一个合适的概率模型，利用此概率模型试验所得的随机事件的统计结果等价于待求问题的解。
- 设想一个物理系统由相互作用的多个粒子组成，如凝聚态物质中的原子、原子中的电子，而每个粒子都可以有几个自由度，要描述这个系统就要涉及高维的积分。
- 如对m个原子组成的气体，经典配分函数是：

$$Z = \int d\mathbf{r}_1 ... d\mathbf{r}_m exp\left\{ -\beta \sum_{i<j} U(r_{ij}) \right\}$$

- 这里的$\beta = (k_B T)^{-1}$代表温度，$U(r_{ij})$是两体相互作用势。
- 除非m相当小时，这个式子要用其他任意一种数值积分计算方法都是不可能算出的。

投点法

- 假设我们要计算地图上某个区域的面积，可以均匀的在包围此区域的一个方框里撒上小石子，待计算面积与正方形面积之比即为该区域中的石子数目与总数之比。显然，石子越小，撒的数目越多越均匀，则求得的面积越准确。这就是用Monte Carlo 方法计算定积分的原理。



- 因为f(x)的定积分值即为曲线下的面积值

$$S = \int_a^b f(x)dx = S_0(n/N)$$

- 这里的S_0为方形区域的面积，N是总点数，n是掷入f(x)下面积区域中的点数，该方法从原理上与舍选抽样法是一致的
- 例如，用MonteCarlo方法求π值。产生一对在$[-1,1]$区间中均匀分布的随机数作为点的坐标(x, y)值，判断条件$x^2+y^2\leq 1$是否成立，成立则计数n值，当总点数N足够大时，有：

$$\frac{\pi}{2^2} = \lim_{N\to\infty}\left(\frac{n}{N}\right)$$

平均值法

- 平均值法是建立在大数定理的基础之上；
- 如果函数f(x)在$[a,b]$区间，以均匀的概率分布密度随机地取N个数x_i，对每个x_i 计算出函数值$f(x_i)$。大数法则告诉我们这些函数值之和除以N所得的值将收敛于函数f的期望值，即：

$$\langle f\rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i) = \frac{1}{(b-a)}\int_a^b f(x)dx$$

$$Var(<f>) = \frac{V^2}{N^2}\sum_{1}^{N} Var(f) = V^2\frac{Var(f)}{N}$$

- 故有：

$$\int_a^b f(x)dx = \frac{(b-a)}{N}\sum_{i=1}^{N} f(x_i)$$

- 在简单抽样中，我们由均匀分布中选取随机数，并不考虑被积函数的具体情况。因此，被积函数的极大处和极小处有相同的抽样权重。
- 但是对积分贡献较大的更多在函数值较大处，故直观上就可以看出，非权重的简单抽样效率较低。即为了获得较高计算结果的精度，需要大量的抽样。

中心极限定理

- 与大数法则一样，概率论中的中心极限定理同样是MonteCarlo方法应用于统计计算的基础。它是对积分的蒙特卡洛估计值收敛于该积分的正确结果的收敛程度的研究
- 中心极限定理告诉我们：在有足够大，但又有限的抽样数N的情况下，蒙特卡洛估计值的误差分布。即：

$$P\left\{\left|\frac{\langle f\rangle-\mu}{\sigma_f/\sqrt{N}}\right|<\beta\right\}\rightarrow\Phi(\beta)$$

- 其中$\mu=\frac{1}{b-a}\int_a^b f(x)dx$，$\sigma_f$是函数f(x)的标准偏差，$\Phi(\beta)$是Gauss正态分布，那么对于积分值得标准偏差$\sigma_s$，就有：

$$\sigma_s=|\langle f\rangle-\mu|\propto\frac{\sigma_f}{\sqrt{N}}$$

- 从上式可以看到蒙特卡洛积分的误差和σ_f与N 有关。在方差固定时，增加模拟次数可以有效地减小误差。

```
//  int_0^1 4/(1+x^2) dx = pi
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
double func(double x);
int main()
{ srand(unsigned(time(0)));  // seed the randomizer
   int i, n;
   double crude_mc, x, sum_sigma, fx, variance;
cin >> n;
   crude_mc = sum_sigma=0. ;
for ( i = 1; i <= n; i++){
      x=rand()/(double)RAND_MAX;
       //cout<<x<<endl;
      fx=func(x);
      crude_mc += fx;
      sum_sigma += fx*fx;  }
   crude_mc = crude_mc/((double) n );
   sum_sigma = sum_sigma/((double) n );
   variance=sum_sigma-crude_mc*crude_mc;
   cout << " variance= " << variance << " Integral =
"<< crude_mc<< endl;
} // end of main program
```

```
// this function defines the function to integrate
double func(double x)
{  double value;
   value = 4/(1.+x*x);
   return value;
} // end of function to evaluate
```

Crude MC

To compare with Vegas

多重定积分

- 如试验次数增加100倍，精度提高10倍。当然这样做就增加了计算的机时，提高了费用。所以在考虑蒙特卡洛方法的精确度时，不能只是简单地减少方差和增加模拟次数，还要同时兼顾计算费用，即机时耗费。通常以方差和费用的乘积作为衡量方法优劣的标准。
- 多重定积分 用简单的抽样的蒙特卡洛方法(不考虑权重)，则前面一维积分的式子，

$$\int_a^b f(x)dx = \frac{(b-a)}{N}\sum_{i=1}^{N} f(x_i)$$

- 可以推广为：

$$\int_{a_1}^{b_1} dx_1 \ldots \int_{a_n}^{b_n} dx_n f(x_1, x_2, \ldots x_n) = \frac{1}{N}\left[\prod_{j=1}^{n}(b_j - a_j)\right]\sum_{i=1}^{N} f(x_{1_i}, \ldots x_{n_i})$$

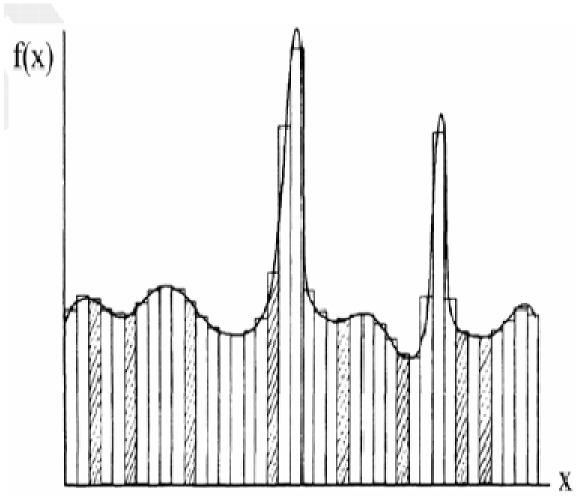
- 可以看到其中n积分的维度数目，N是抽样的次数
- 其中对每个坐标的抽样值是在相应的区间范围内均匀抽取的

多重定积分

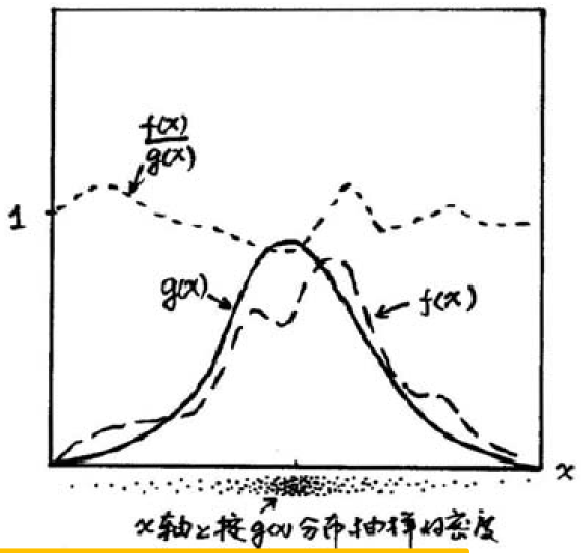
- 对于固定的样本数N值，蒙特卡洛方法给出的误差$\sim 1/\sqrt{N}$，而对于固定网格点法，由于每一维上的平均点数为$\mathrm{N}^{1/\mathrm{d}}$，因此误差$\sim 1/\mathrm{N}^{1/2\mathrm{d}} > 1/\sqrt{\mathrm{N}}$ 当多重积分的维数$d > 4$时，几乎没有其他数值计算方法可以超过蒙特卡洛方法。
- 在实际计算时如何选取N值是根据被积函数的性质和要求的精度确定的，通常要比较不同N值下的结果，看它们的差异再确定需要增加的N值
- 如果积分中的被积函数不光滑时，则用目前的非权重简单抽样蒙特卡洛方法得到的精度可能很差，如图中有一些极大值区域没有被随机抽出。统计力学中Boltzman分布函数在相空间的大部分区域其值都是很小的，真正有贡献的区域范围很窄。
- 简单抽样为了提高精度，增大抽样量的办法不可取，因为这样的计算效率很低。
- 引入带权重的重要抽样法，将既可以保证计算精度又有很高的效率。

重要抽样法

简单抽样法积分近似度较差



重要抽样法示意图



- 重要抽样法的原理起源于数学上的变量代换方法的思想，即：

$$\int_0^1 f(x)dx = \int_0^1 \frac{f(x)}{g(x)} g(x)dx = \int \frac{f(x)}{g(x)} dG(x)$$

- 此时随机点的选择不再是简单抽样法中的均匀选择，而是以分布函数G(x)分布的
- 这里g(x)称为偏倚分布密度函数。

重要抽样法

- 该方法使原本对f(x)的抽样，变成由另一个分布密度函数$\frac{f(x)}{g(x)}$中产生简单子样，并附带一个权重g(x)。这种方法也称为偏倚抽样法。
- 公式右边积分中被积函数的方差为$\sigma^2\{f/g\}$。如果g(x)选择恰当，并使它在积分域内的函数曲线形状与f(x)，则该方差可以变得很小。
- 函数g(x)应当满足如下条件：
  - (1)g(x)应当是个分布密度函数，即$\int_0^1 g(x)dx = 1$
  - (2)f(x)/g(x)不应在积分域内起伏太大，如上图，使f(x)/g(x) ~ 1，以保证方差$\sigma^2\{f/g\}$比$\sigma^2\{f\}$小
  - (3)分布密度函数g(x)所对应的分布函数G(x) 能够比较方便地解析求出。
  - (4)能方便地产生在积分域内满足分布函数G(x)分布的随机点。
- 如能按上述条件找到函数g(x)，我们就可以依下列步骤求积分：
  - (1)根据分布密度函数g(x)产生随机点x。例如采用反函数法。
  - (2)求出各抽样点x 的函数值f(x)/g(x)，并将所有点上的该函数值迭加起来，再除以抽样点数就得到积分结果。

重要抽样法 (Importance Sampling)

- 公式表示为：

$$\int_0^1 f(x)dx = \langle f/g \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{g(x_i)}$$

- 重要抽样法无疑是蒙特卡洛计算中最基本和常用的技巧之一。它在提高计算速度和增加数值结果的稳定性方面都有很大的潜力。
- 它的局限性是能寻找出某分布密度函数g(x)，并能解析求出其对应的分布函数G(x)的情况并不多。
- 例如，计算积分：

$$I = \int_0^1 \exp(-x/2)dx$$

- 该式当然可以解析算出，这里主要举例说明重要抽样法的求解过程。
- 式中的指数项随x增加而迅速减小，g(x)必须是另一有相同行为的函数，我们用指数函数在x = 0附近的渐近展开来近似：

重要抽样法

$$g(x) = 1 - \frac{x}{2} + \frac{x^2}{8} - \frac{x^3}{48} \ldots \approx 1 - \frac{x}{2}$$

- 由反函数法抽样：

$$G(x) = \int_0^x (1 - x/2)dx = x - x^2/4$$

$$\eta = 2(1 \pm \sqrt{1-\xi})$$

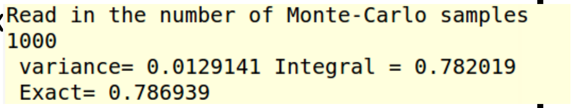
- 由于G(0) = 0，故上式中括号内应取负号，即：

$$\eta = 2(1 - \sqrt{1-\xi})$$

- 原积分即可表示成：

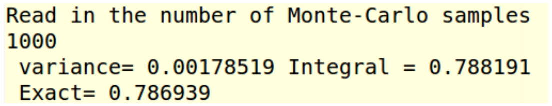
$$I = \frac{1}{N}\sum_{i=1}^{N} \exp\frac{-(1-\sqrt{1-\xi_i})}{\sqrt{1-\xi_i}}$$

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;
double func(double x);
int main()
{ srand(unsigned(time(0)));  // seed the randomizer
  int i, n;
  double crude_mc, x, sum_sigma, fx, variance;
  cin >> n;
  crude_mc = sum_sigma=0. ;
  // a crude Monte-Carlo method
  for ( i = 1; i <= n; i++){
    x=rand()/(double)RAND_MAX;
    fx=func(x);
    crude_mc += fx;
    sum_sigma += fx*fx;   }
  crude_mc = crude_mc/((double) n );
  sum_sigma = sum_sigma/((double) n );
  variance=sum_sigma-crude_mc*crude_mc;
  variance=sqrt(variance)/double(n); }

double func(double x
{
  double value;
  value = exp(-x/2.0);
  return value;
} // end of function to evaluate
```



```
Read in the number of Monte-Carlo samples
1000
 variance= 0.0129141 Integral = 0.782019
 Exact= 0.786939
```

```
//  int_0^1 exp(-x/2) dx =2(1-exp(-0.5))
//  importance sampling: g(x)=1-x/2.0
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;
double func(double x);
int main()
{ srand(unsigned(time(0)));  // seed the randomizer
  int i, n;
  double crude_mc, tx, x, sum_sigma, fx, variance;
  cin >> n;
  crude_mc = sum_sigma=0. ;
  for ( i = 1; i <= n; i++){
  tx=3.0/4.0*rand()/(double)RAND_MAX;
  //0=<tx<=3/4;
   x=2.*(1-sqrt(1.0-tx));
   fx=3.0/4.0*func(x);
   crude_mc += fx;
   sum_sigma += fx*fx;  }
  crude_mc = crude_mc/((double) n );
  sum_sigma = sum_sigma/((double) n );
  variance=sum_sigma-crude_mc*crude_mc;
  variance=sqrt(variance)/double(n);}
double func(double x)
{ double value;
  value = exp(-x/2.0)/(1.-x/2.0);
  return value;
} // end of function to evaluate
```



```
Read in the number of Monte-Carlo samples
1000
 variance= 0.00178519 Integral = 0.788191
 Exact= 0.786939
```

```
//  Breit-Wigner
//   mk=10., gk=1.0;
//   int_0^200 sqrt(x)/((x-mk*mk)*(x-mk*mk)+mk*mk*gk*gk);
//   theta: atan(-10.) to atan(10.)
```

$$\theta_k = \tan^{-1}\left(\frac{s_k - m_k^2}{m_k\Gamma_k}\right), \qquad ds_k = \frac{(s_k - m_k^2)^2 + m_k^2\Gamma_k^2}{m_k\Gamma_k} d\theta_k ,$$

```
int main()
{ srand(unsigned(time(0)));  // seed the randomizer
   int i, n;
   double crude_mc, x, sum_sigma, fx, variance;
    cin >> n;
   crude_mc = sum_sigma=0. ;
   for ( i = 1; i <= n; i++){
     x=rand()/(double)RAND_MAX;
     x=200.*x; // 0-200
     fx=func(x);
     crude_mc += 200.*fx;
     sum_sigma += 200.*200.*fx*fx; }
   crude_mc = crude_mc/((double) n );
   sum_sigma = sum_sigma/((double) n );
   variance=sum_sigma-crude_mc*crude_mc;
   variance=sqrt(variance)/double(n);
   cout << " variance= " << variance << " Integral = "
        << crude_mc<< endl;
} // end of main program

// this function defines the function to integrate
double func(double x)
{
   double value;
   double mk=10., gk=1.0;
  value = sqrt(x)/((x-mk*mk)*(x-mk*mk)+mk*mk*gk*gk);
   return value;}
```

```
int main()
{ srand(unsigned(time(0)));  // seed the randomizer
   int i, n;
   double crude_mc, x, sum_sigma, fx, variance;
   cin >> n;
   crude_mc = sum_sigma=0. ;
   double pi=3.1415926535897;
   for ( i = 1; i <= n; i++){
     x=rand()/(double)RAND_MAX;
     x=-atan(10.)+2.*atan(10.)*x; // atan(-10) to atan(10)
     fx=func(x);
     crude_mc += 2.*atan(10.)*fx;
     sum_sigma += 2.*atan(10.)*2.*atan(10.)*fx*fx; }
   crude_mc = crude_mc/((double) n );
   sum_sigma = sum_sigma/((double) n );
   variance=sum_sigma-crude_mc*crude_mc;
 variance=sqrt(variance)/double(n);
cout << " variance= " << variance << " Integral = "
       << crude_mc<< endl;
} // end of main program

double func(double x)
{
   double value;
   double mk=10., gk=1.0;
   double ax=tan(x)*mk*gk+mk*mk;
   value = sqrt(ax)/mk/gk;
   return value; } // end of function to evaluate
```

Stratified抽样法

The idea of stratified sampling is quite different from importance sampling, **by dividing volume V into sub-regions**. The optimal allocation is to have the number of points in each region j proportional to σ_j

$$\langle\langle f\rangle\rangle \equiv \frac{1}{V}\int f\,dV \qquad \langle f\rangle \equiv \frac{1}{N}\sum_i f(x_i) \qquad \mathrm{Var}\left(\langle f\rangle\right) = \frac{\mathrm{Var}\left(f\right)}{N}$$

Suppose we divide the volume V into two equal, disjoint subvolumes, denoted a and b, and sample $N/2$ points in each subvolume. Then another estimator for $\langle\langle f\rangle\rangle$, different from equation $\langle f\rangle$, which we denote $\langle f\rangle'$, is

$$\langle f\rangle' \equiv \frac{1}{2}\left(\langle f\rangle_a + \langle f\rangle_b\right)$$

in other words, the mean of the sample averages in the two half-regions. The variance of estimator $\langle f\rangle'$ is given by

$$\begin{aligned}\mathrm{Var}\left(\langle f\rangle'\right) &= \frac{1}{4}\left[\mathrm{Var}\left(\langle f\rangle_a\right) + \mathrm{Var}\left(\langle f\rangle_b\right)\right] \\ &= \frac{1}{4}\left[\frac{\mathrm{Var}_a\left(f\right)}{N/2} + \frac{\mathrm{Var}_b\left(f\right)}{N/2}\right] \\ &= \frac{1}{2N}\left[\mathrm{Var}_a\left(f\right) + \mathrm{Var}_b\left(f\right)\right]\end{aligned}$$

Stratified后的方差总是小于之前的

From the definitions already given, it is not difficult to prove the relation

$$\mathrm{Var}\left(f\right) = \frac{1}{2}\left[\mathrm{Var}_a\left(f\right) + \mathrm{Var}_b\left(f\right)\right] + \frac{1}{4}\left(\langle\langle f\rangle\rangle_a - \langle\langle f\rangle\rangle_b\right)^2$$

VEGAS

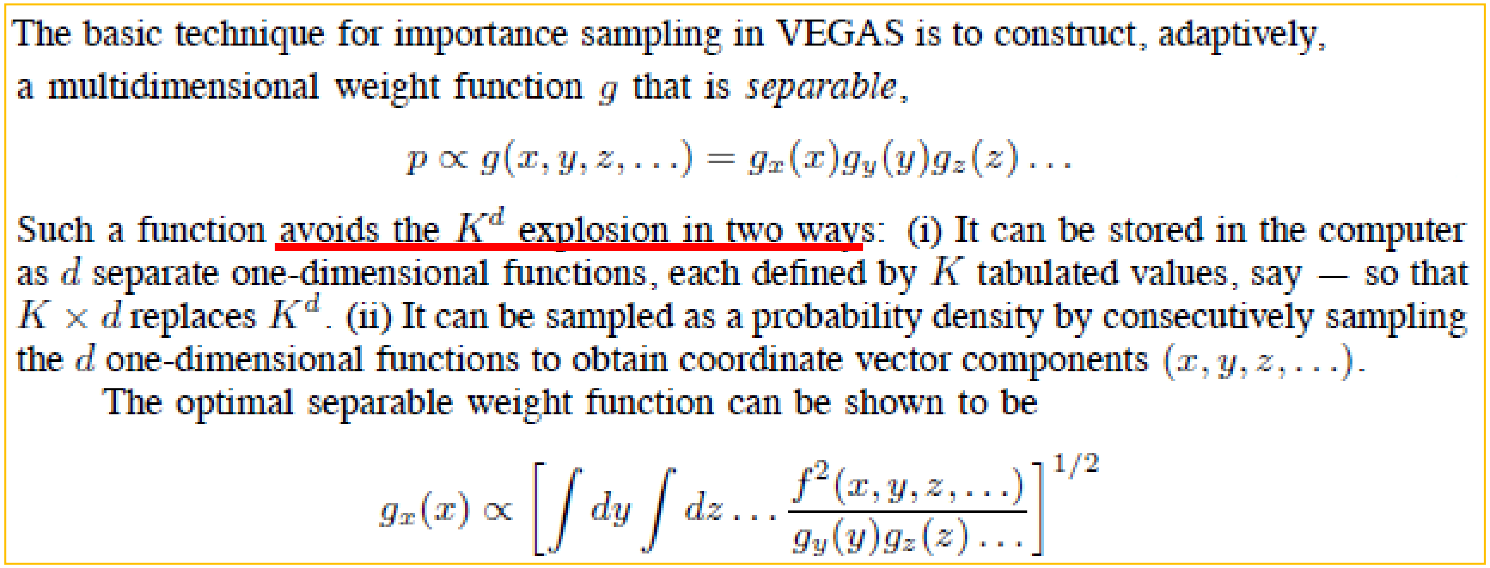
Journal of Computational Physics
Volume 27, Issue 2, May 1978, Pages 192-203

A new algorithm for adaptive multidimensional integration ☆

G Peter Lepage

Abstract

A new general purpose algorithm for multidimensional integration is described. It is an iterative and adaptive Monte Carlo scheme. The new algorithm is compared with several others currently in use, and shown to be considerably more efficient than all of these for a number of sample integrals of high dimension ($n \gtrsim 4$).



The basic technique for importance sampling in VEGAS is to construct, adaptively, a multidimensional weight function g that is *separable*,

$$p \propto g(x, y, z, \ldots) = g_x(x) g_y(y) g_z(z) \ldots$$

Such a function avoids the K^d explosion in two ways: (i) It can be stored in the computer as d separate one-dimensional functions, each defined by K tabulated values, say — so that $K \times d$ replaces K^d. (ii) It can be sampled as a probability density by consecutively sampling the d one-dimensional functions to obtain coordinate vector components $(x, y, z, \ldots)$.

The optimal separable weight function can be shown to be

$$g_x(x) \propto \left[\int dy \int dz \ldots \frac{f^2(x, y, z, \ldots)}{g_y(y) g_z(z) \ldots} \right]^{1/2}$$

GSL - GNU Scientific Library

GNU Operating System

Sponsored by the ***Free Software Foundation***

Introduction

The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It is free software under the GNU General Public License.

The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite.

VEGAS

```
c  int_0^1 exp(x/2)dx
      include 'treefuc.f'
      program main
      implicit double precision (a-h,m,o-z)
     common/bveg1/ ncall, itmx, nprn, ndev,
xl(10), xu(10), acc
      external fes
      open(10, file='output.txt', status='replace')
     read (*,*) ncall
      write (10,*)'ncall=',ncall
      nprn=-1
      xl(1)=0.0d0
      xu(1)=1.0d0
      itmx=3
      call vegas(1,fes,vfes,sd,chi2a)
      write(10,*)vfes, sd, chi2a
      close(10)
      end
```

```
      double precision function fes(xx,wgt)
       implicit double precision (a-h,m,o-z)
       dimension xx(10)
       double precision xxx1
       xxx1=xx(1)
       fes=dexp(-xxx1/2.0)
       return
       end
```

makefile

```
FFLAGS = -g -m64 -fno-automatic -ffixed-line-
length-none
FC=gfortran
OBJECTS =  vegas.o \
        main.o \

lq:  $(OBJECTS)
             $(FC) $(FFLAGS) -o lq $(OBJECTS)

f.o:  $(FC) -c -o $*.o  $(FFLAGS) $*.f
```

VEGAS

https://pypi.org/project/vegas/

vegas 3.3.4

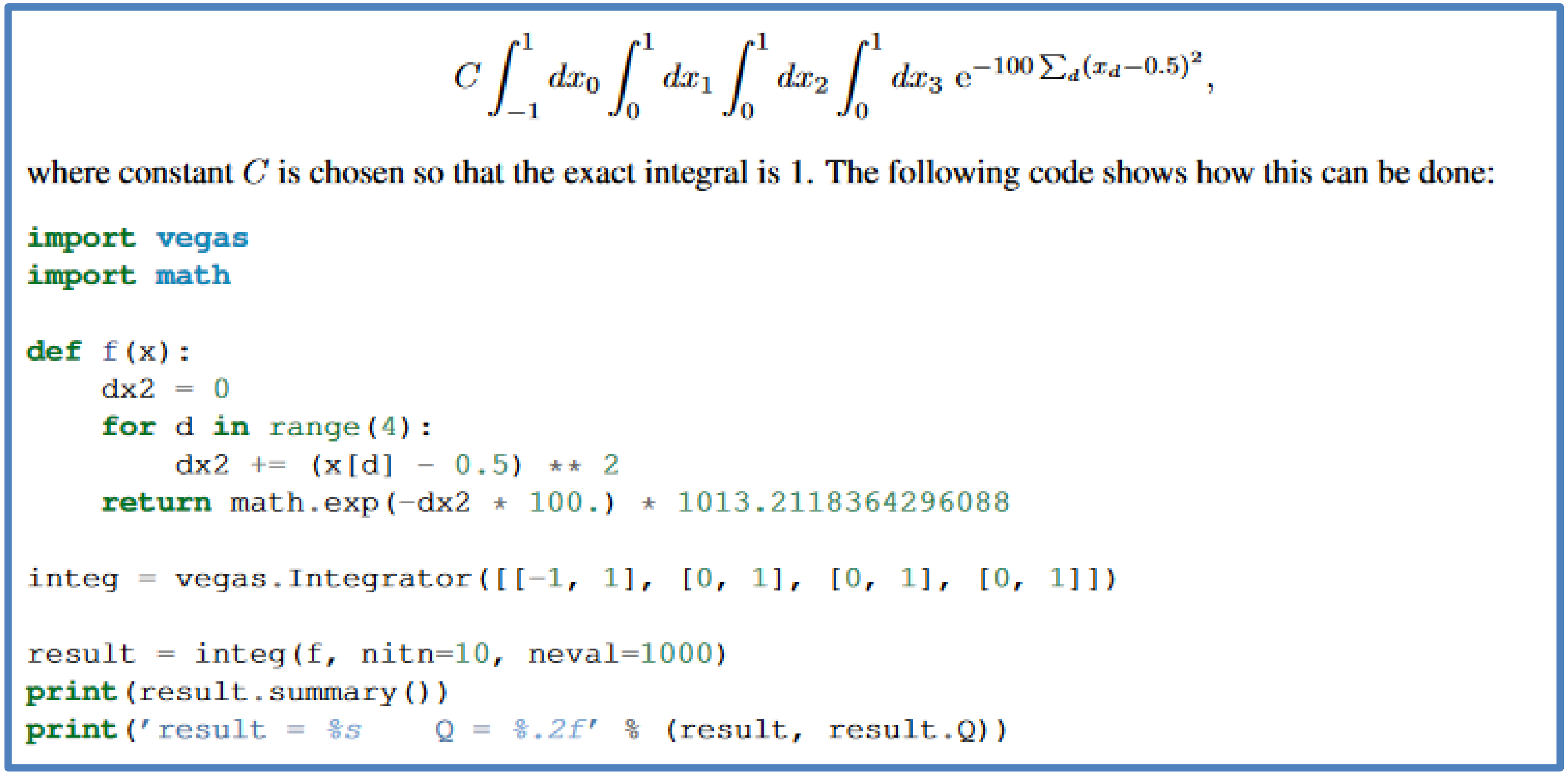
```
pip install vegas
```

Project description

This package provides tools evaluating multidimensional

integrals numerically using an enhanced version of

the adaptive Monte Carlo vegas algorithm (G. P. Lepage, J. Comput. Phys. 27(1978) 192).



$$C \int_{-1}^{1} dx_0 \int_0^1 dx_1 \int_0^1 dx_2 \int_0^1 dx_3 \; e^{-100 \sum_d (x_d - 0.5)^2},$$

where constant C is chosen so that the exact integral is 1. The following code shows how this can be done:

```
import vegas
import math

def f(x):
    dx2 = 0
    for d in range(4):
        dx2 += (x[d] - 0.5) ** 2
    return math.exp(-dx2 * 100.) * 1013.2118364296088

integ = vegas.Integrator([[-1, 1], [0, 1], [0, 1], [0, 1]])

result = integ(f, nitn=10, neval=1000)
print(result.summary())
print('result = %s    Q = %.2f' % (result, result.Q))
```

using `nitn=10` iterations of the `vegas` algorithm, each of which uses no more than `neval=1000` evaluations of the integrand. Each iteration produces an independent estimate of the integral. The final estimate is the weighted average of the results from all 10 iterations, and is returned by `integ(f ...)`. The call `result.summary()` returns a summary of results from each iteration.

This code produces the following output:

```
itn   integral        wgt average     chi2/dof        Q
-------------------------------------------------------
  1   2.4(1.9)        2.4(1.9)            0.00     1.00
  2   1.19(32)        1.23(32)            0.42     0.52
  3   0.910(90)       0.934(87)           0.68     0.51
  4   1.041(70)       0.999(55)           0.76     0.52
  5   1.090(43)       1.055(34)           1.00     0.41
  6   0.984(34)       1.020(24)           1.24     0.29
  7   1.036(27)       1.027(18)           1.07     0.38
  8   0.987(22)       1.011(14)           1.20     0.30
  9   0.995(18)       1.005(11)           1.11     0.35
 10   0.993(17)       1.0015(91)          1.02     0.42

result = 1.0015(91)    Q = 0.42
```

`result.mean` — weighted average of all estimates of the integral;

`result.sdev` — standard deviation of the weighted average;

`result.chi2` — χ^2 of the weighted average;

`result.dof` — number of degrees of freedom;

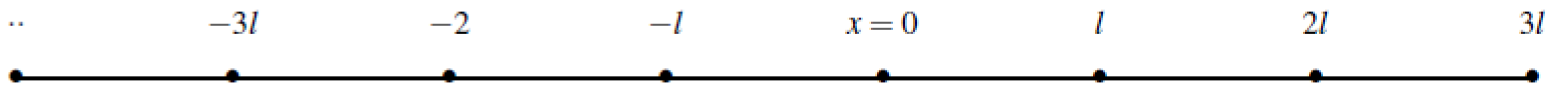
`result.Q` — *Q* or *p-value* of the weighted average's χ^2;

`result.itn_results` — list of the integral estimates from each iteration.

In this example the final Q is 0.42, indicating that the χ^2 for this average is not particularly unlikely and thus the error estimate is most likely reliable.

随机行走

随机行走



Consider now a random walker in one dimension, with probability R of moving to the right and L for moving to the left. At $t=0$ we place the walker at $x=0$, as indicated in Fig. 12.2. The walker can then jump, with the above probabilities, either to the left or to the right for each time step. Note that in principle we could also have the possibility that the walker remains in the same position. This is not implemented in this example. Every step has length $\Delta x = l$. Time is discretized and we have a jump either to the left or to the right at every time step. Let us now assume that we have equal probabilities for jumping to the left or to the right, i.e., $L = R = 1/2$. The average displacement after n time steps is

$$\langle x(n)\rangle = \sum_i^n \Delta x_i = 0 \qquad \Delta x_i = \pm l,$$

$$\langle x(n)^2\rangle = \left(\sum_i^n \Delta x_i\right)\left(\sum_j^n \Delta x_j\right) = \sum_i^n \Delta x_i^2 + \sum_{i\neq j}^n \Delta x_i \Delta x_j = l^2 n.$$

For many enough steps the non-diagonal contribution is

$$\sum_{i\neq j}^N \Delta x_i \Delta x_j = 0,$$

since $\Delta x_{i,j} = \pm l$. The variance is then

$$\langle x(n)^2\rangle - \langle x(n)\rangle^2 = l^2 n.$$

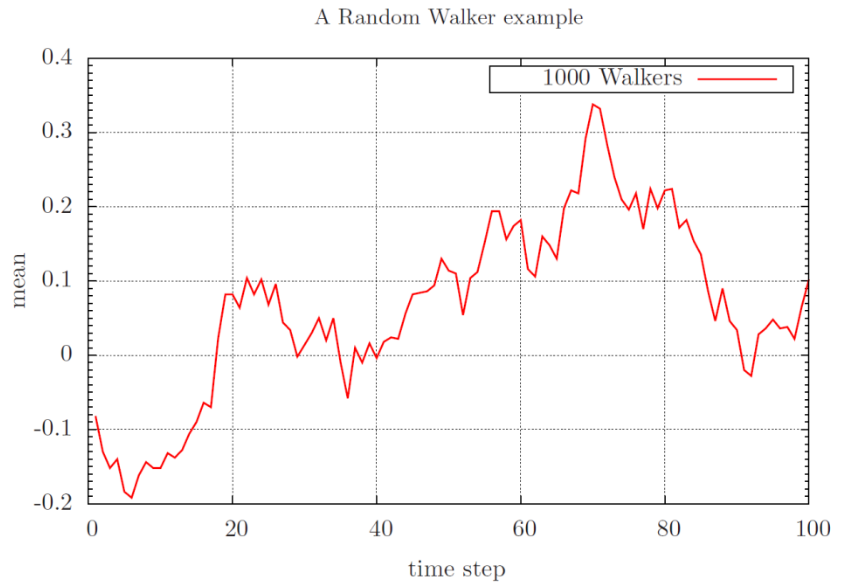
It is also rather straightforward to compute the variance for $L \neq R$. The result is

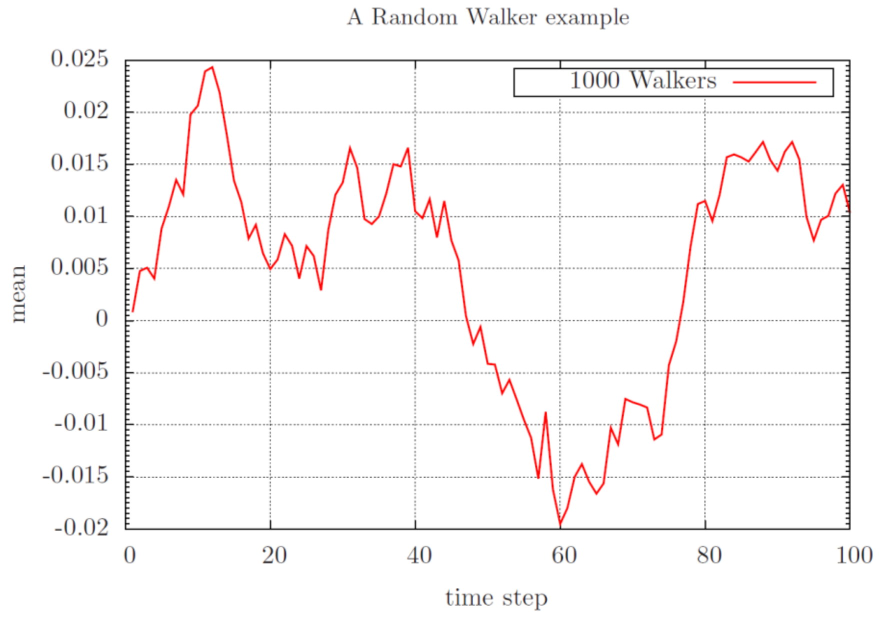
$$\langle x(n)^2\rangle - \langle x(n)\rangle^2 = 4LRl^2 n.$$

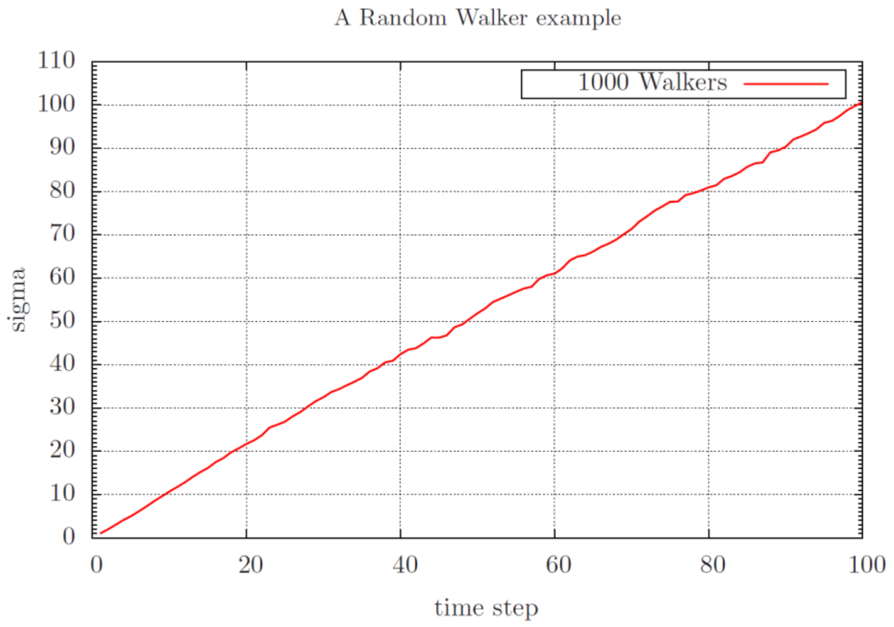
$$\boxed{\begin{gathered}\frac{\partial\langle x^2\rangle}{\partial t} = -Dxw(x,t)|_{x=\pm\infty} + 2D\int_{-\infty}^{\infty} w(x,t)dx = 2D,\\ \langle x^2\rangle = 2Dt,\\ \langle x^2\rangle - \langle x\rangle^2 = 2Dt.\end{gathered}}$$

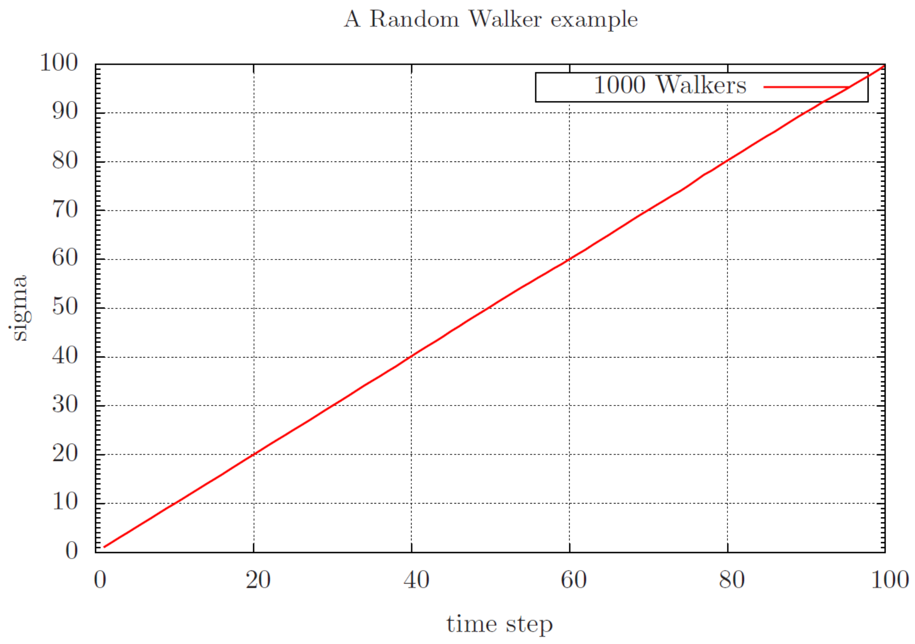
与扩散方程对比，得到：

$$D = \frac{l^2}{2\Delta t}$$

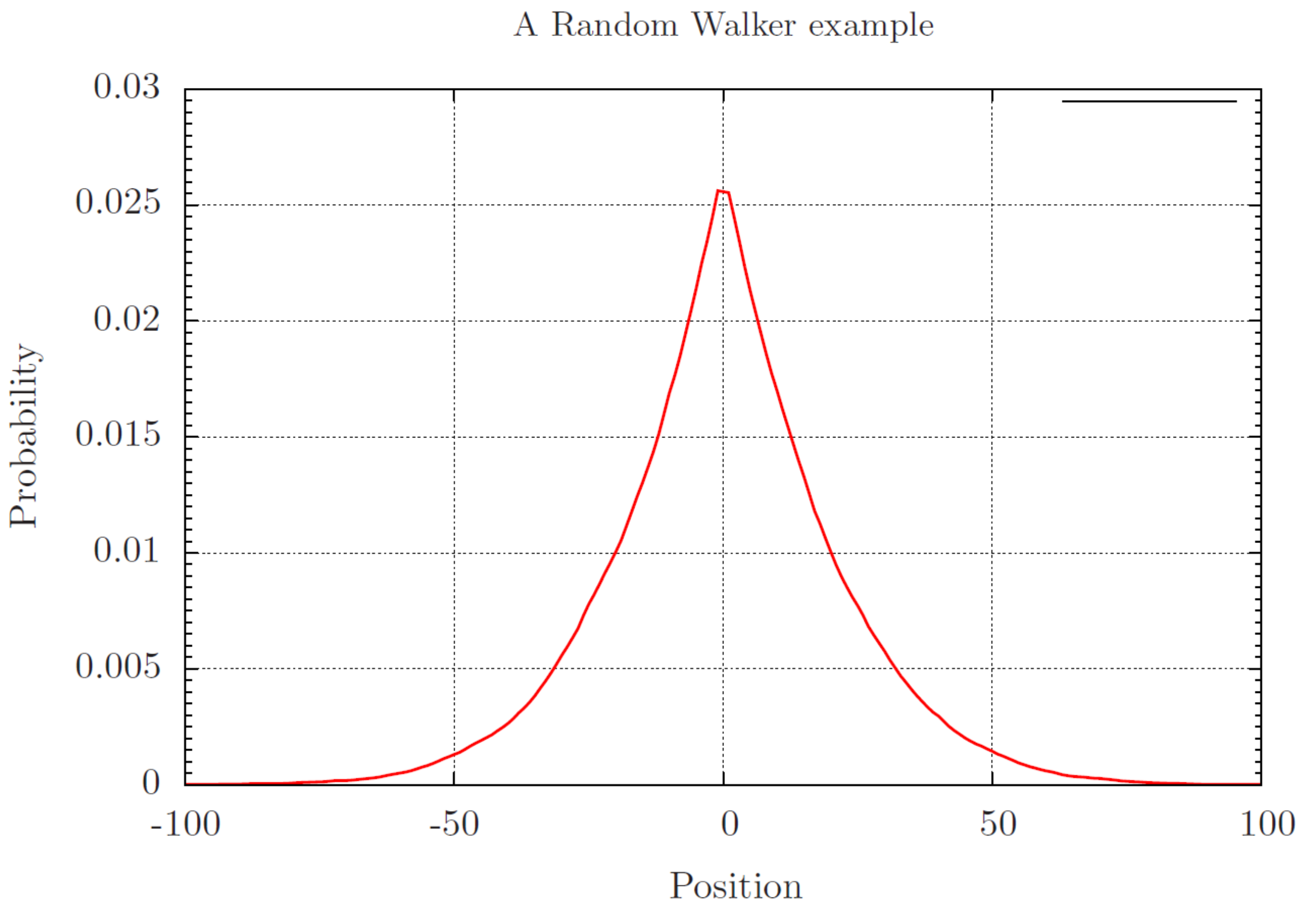

A Random Walker example
1000 Walkers
mean
time step
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
20
40
60
80
100


A Random Walker example
1000 Walkers
mean
time step
0.025
0.02
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
0
20
40
60
80
100


A Random Walker example
1000 Walkers
sigma
time step
110
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100


A Random Walker example
1000 Walkers
sigma
time step
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100

```
for( int i = -number_walks; i <= number_walks; i++){
  double histogram = probability[i+number_walks]/norm;
  probfile << setiosflags(ios::showpoint | ios::uppercase);
  probfile << setw(6) << i;
  probfile << setw(15) << setprecision(8) << histogram << endl;  }
```

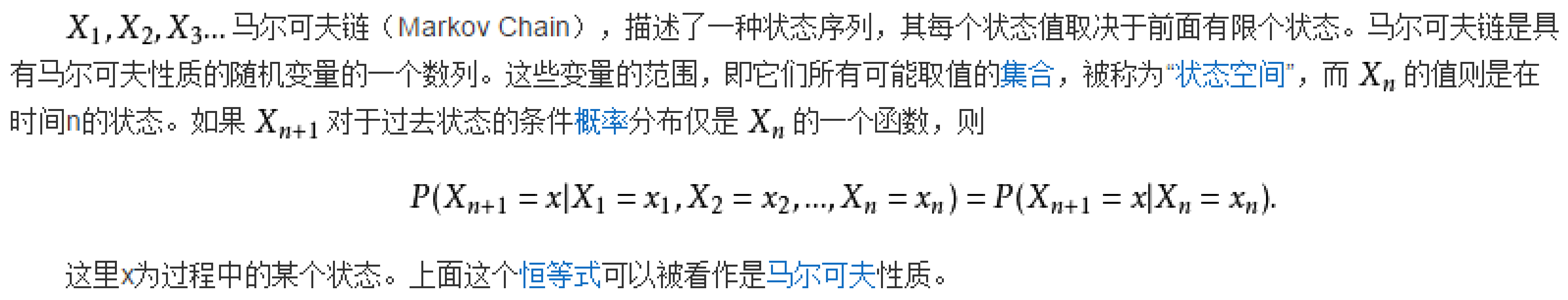


Walker: 10K
TimeStep: 900

Markov Chain

马尔可夫链，因安德烈·马尔可夫（A.A.Markov，1856－1922）得名，是指数学中具有马尔可夫性质的离散事件随机过程。该过程中，在给定当前知识或信息的情况下，**过去（即当前以前的历史状态）对于预测将来（即当前以后的未来状态）是无关的。**

在马尔可夫链的每一步，系统根据概率分布，可以从一个状态变到另一个状态，也可以保持当前状态。状态的改变叫做转移，与不同的状态改变相关的概率叫做**转移概率**。随机漫步就是马尔可夫链的例子



X_1, X_2, X_3... 马尔可夫链（Markov Chain），描述了一种状态序列，其每个状态值取决于前面有限个状态。马尔可夫链是具有马尔可夫性质的随机变量的一个数列。这些变量的范围，即它们所有可能取值的集合，被称为“状态空间”，而 X_n 的值则是在时间n的状态。如果 X_{n+1} 对于过去状态的条件概率分布仅是 X_n 的一个函数，则

$$P(X_{n+1}=x|X_1=x_1, X_2=x_2, \ldots, X_n=x_n)=P(X_{n+1}=x|X_n=x_n).$$

这里x为过程中的某个状态。上面这个恒等式可以被看作是马尔可夫性质。

随机行走

We wish to study the time-development of a PDF after a given number of time steps. We define our PDF by the function $w(t)$. In addition we define a transition probability W. The time development of our PDF $w(t)$, after one time-step from $t = 0$ is given by

$$w_i(t = \epsilon) = W(j \rightarrow i)w_j(t = 0).$$

This equation represents the discretized time-development of an original PDF. We can rewrite this as a

$$w_i(t = \epsilon) = W_{ij}w_j(t = 0).$$

with the transition matrix W for a random walk left or right (cannot stay in the same position) given by

$$W_{ij}(\epsilon) = W(il - jl, \epsilon) = \begin{cases} \frac{1}{2} & |i - j| = 1 \\ 0 & \text{else} \end{cases}$$

We call W_{ij} for the transition probability and we represent it as a matrix.

随机行走

Both W and w represent probabilities and they have to be normalized, meaning that that at each time step we have

$$\sum_i w_i(t) = 1,$$

and

$$\sum_j W(j \to i) = 1.$$

Further constraints are $0 \leq W_{ij} \leq 1$ and $0 \leq w_j \leq 1$. We can thus write the action of W as

$$w_i(t+1) = \sum_j W_{ij} w_j(t),$$

or as vector-matrix relation

$$\hat{\mathbf{w}}(t+1) = \hat{\mathbf{W}}\hat{\mathbf{w}}(t),$$

and if we have that $||\hat{\mathbf{w}}(t+1) - \hat{\mathbf{w}}(t)|| \to 0$, we say that we have reached the most likely state of the system, the so-called steady state or $\underline{\text{equilibrium state.}}$ Another way of phrasing this is

$$\boxed{\mathbf{w}(t=\infty) = \mathbf{W}\mathbf{w}(t=\infty).}$$

随机行走

Consider the simple 3×3 matrix $\hat{W}$

$$\hat{W} = \begin{pmatrix} 1/4 & 1/8 & 2/3 \\ 3/4 & 5/8 & 0 \\ 0 & 1/4 & 1/3 \end{pmatrix},$$

and we choose our initial state as

$$\hat{w}(t=0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The first iteration is

$$w_i(t=\epsilon) = W(j \rightarrow i) w_j(t=0),$$

resulting in

$$\hat{w}(t=\epsilon) = \begin{pmatrix} 1/4 \\ 3/4 \\ 0 \end{pmatrix}.$$

The next iteration results in

$$w_i(t=2\epsilon) = W(j \rightarrow i) w_j(t=\epsilon),$$

resulting in

$$\hat{w}(t=2\epsilon) = \begin{pmatrix} 5/23 \\ 21/32 \\ 6/32 \end{pmatrix}.$$

Note that the vector $\hat{w}$ is always normalized to 1. We find the steady state of the system by solving the linear set of equations

$$\mathbf{w}(t=\infty) = \mathbf{W}\mathbf{w}(t=\infty).$$

随机行走

This linear set of equations reads

$$\begin{aligned}
W_{11}w_1(t=\infty)+W_{12}w_2(t=\infty)+W_{13}w_3(t=\infty) &= w_1(t=\infty)\\
W_{21}w_1(t=\infty)+W_{22}w_2(t=\infty)+W_{23}w_3(t=\infty) &= w_2(t=\infty)\\
W_{31}w_1(t=\infty)+W_{32}w_2(t=\infty)+W_{33}w_3(t=\infty) &= w_3(t=\infty)
\end{aligned}$$

with the constraint that

$$\sum_i w_i(t=\infty)=1,$$

yielding as solution

$$\hat{w}(t=\infty)=\begin{pmatrix}4/15\\8/15\\3/15\end{pmatrix}.$$

Convergence of the simple example

Iteration	w_1	w_2	w_3
0	1.00000	0.00000	0.00000
1	0.25000	0.75000	0.00000
2	0.15625	0.62625	0.18750
3	0.24609	0.52734	0.22656
4	0.27848	0.51416	0.20736
5	0.27213	0.53021	0.19766
6	0.26608	0.53548	0.19844
7	0.26575	0.53424	0.20002
8	0.26656	0.53321	0.20023
9	0.26678	0.53318	0.20005
10	0.26671	0.53332	0.19998
11	0.26666	0.53335	0.20000
12	0.26666	0.53334	0.20000
13	0.26667	0.53333	0.20000
$\hat{w}(t=\infty)$	0.26667	0.53333	0.20000

随机行走

We have after t-steps

$$\hat{\mathbf{w}}(t) = \hat{\mathbf{W}}^t \hat{\mathbf{w}}(0),$$

with $\hat{\mathbf{w}}(0)$ the distribution at $t = 0$ and $\hat{\mathbf{W}}$ representing the transition probability matrix
We can always expand $\hat{\mathbf{w}}(0)$ in terms of the right eigenvectors $\hat{\mathbf{v}}$ of $\hat{\mathbf{W}}$ as

$$\hat{\mathbf{w}}(0) = \sum_i \alpha_i \hat{\mathbf{v}}_i,$$

resulting in

$$\hat{\mathbf{w}}(t) = \hat{\mathbf{W}}^t \hat{\mathbf{w}}(0) = \hat{\mathbf{W}}^t \sum_i \alpha_i \hat{\mathbf{v}}_i = \sum_i \lambda_i^t \alpha_i \hat{\mathbf{v}}_i,$$

with λ_i the i^{th} eigenvalue corresponding to the eigenvector $\hat{\mathbf{v}}_i$.

If we assume that λ_0 is the largest eigenvector we see that in the limit $t \to \infty$, $\hat{\mathbf{w}}(t)$ becomes proportional to the corresponding eigenvector $\hat{\mathbf{v}}_0$. This is our steady state or final distribution.

$$\langle \mathcal{M}(t) \rangle = \hat{\mathbf{w}}(t)\mathbf{m} = \sum_i \lambda_i^t \alpha_i \hat{\mathbf{v}}_i \mathbf{m}_i. \qquad \langle \mathcal{M}(t) \rangle = \sum_i \lambda_i^t \alpha_i m_i.$$

Since we have that in the limit $t \to \infty$ the mean magnetization is dominated by the largest eigenvalue λ_0, we can rewrite the last equation as

$$\langle \mathcal{M}(t) \rangle = \langle \mathcal{M}(\infty) \rangle + \sum_{i \neq 0} \lambda_i^t \alpha_i m_i.$$

收敛速度取决于第二大的本征值

随机行走

$$\hat{L}\hat{R} = \hat{R}\hat{L} = I, \quad \hat{L} = \hat{R}^{-1}$$

$$\hat{W} = \frac{1}{2}\left(\hat{L}+\hat{R}\right), \quad \hat{W}^2(2\varepsilon) = \frac{1}{4}\left(\hat{L}^2+\hat{R}^2+2\hat{R}\hat{L}\right), \quad \hat{W}^3(3\varepsilon) = \frac{1}{8}\left(\hat{L}^3+\hat{R}^3+3\hat{R}\hat{L}^2+3\hat{R}^2\hat{L}\right)$$

$$\hat{W}^n(n\varepsilon)) = \frac{1}{2^n}\sum_{k=0}^{n}\binom{n}{k}\hat{R}^k\hat{L}^{n-k}, \qquad \sum_{k=0}^{n}\binom{n}{k}\hat{a}^k\hat{b}^{n-k} = (a+b)^n,$$

and using $R^m_{ij} = \delta_{i,(j+m)}$ and $L^m_{ij} = \delta_{(i+m),j}$ we arrive at

$$W(il-jl,n\varepsilon) = \begin{cases} \frac{1}{2^n}\binom{n}{\frac{1}{2}(n+i-j)} & |i-j| \leq n \\ 0 & \text{else} \end{cases},$$

If we place the walker at $x=0$ at $t=0$ we can represent the initial PDF with $w_i(0) = \delta_{i,0}$.

$$w_i(n\varepsilon) = \sum_j (W^n(\varepsilon))_{ij} w_j(0) = \sum_j \frac{1}{2^n}\binom{n}{\frac{1}{2}(n+i-j)}\delta_{j,0},$$

resulting in

$$\boxed{w_i(n\varepsilon) = \frac{1}{2^n}\binom{n}{\frac{1}{2}(n+i)}} \qquad |i| \leq n.$$

随机行走

We can then use the recursion relation for the binomials

$$\binom{n+1}{\frac{1}{2}(n+1+i)} = \binom{n}{\frac{1}{2}(n+i+1)} + \binom{n}{\frac{1}{2}(n+i-1)}$$

$$\Rightarrow \quad w(x,t+\varepsilon) = \frac{1}{2}w(x+l,t) + \frac{1}{2}w(x-l,t).$$

$$\frac{w(x,t+\varepsilon)-w(x,t)}{\varepsilon} = \frac{l^2}{2\varepsilon}\frac{w(x+l,t)-2w(x,t)+w(x-l,t)}{l^2}$$

If we identify $D = l^2/2\varepsilon$ and $l = \Delta x$ and $\varepsilon = \Delta t$ we see that this is nothing but the discretized version of the diffusion equation. Taking the limits $\Delta x \to 0$ and $\Delta t \to 0$ we recover

$$\frac{\partial w(x,t)}{\partial t} = D\frac{\partial^2 w(x,t)}{\partial x^2},$$

the diffusion equation.

随机行走求解泊松微分方程

参见《计算物理学》马文淦， 科学出版社

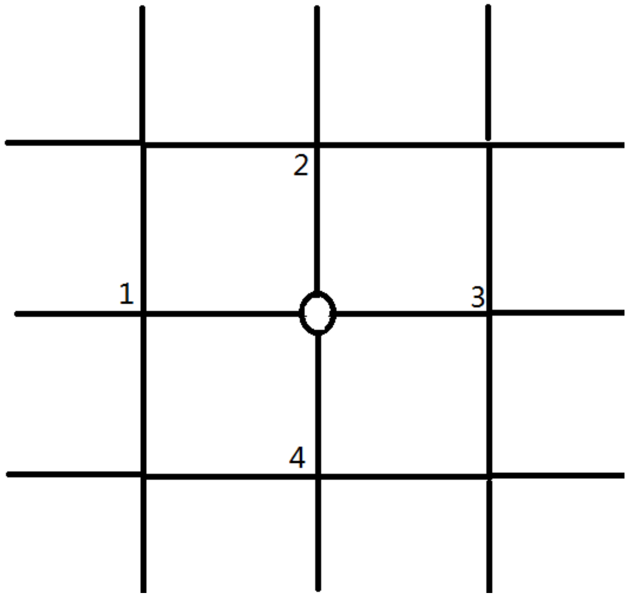
$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=q(x,y)$$

$$\phi|_{\Gamma}=F(s)$$

Γ为求解区域D的边界，s为边界Γ上的点

等步长h正方形格点划分

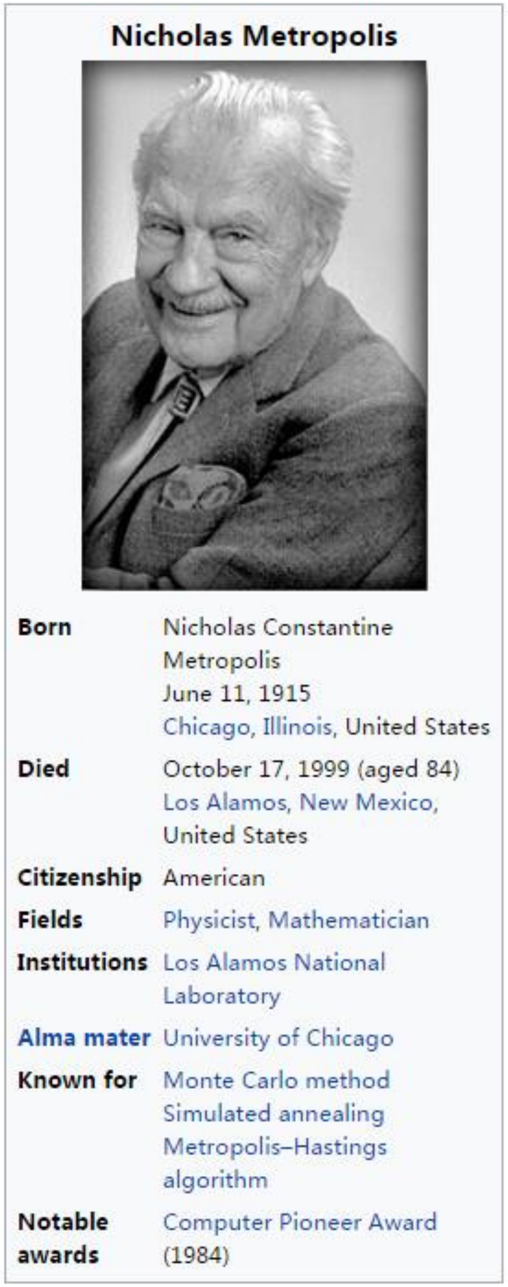
对区域D中任意内点O（其相邻节点1，2，3，4都在D内），



$$\phi_O=\frac{1}{4}(\phi_1+\phi_2+\phi_3+\phi_4)$$

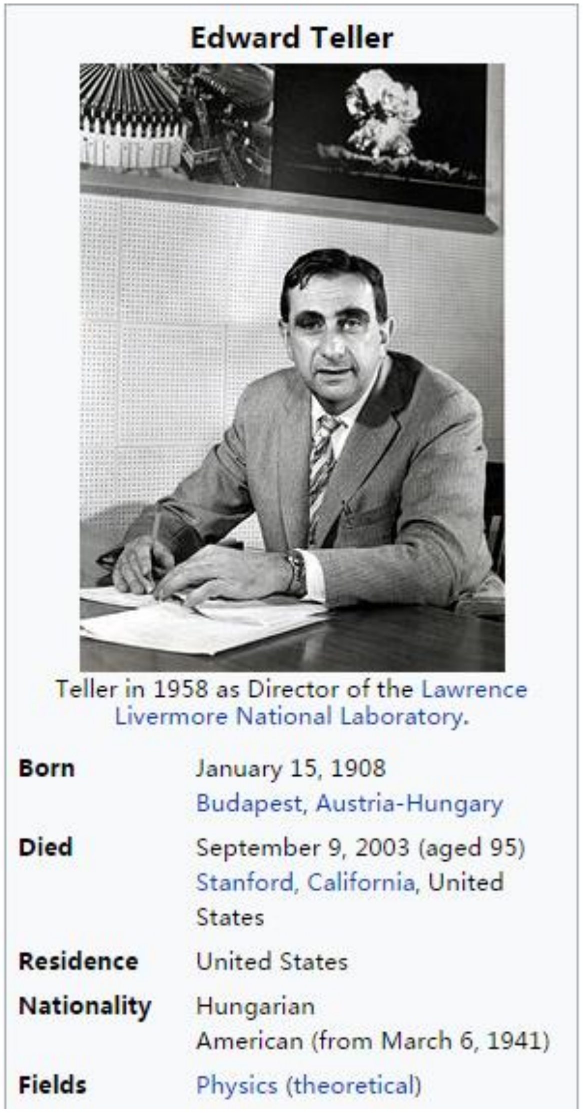
选取[0-1]均匀分布的随机数，根据其范围决定O游走到哪一个邻点。依次游走，直到边界。
反复从O点开始进行N次上述随机游走，得到N个ϕ_O的估计值，取平均。

Metropolis 方法



Nicholas Metropolis

Born	Nicholas Constantine Metropolis June 11, 1915 Chicago, Illinois, United States
Died	October 17, 1999 (aged 84) Los Alamos, New Mexico, United States
Citizenship	American
Fields	Physicist, Mathematician
Institutions	Los Alamos National Laboratory
Alma mater	University of Chicago
Known for	Monte Carlo method Simulated annealing Metropolis–Hastings algorithm
Notable awards	Computer Pioneer Award (1984)



Edward Teller

Teller in 1958 as Director of the Lawrence Livermore National Laboratory.

Born	January 15, 1908 Budapest, Austria-Hungary
Died	September 9, 2003 (aged 95) Stanford, California, United States
Residence	United States
Nationality	Hungarian American (from March 6, 1941)
Fields	Physics (theoretical)

Instead of choosing configurations randomly, then weighting them with exp(−E/kT), we choose configurations with a probability exp(−E/kT) and weight them evenly.

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

I. INTRODUCTION

THE purpose of this paper is to describe a general method, suitable for fast electronic computing machines, of calculating the properties of any substance which may be considered as composed of interacting individual molecules. Classical statistics is assumed, only two-body forces are considered, and the potential

II. THE GENERAL METHOD FOR AN ARBITRARY POTENTIAL BETWEEN THE PARTICLES

In order to reduce the problem to a feasible size for numerical work, we can, of course, consider only a finite number of particles. This number N may be as high as several hundred. Our system consists of a square† containing N particles. In order to minimize the surface

Metropolis 方法

In most situations, the transition probability $W_{ij} = W(j \rightarrow i)$ is not known . It can represent a complicated set of chemical reactions which we are not capable of modeling or, we are able to write down and account for all the boundary and the initial conditions needed to describe $W(j \rightarrow i)$. A Markov chain is a process where this probability is in general unknown. The question then is how can we model anything under such a severe lack of knowledge? The Metropolis algorithm comes to our rescue here. Since $W(j \rightarrow i)$ is unknown, we model it as the product of two probabilities, a probability for accepting the proposed move from the state j to the state j, and a probability for making the transition to the state i being in the state j. We label these probabilities $A(j \rightarrow i)$ and $T(j \rightarrow i)$, respectively. Our total transition probability is then

$$W(j \rightarrow i) = T(j \rightarrow i)A(j \rightarrow i).$$

- We make a suggested move to the new state i with some transition or moving probability $T_{j \rightarrow i}$.
- We accept this move to the new state with an acceptance probability $A_{j \rightarrow i}$. The new state i is in turn used as our new starting point for the next move. We reject this proposed moved with a $1 - A_{j \rightarrow i}$ and the original state j is used again as a sample.

$$w_i(t+1) = \sum_j \left[w_j(t) T_{j\to i} A_{j\to i} + w_i(t) T_{i\to j} (1 - A_{i\to j}) \right] .$$

$$\textstyle\sum_j T_{i\to j} = 1 .$$

$\Rightarrow$

$$w_i(t+1) = w_i(t) + \sum_j \left[w_j(t) T_{j\to i} A_{j\to i} - w_i(t) T_{i\to j} A_{i\to j} \right] ,$$

$$\frac{dw_i(t)}{dt} = \sum_j \left[W(j \to i) w_j - W(i \to j) w_i \right] ,$$

平衡条件 $\dfrac{dw_i(t)}{dt} = 0.$ $\Rightarrow$ $\displaystyle\sum_j w_j T_{j\to i} A_{j\to i} = \sum_j w_i T_{i\to j} A_{i\to j},$

$$\textstyle\sum_j W_{i\to j} = 1$$

$\Rightarrow$ $\displaystyle w_i = \sum_j w_j T_{j\to i} A_{j\to i} = \sum_j w_j W_{j\to i},$

细致平衡

However, the condition that the rates should equal each other is in general not sufficient to guarantee that we, after many simulations, generate the correct distribution. We may risk to end up with so-called cyclic solutions. To avoid this we therefore introduce an additional condition, namely that of detailed balance

$$W(j \to i)w_j = W(i \to j)w_i.$$

These equations were derived by Lars Onsager when studying irreversible processes

At equilibrium detailed balance gives thus

$$\frac{W(j \to i)}{W(i \to j)} = \frac{w_i}{w_j}.$$

$$\Rightarrow \quad \frac{T_{j \to i} A_{j \to i}}{T_{i \to j} A_{i \to j}} = \frac{w_i}{w_j}.$$

玻尔兹曼分布，配分函数

We introduce the Boltzmann distribution

$$w_i = \frac{\exp\left(-\beta(E_i)\right)}{Z},$$

which states that probability of finding the system in a state i with energy E_i at an inverse temperature $\beta = 1/k_B T$ is $w_i \propto \exp\left(-\beta(E_i)\right)$. The denominator Z is a normalization constant which ensures that the sum of all probabilities is normalized to one. It is defined as the sum of probabilities over all microstates j of the system

$$Z = \sum_j \exp\left(-\beta(E_i)\right).$$

From the partition function we can in principle generate all interesting quantities for a given system in equilibrium with its surroundings at a temperature T.

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array} \qquad \boldsymbol{E_i}$$

A single spinflip

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array} \qquad \boldsymbol{E_j}$$

玻尔兹曼分布，配分函数

With the probability distribution given by the Boltzmann distribution we are now in the position where we can generate expectation values for a given variable A through the definition

$$\langle A\rangle = \sum_j A_j w_j = \frac{\sum_j A_j \exp\left(-\beta(E_j\right)}{Z}.$$

In general, most systems have an infinity of microstates making thereby the computation of Z practically impossible and a brute force Monte Carlo calculation over a given number of randomly selected microstates may therefore not yield those microstates which are important at equilibrium. To select the most important contributions we need to use the condition for detailed balance. Since this is just given by the ratios of probabilities, we never need to evaluate the partition function Z. For the Boltzmann distribution, detailed balance results in

$$\frac{w_i}{w_j} = \exp\left(-\beta(E_i - E_j)\right).$$

Metropolis 模拟

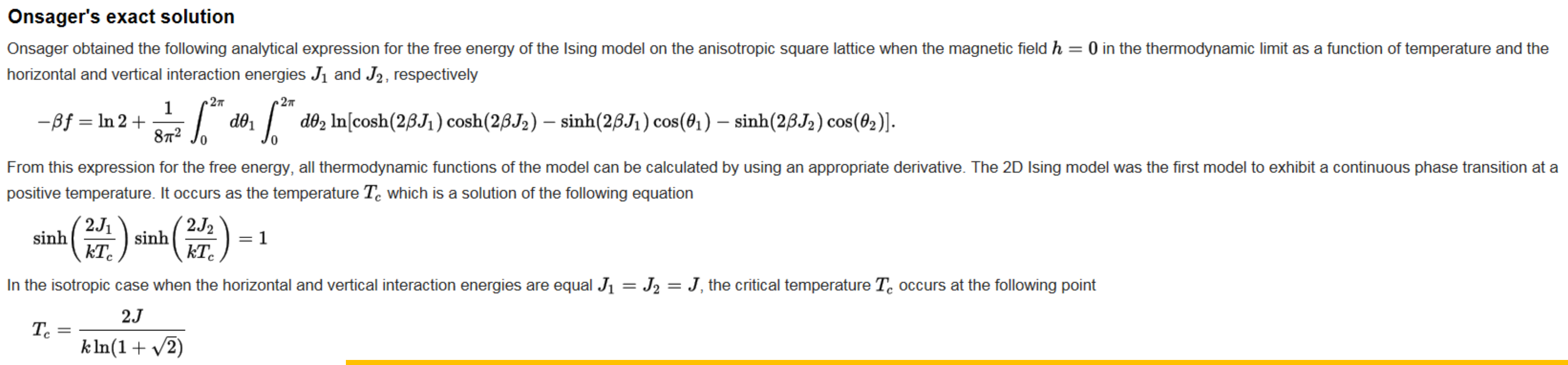
$$\frac{P(A)}{P(B)} = \frac{e^{-E_A/T}}{e^{-E_B/T}} = e^{-(E_A-E_B)/T}$$

1. Starting from a configuration A, with known energy E_A, make a change in the configuration to obtain a new (nearby) configuration B.
2. Compute E_B (typically as a small change from E_A.
3. If $E_B < E_A$, assume the new configuration, since it has lower energy (a desirable thing, according to the Boltzmann factor).
4. If $E_B > E_A$, accept the new (higher energy) configuration with probability $p = e^{-(E_B-E_A)/T}$. This means that when the temperature is high, we don't mind taking steps in the "wrong" direction, but as the temperature is lowered, we are forced to settle into the lowest configuration we can find in our neighborhood.

Ising模型

The Ising model was invented by the physicist Wilhelm Lenz (1920), who gave it as a problem to his student Ernst Ising. **The one-dimensional Ising model has no phase transition and was solved by Ising** (1925) himself in his 1924 thesis. The two-dimensional square lattice Ising model is much harder, and was given an analytic description much later, by Lars Onsager (1944). It is usually solved by a transfer-matrix method, although there exist different approaches, more related to quantum field theory.In dimensions greater than four, the phase transition of the Ising model is described by mean field theory.

$$H(\sigma) = -J\sum_{\langle ij\rangle}\sigma_i\sigma_j - h\sum_j \sigma_j.$$



Onsager's exact solution

Onsager obtained the following analytical expression for the free energy of the Ising model on the anisotropic square lattice when the magnetic field $h = 0$ in the thermodynamic limit as a function of temperature and the horizontal and vertical interaction energies J_1 and J_2, respectively

$$-\beta f = \ln 2 + \frac{1}{8\pi^2}\int_0^{2\pi} d\theta_1 \int_0^{2\pi} d\theta_2 \ln[\cosh(2\beta J_1)\cosh(2\beta J_2) - \sinh(2\beta J_1)\cos(\theta_1) - \sinh(2\beta J_2)\cos(\theta_2)].$$

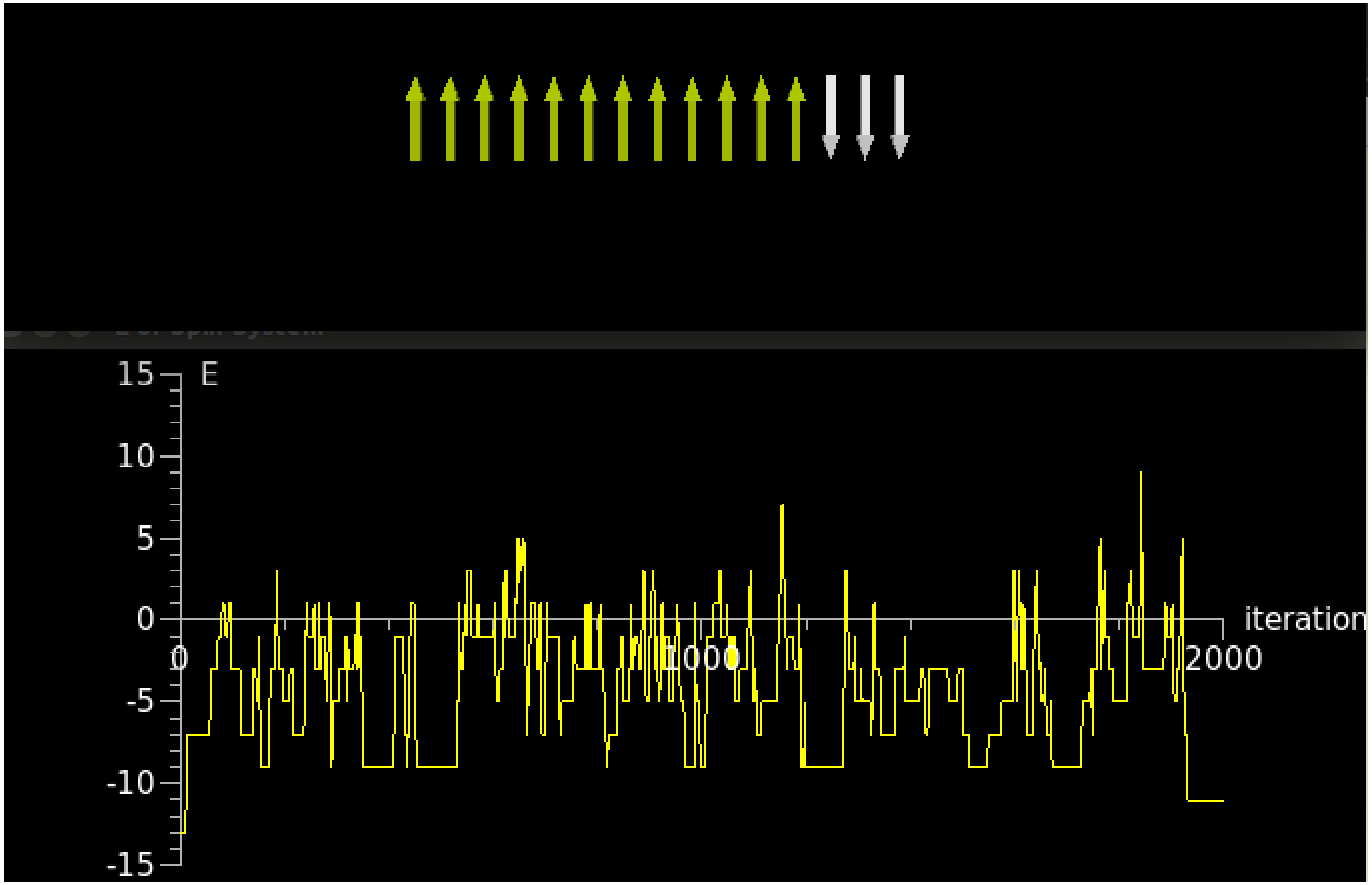
From this expression for the free energy, all thermodynamic functions of the model can be calculated by using an appropriate derivative. The 2D Ising model was the first model to exhibit a continuous phase transition at a positive temperature. It occurs as the temperature T_c which is a solution of the following equation

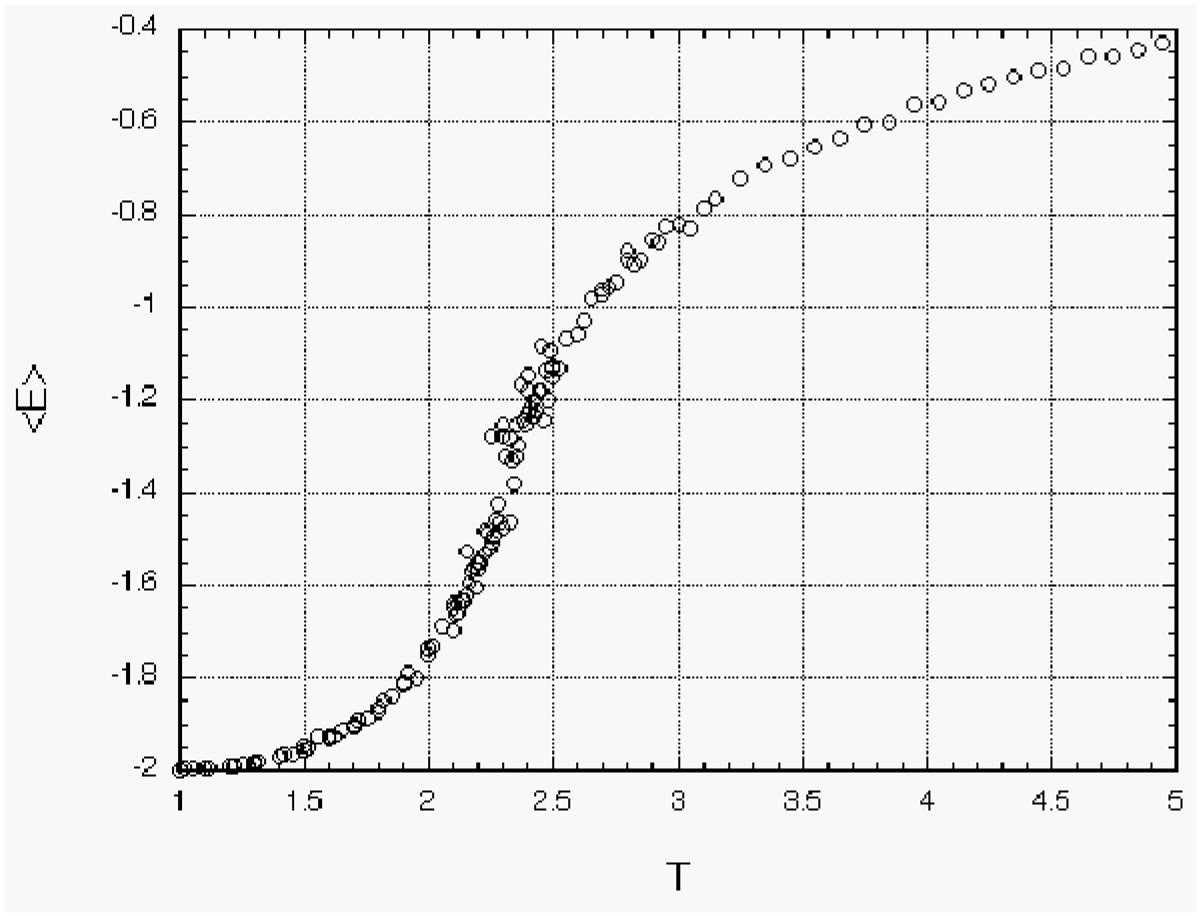
$$\sinh\left(\frac{2J_1}{kT_c}\right)\sinh\left(\frac{2J_2}{kT_c}\right) = 1$$

In the isotropic case when the horizontal and vertical interaction energies are equal $J_1 = J_2 = J$, the critical temperature T_c occurs at the following point

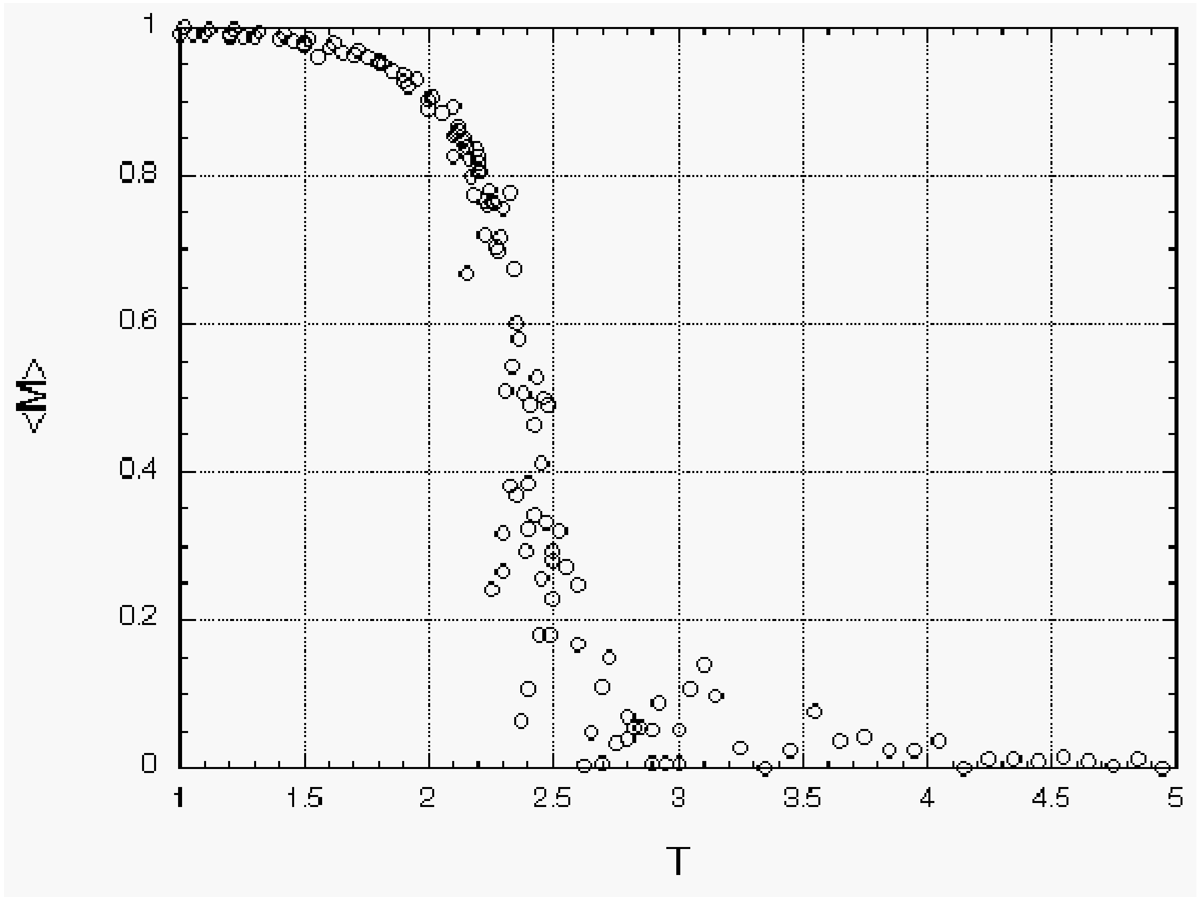
$$T_c = \frac{2J}{k\ln(1+\sqrt{2})}$$

$$T_c = 2/\ln(1+\sqrt{2}) \sim 2.27, \qquad for\ J = 1, k = 1$$


E
15
10
5
0
-5
-10
-15
0
1000
2000
iteration



The energy is a continuous function of temperature, which, as we expect, increases as a function of T.

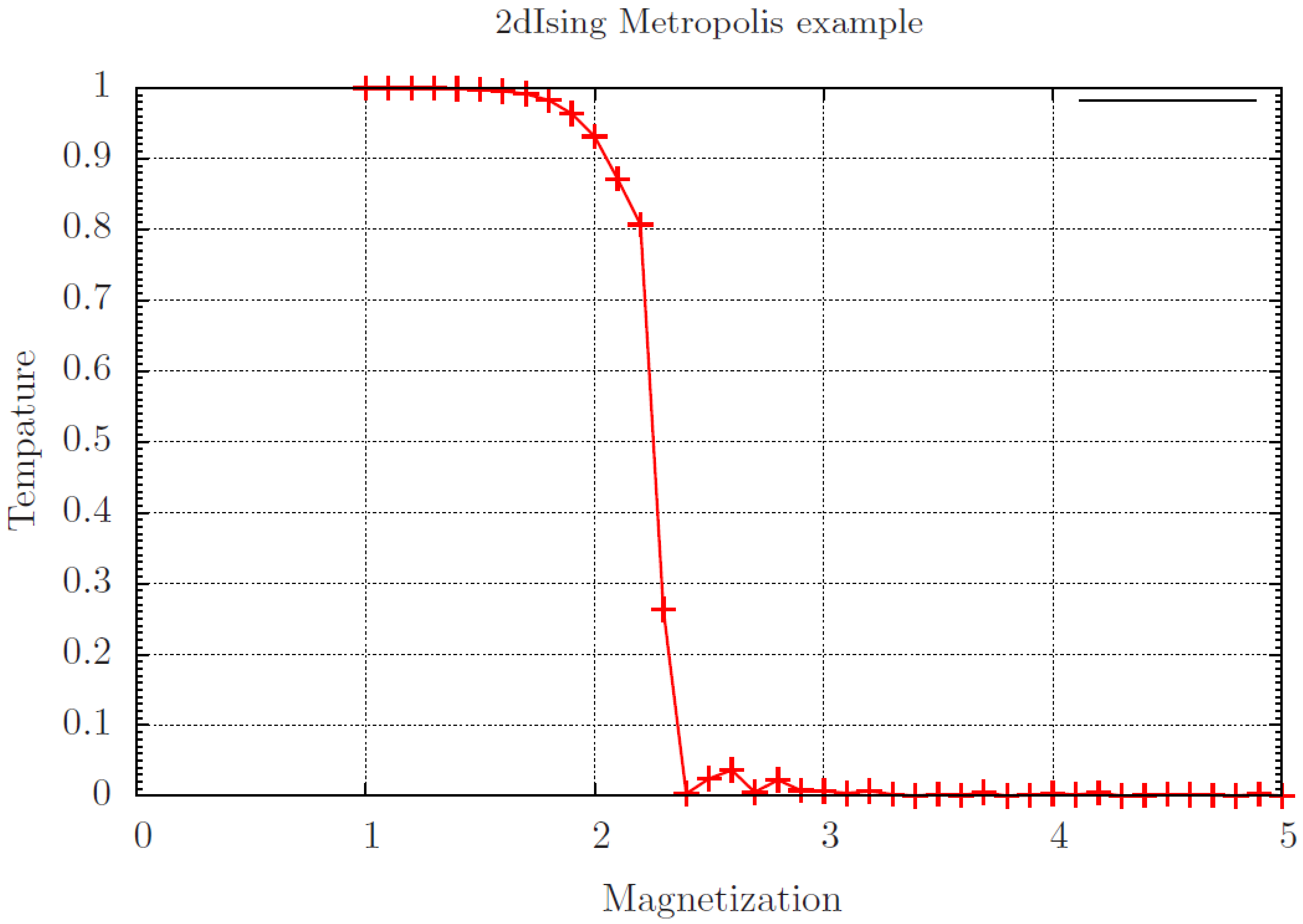


The magnetization drops off sharply near the critical temperature, which, in our units where $k = J = 1$, is approximately 2.3.

Ising模型：Metropolis 模拟

```
do ipass = 0, npass
    ! If ipass is greater than nequil (the number of
equilibration steps), calculate the magnetization and
energy:
    if (ipass > nequil) then
    output_count = output_count + 1
    magnetization =
sum(A(2:nrows+1,2:nrows+1))/(ncols*nrows*1.0)
    magnetization_ave = magnetization_ave +
magnetization
    energy = 0.0
    do i = 2, nrows + 1
    do j = 2, ncols + 1
    energy = energy - A(m,n)*(A(m-
1,n)+A(m+1,n)+A(m,n-1)+A(m,n+1))
    enddo
    enddo
    ! Divide the energy by the total number of
spins to get the ave
    ! energy per spin, and divide by 2 to account
for double counting.
    energy = energy/(ncols*nrows*2.0)
    energy_ave = energy_ave + energy
    endif
```

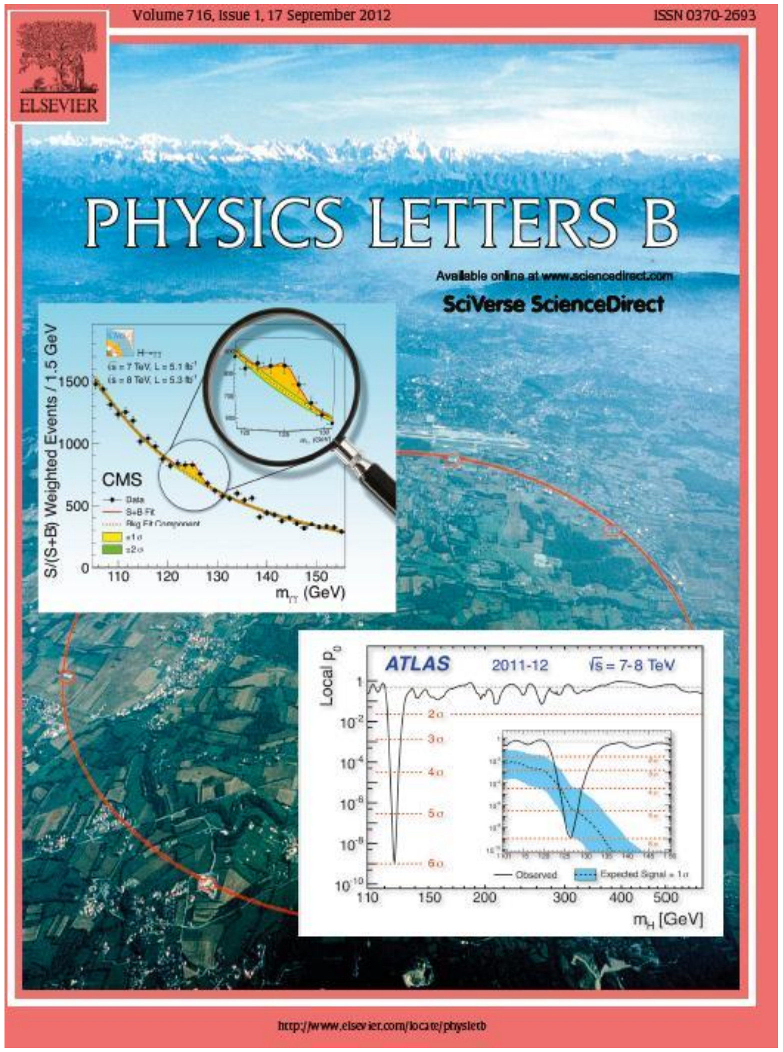
```
 ! Randomly choose a spin to change:
    m = nint((nrows-1)*ran1(5) + 2) ! choose a
random row
    n = nint((ncols-1)*ran1(5) + 2) ! choose a
random column
    trial_spin = -A(m,n) ! trial spin value
    ! Find change in energy (deltaU) due to trial
move.
    ! If exp(-beta*deltaU) > eta, where eta is
random, accept move:
    deltaU = -trial_spin*(A(m-
1,n)+A(m+1,n)+A(m,n-1)+A(m,n+1))*2
    log_eta = dlog(ran1(5) + 1.0d-10) ! random
number 0-1 (+ tiny offset)
    if (-beta*deltaU > log_eta) then
    A(m,n) = trial_spin
    if (m == 2) A(nrows+2,n) = trial_spin
    if (m == nrows+1) A(1,n) = trial_spin
    if (n == 2) A(m,ncols+2) = trial_spin
    if (n == ncols+1) A(m,1) = trial_spin
    endif
    enddo MC_passes
```



2dIsing Metropolis example
Tempature
Magnetization
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5

补充参考

高能物理： 对撞机事例生成器



The Nobel Prize in Physics 2013

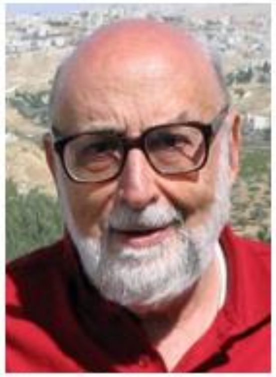


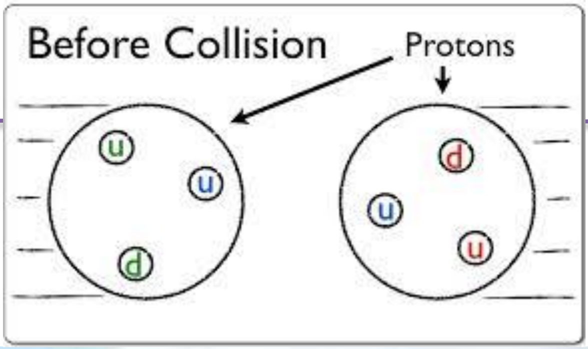
Photo: Pnicolet via Wikimedia Commons
François Englert

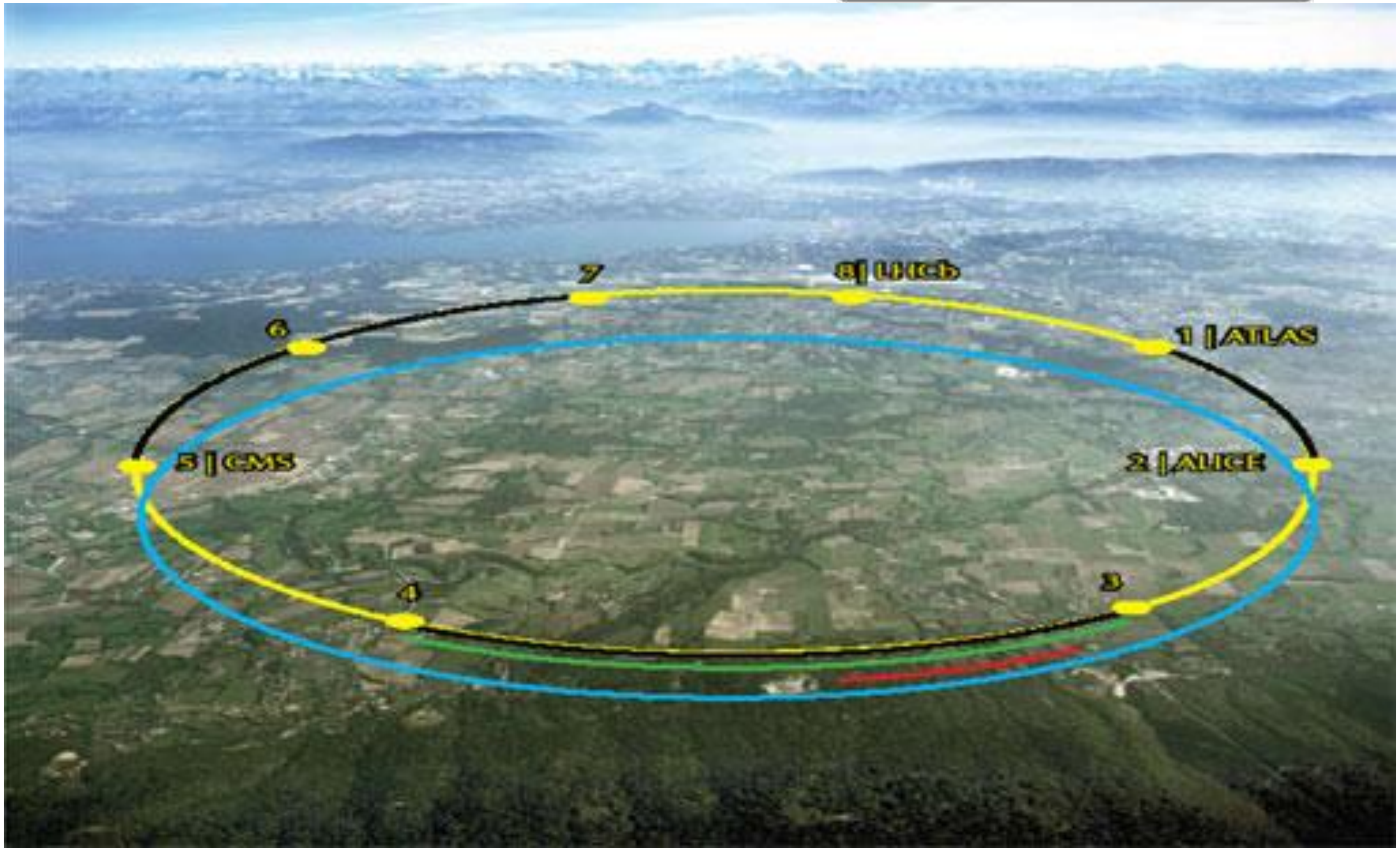
Photo: G-M Greuel via Wikimedia Commons
Peter W. Higgs

The Nobel Prize in Physics 2013 was awarded jointly to François Englert and Peter W. Higgs *"for the theoretical discovery of a mechanism that contributes to our understanding of the origin of mass of subatomic particles, and which recently was confirmed through the discovery of the predicted fundamental particle, by the ATLAS and CMS experiments at CERN's Large Hadron Collider"*

Collider

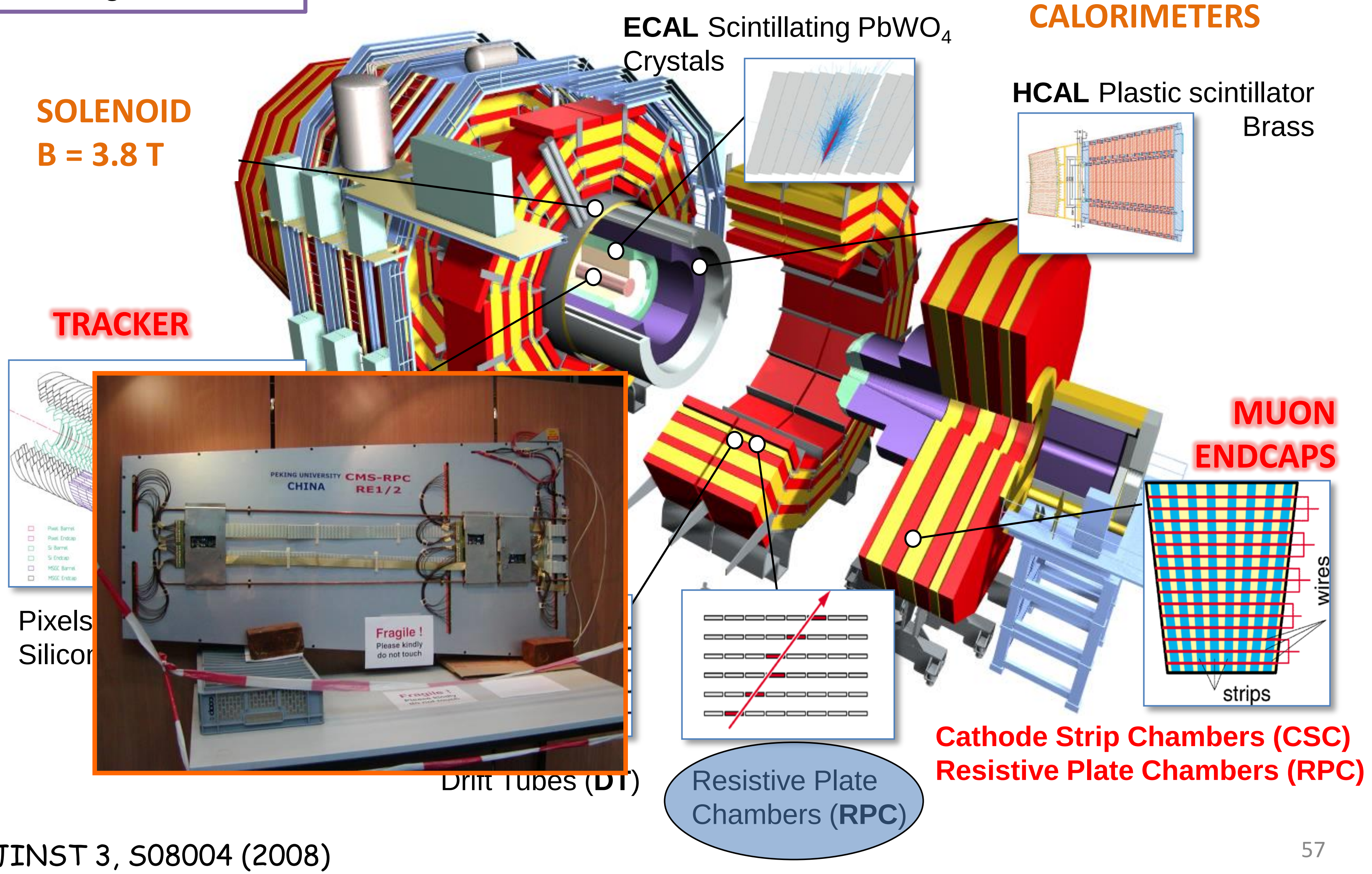
$$1fm \sim 5GeV^{-1}$$



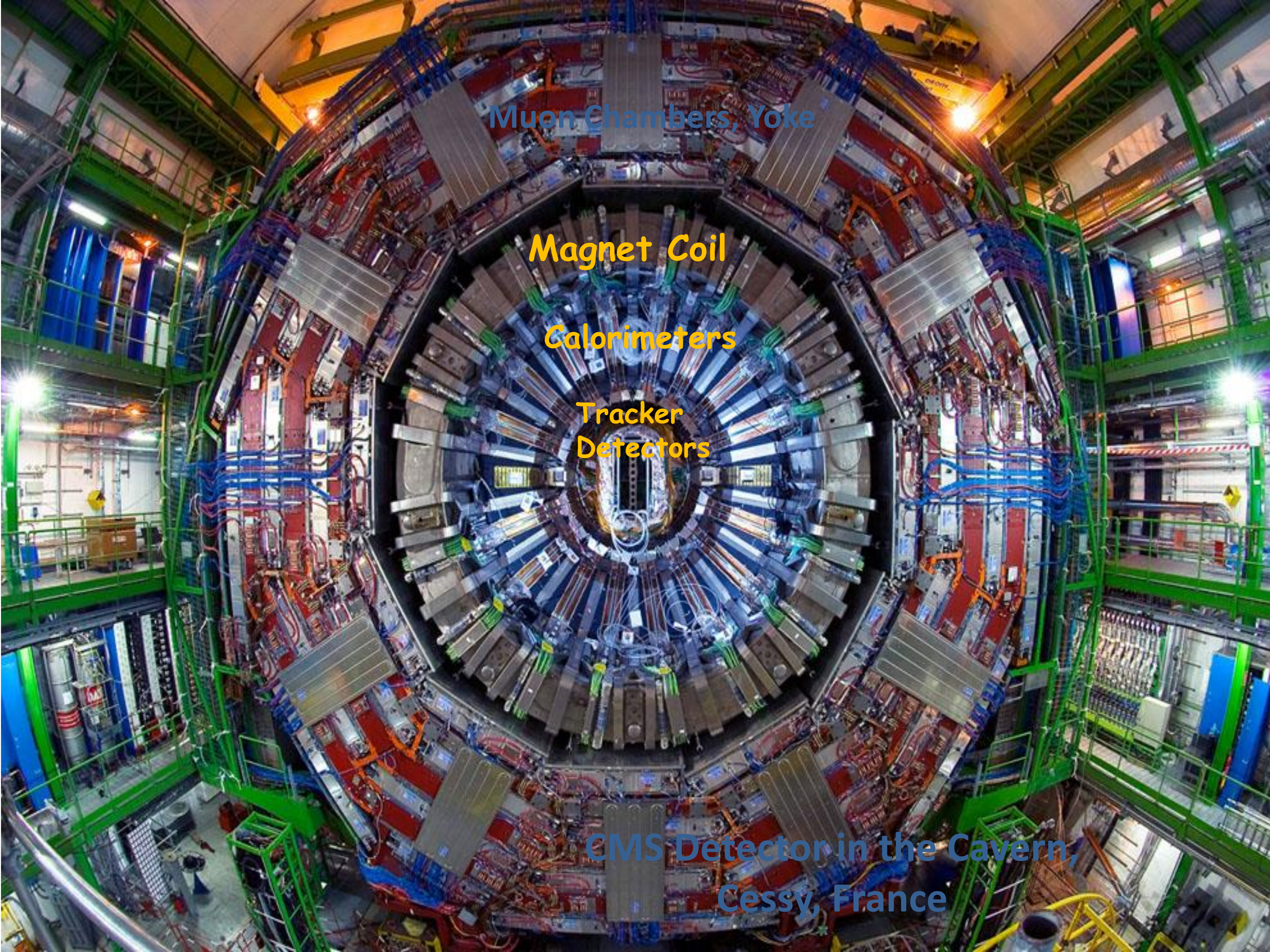


CMS Detector

weight: 12500 t
overall diameter: 15 m
overall length: 21.6 m



CALORIMETERS
ECAL Scintillating $PbWO_4$ Crystals
HCAL Plastic scintillator Brass
SOLENOID
B = 3.8 T
TRACKER
Pixel Barrel
Pixel Endcap
Si Barrel
Si Endcap
MSGC Barrel
MSGC Endcap
Pixels
Silicon
Drift Tubes (**DT**)
Resistive Plate Chambers (**RPC**)
MUON ENDCAPS
wires
strips
Cathode Strip Chambers (CSC)
Resistive Plate Chambers (RPC)



CMS Detector in the Cavern, Cessy, France

How to search for a Higgs particle?

Not so easy!

Needles in a haystack

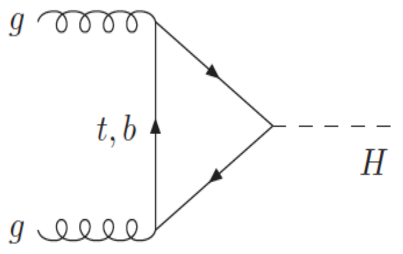
In ATLAS, up to July 4, 2012:

- **A million billion** collisions
- **4.2 billion** events analyzed
- **240,000** Higgs particles produced
- ~**350** diphoton Higgs events detected
- ~**8** four-lepton Higgs events detected

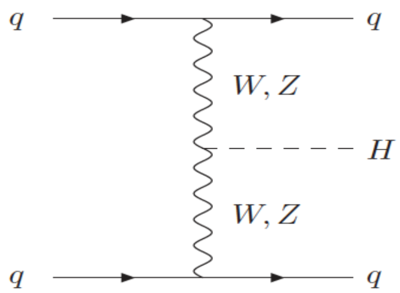
~540k
~900
~21

Main production processes

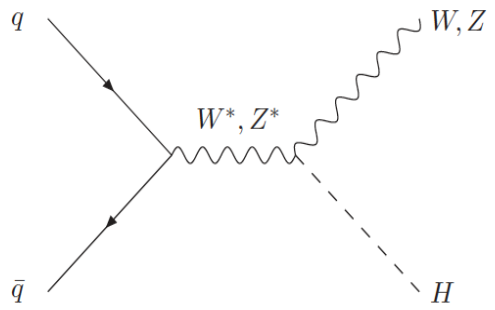
ggH



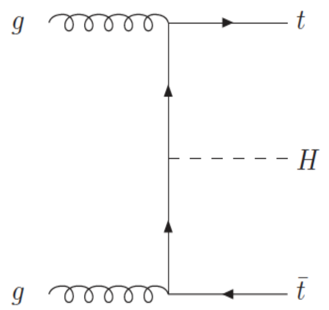
VBF

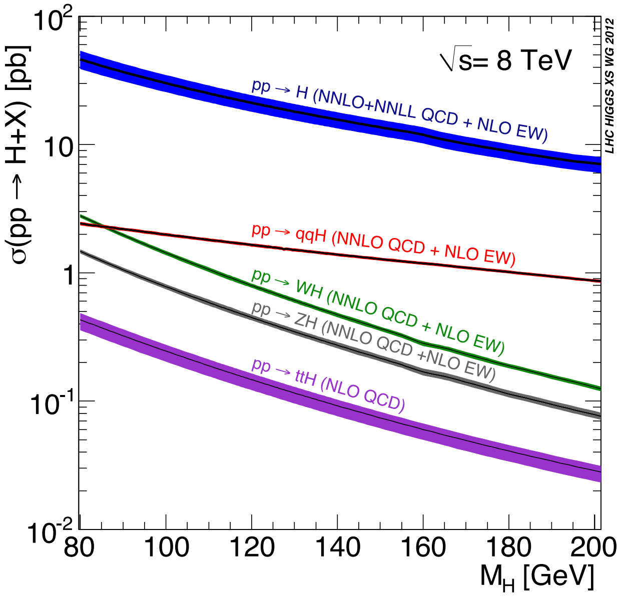


VH



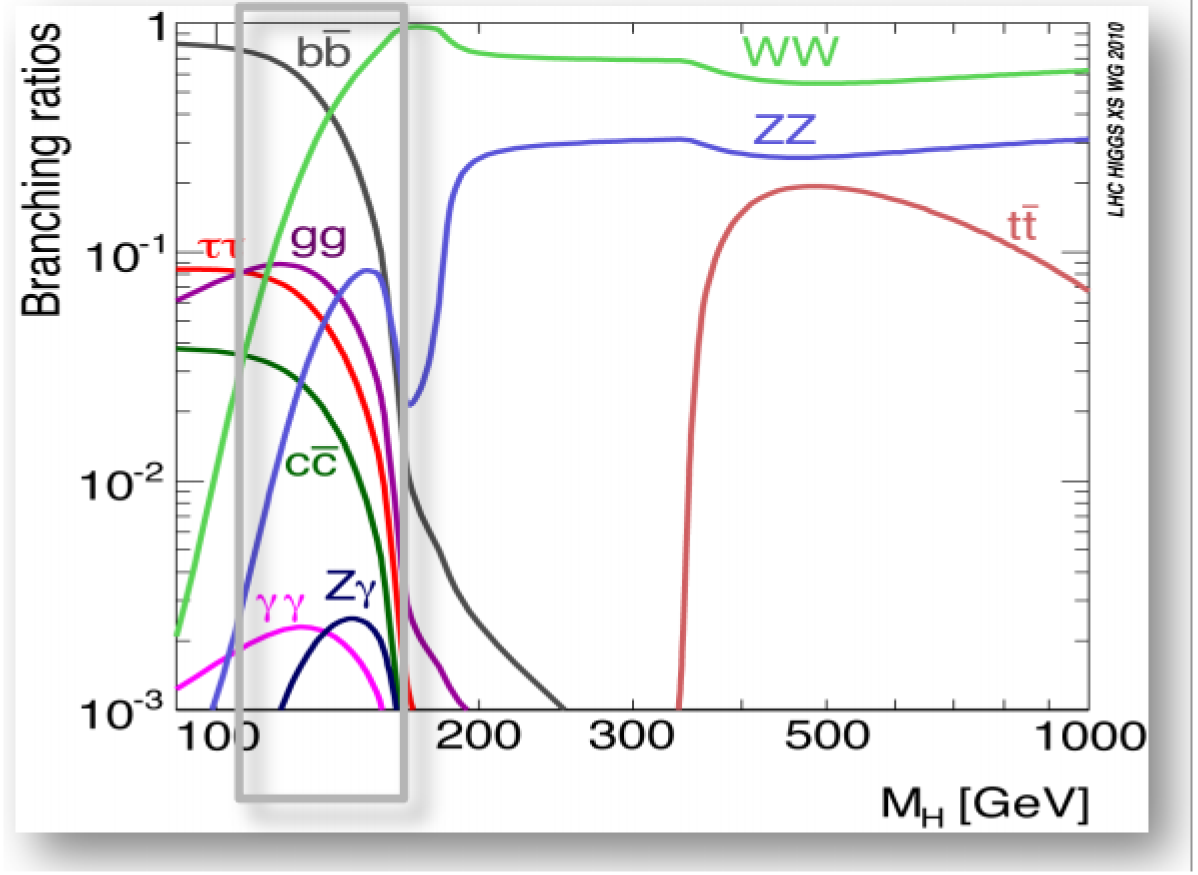
ttH





Production process	Cross section [pb] $\sqrt{s}$ = 7 TeV	Cross section [pb] $\sqrt{s}$ = 8 TeV	Order of calculation
ggF	15.0 ± 1.6	19.2 ± 2.0	NNLO(QCD)+NLO(EW)
VBF	1.22 ± 0.03	1.58 ± 0.04	NLO(QCD+EW)+APP.NNLO(QCD)
WH	0.577 ± 0.016	0.703 ± 0.018	NNLO(QCD)+NLO(EW)
ZH	0.357 ± 0.015	0.446 ± 0.019	NNLO(QCD)+NLO(EW)
ZH: $gg \to ZH$			LO(QCD)
bbH	0.156 ± 0.021	0.203 ± 0.028	5FS NLO(QCD) + 4FS NLO(QCD)
ttH	0.086 ± 0.009	0.129 ± 0.014	NLO(QCD)
tH	0.012 ± 0.001	0.018 ± 0.001	NLO(QCD)
Total	17.4 ± 1.6	22.3 ± 2.0	

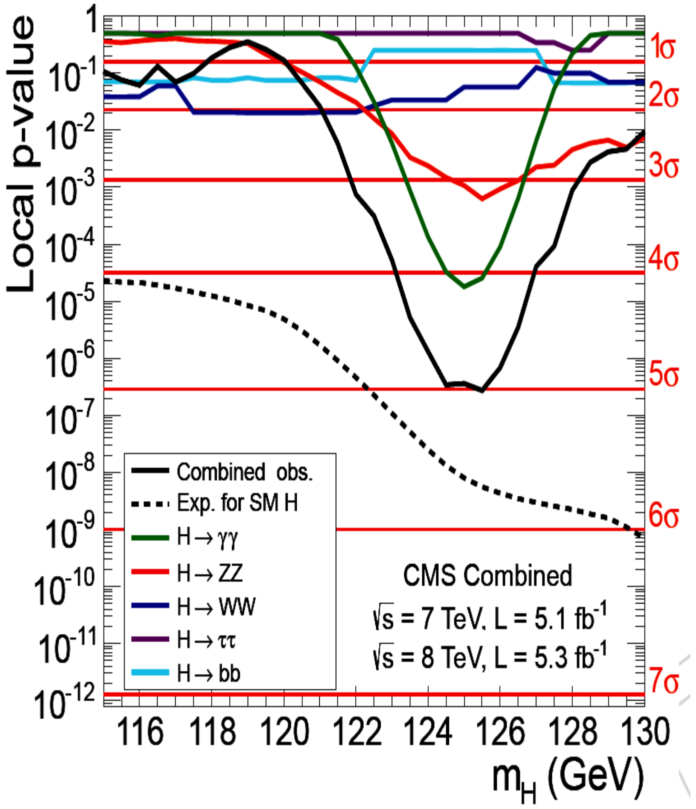
SM ggF, ttH, bbH theory uncertainty: ~10%
VBF, VH, ZH: 2-3%

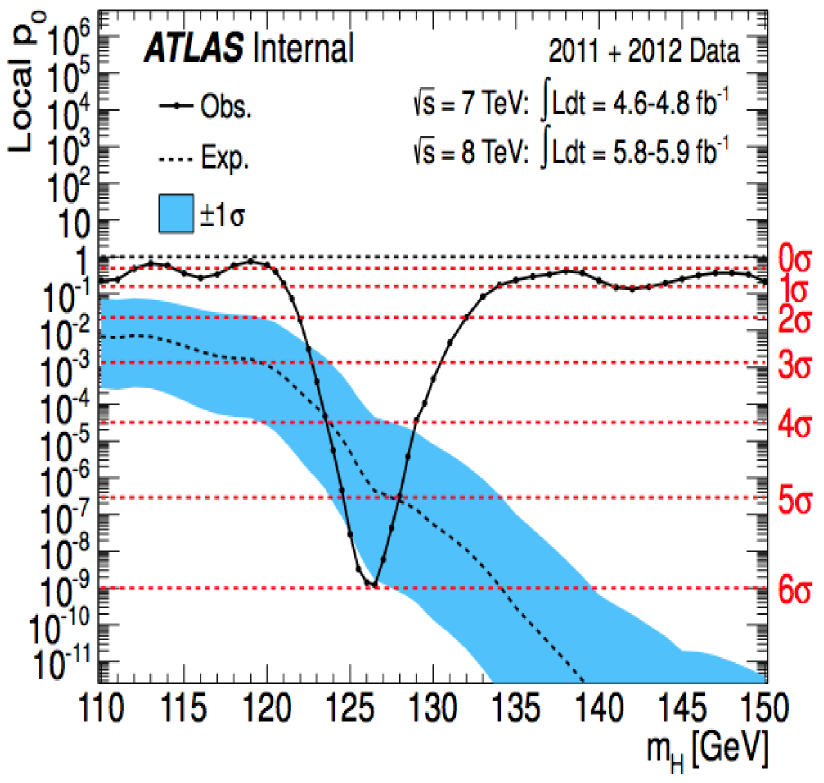


Decay / Production	Untagged	VBF	VH	ttH
H→γγ	🟩	🟩	🟩	🟩
H→ZZ→4l	🟩	🟩	🟩	🟩
H→WW→2l2ν	🟩	🟩	🟩	🟩
H→ττ	🟩	🟩	🟩	🟩
H→bb			🟩	🟩
H→μμ	🟩	🟩		

July 4th , 2012

Discovery of a new boson.



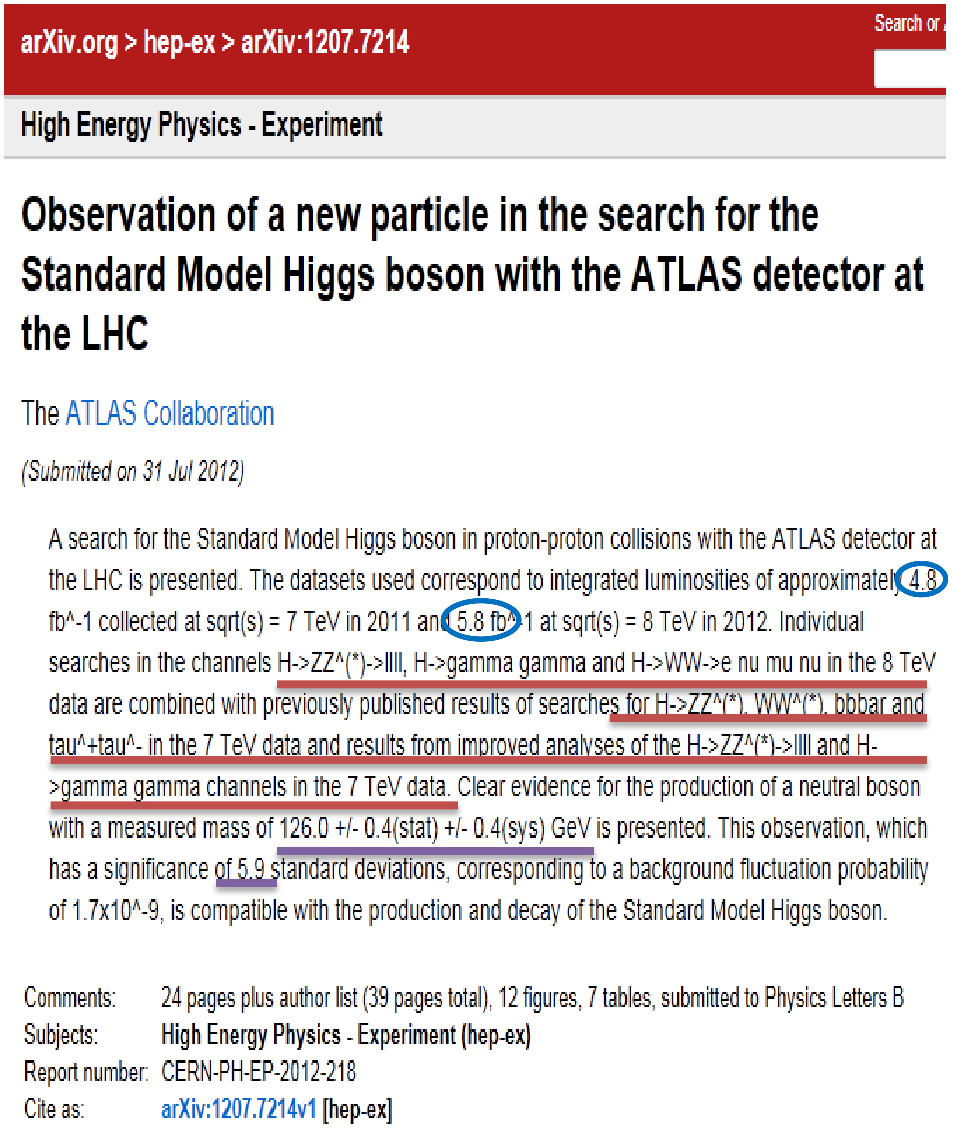


Combined significance 5.0σ for CMS and 5.9σ for ATLAS

125.3+-0.4+-0.5GeV
0.87+-0.23

126.0+-0.4+-0.4GeV
1.4+-0.3

2012.07 Big Discovery



arXiv.org > hep-ex > arXiv:1207.7214

Search or

High Energy Physics - Experiment

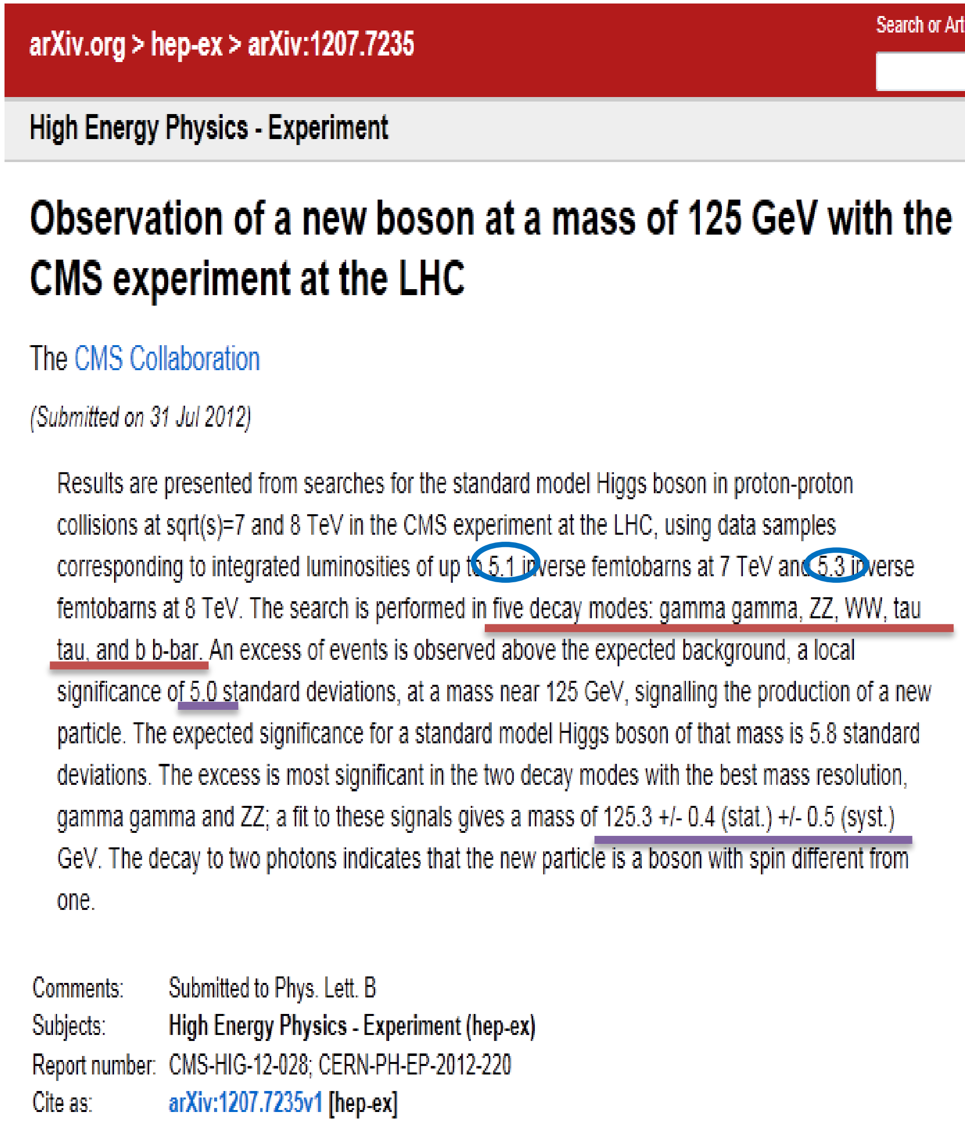
Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC

The ATLAS Collaboration

(Submitted on 31 Jul 2012)

A search for the Standard Model Higgs boson in proton-proton collisions with the ATLAS detector at the LHC is presented. The datasets used correspond to integrated luminosities of approximately 4.8 fb^-1 collected at sqrt(s) = 7 TeV in 2011 and 5.8 fb^-1 at sqrt(s) = 8 TeV in 2012. Individual searches in the channels H->ZZ^(*)->llll, H->gamma gamma and H->WW->e nu mu nu in the 8 TeV data are combined with previously published results of searches for H->ZZ^(*), WW^(*), bbbar and tau^+tau^- in the 7 TeV data and results from improved analyses of the H->ZZ^(*)->llll and H->gamma gamma channels in the 7 TeV data. Clear evidence for the production of a neutral boson with a measured mass of 126.0 +/- 0.4(stat) +/- 0.4(sys) GeV is presented. This observation, which has a significance of 5.9 standard deviations, corresponding to a background fluctuation probability of 1.7x10^-9, is compatible with the production and decay of the Standard Model Higgs boson.

Comments:	24 pages plus author list (39 pages total), 12 figures, 7 tables, submitted to Physics Letters B
Subjects:	High Energy Physics - Experiment (hep-ex)
Report number:	CERN-PH-EP-2012-218
Cite as:	arXiv:1207.7214v1 [hep-ex]



arXiv.org > hep-ex > arXiv:1207.7235

Search or Art

High Energy Physics - Experiment

Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC

The CMS Collaboration

(Submitted on 31 Jul 2012)

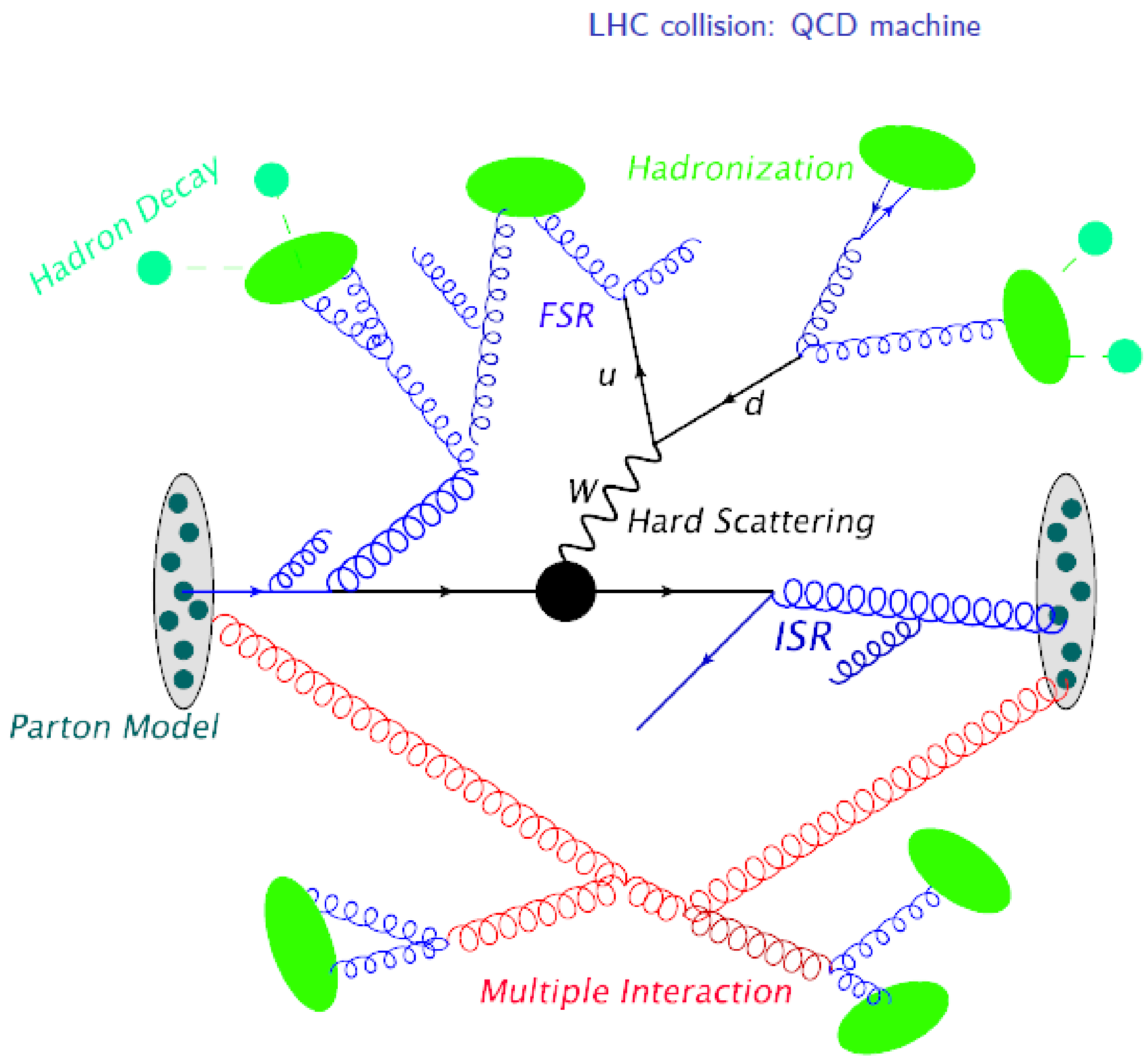
Results are presented from searches for the standard model Higgs boson in proton-proton collisions at sqrt(s)=7 and 8 TeV in the CMS experiment at the LHC, using data samples corresponding to integrated luminosities of up to 5.1 inverse femtobarns at 7 TeV and 5.3 inverse femtobarns at 8 TeV. The search is performed in five decay modes: gamma gamma, ZZ, WW, tau tau, and b b-bar. An excess of events is observed above the expected background, a local significance of 5.0 standard deviations, at a mass near 125 GeV, signalling the production of a new particle. The expected significance for a standard model Higgs boson of that mass is 5.8 standard deviations. The excess is most significant in the two decay modes with the best mass resolution, gamma gamma and ZZ; a fit to these signals gives a mass of 125.3 +/- 0.4 (stat.) +/- 0.5 (syst.) GeV. The decay to two photons indicates that the new particle is a boson with spin different from one.

Comments:	Submitted to Phys. Lett. B
Subjects:	High Energy Physics - Experiment (hep-ex)
Report number:	CMS-HIG-12-028; CERN-PH-EP-2012-220
Cite as:	arXiv:1207.7235v1 [hep-ex]

5.9sigma
Phys.Lett. B716 (2012) 1-29

5.0sigma
Phys. Lett. B 716 (2012) 30

Anatomy of a LHC Collision



Factorization Theorem:
Separate Short Distance Physics from Soft one

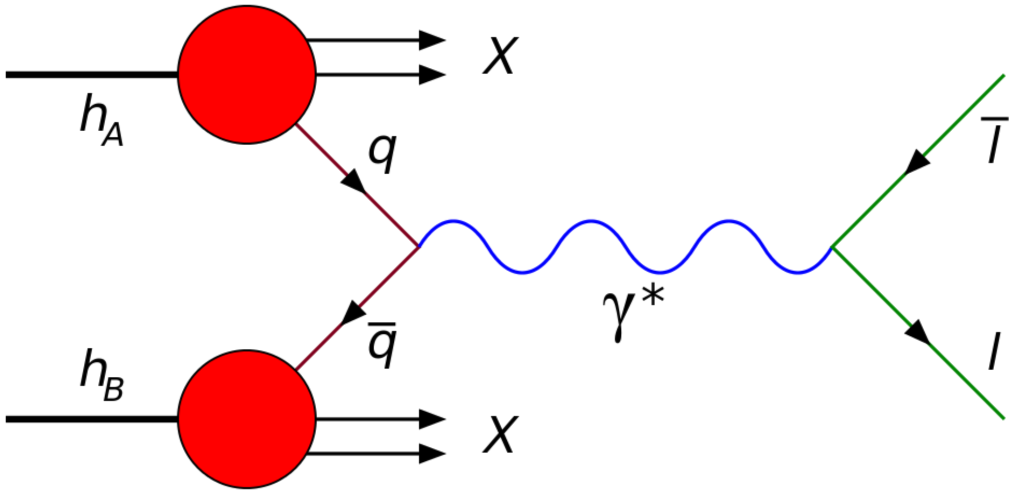
QCD Machine

Factorization

Multi-level

Hard Scattering:

Hard Scattering:
LO, NLO, NNLO QCD, QED..



LO: Born term

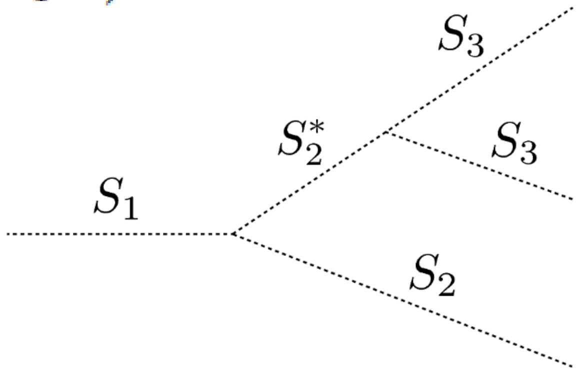
$$d\sigma_{h_1h_2} = \sum_{i,j}\int_0^1 dx_i \int_0^1 dx_j \sum_f \int d\Phi_f \, f_{i/h_1}(x_i,\mu_F^2)\, f_{j/h_2}(x_j,\mu_F^2)\, \frac{d\hat{\sigma}_{ij\to f}}{dx_i\, dx_j\, d\Phi_f}$$

Factorization scale	μ_F
Renormalization Scale	μ_r
Phase Space	$d\Phi_f$

Hard Scattering: Phase Space

$$
\begin{aligned}
d\Phi_n(P, p_1, ..., p_n) &= \prod_{i=1}^{n} \frac{d^4 p_i}{(2\pi)^3} \Theta(p_i^0) \delta(p_i^2 - m_i^2)(2\pi)^4 \delta^4 \left(P - \sum_{i=1}^{n} p_i\right) \\
&= \prod_{i=1}^{n} \frac{d^3 p_i}{(2\pi)^3 2E_i} (2\pi)^4 \delta^4 \left(P - \sum_{i=1}^{n} p_i\right).
\end{aligned}
$$

3n-4+2 =3n-2 dimension



An example of Phase space factorization

-> Recursive in numerical

$$
d\Phi_n(P, p_1, ..., p_n) = \frac{1}{2\pi} dQ^2 d\Phi_j(Q, p_1, ..., p_j) d\Phi_{n-j+1}(P, Q, p_{j+1}, ..., p_n).
$$

MC Technique

$$I \approx I_N \pm \sqrt{\frac{V_N}{N}}$$

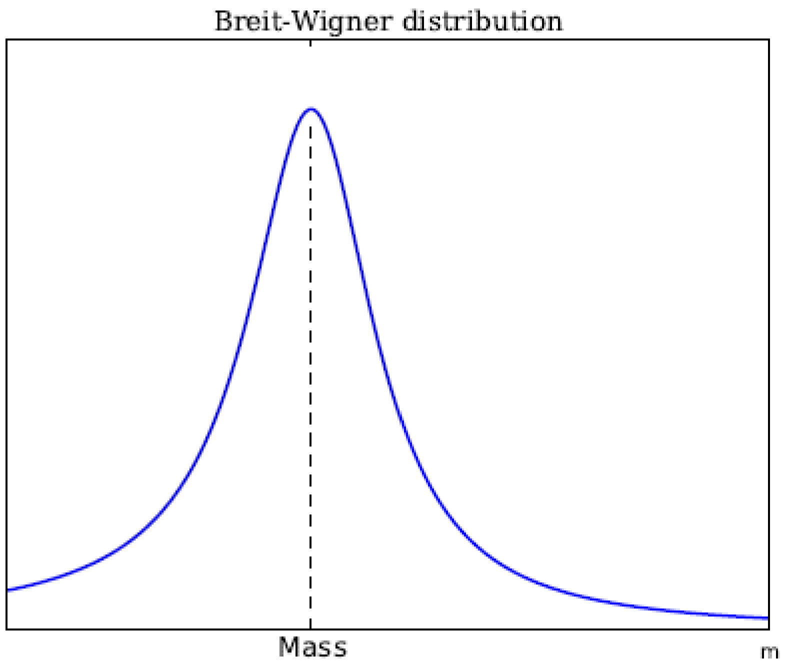
- **Good convergence for high dimension integrals**
- **We also got events randomly distributed**
- $\boldsymbol{V_N}$ **should be small: importance sampling**

$$I = \int_{M^2_{\min}}^{M^2_{\max}} \mathrm{d}m^2 \frac{1}{(m^2 - M^2)^2 + M^2\Gamma^2}$$

$$m^2 = M\Gamma \tan\rho + M^2$$

$$\longrightarrow$$

$$I = \int_{\rho_{\min}}^{\rho_{max}} \mathrm{d}\rho \left|\frac{\partial m^2}{\partial \rho}\right| \frac{1}{(m^2 - M^2)^2 + M^2\Gamma^2}$$
$$= \frac{1}{M\Gamma}\int_{\rho_{\min}}^{\rho_{max}} \mathrm{d}\rho\,.$$



Unweighting

We often want events without weights as mother Nature produce

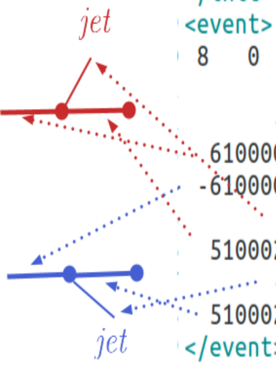
1. Monte Carlo integration and scanning are performed:
   N points are picked randomly

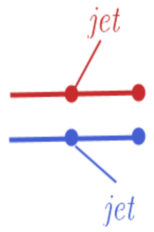
2. The phase-space point which give the maximum weight, Wmax is stored

3. 'hit-or-miss': go through randomly chosen phase-space points and compare the probability of each, given by Wi/Wmax to a random number R in (0, 1).
   If Wi/Wmax > R, we 'accept' the event, otherwise wereject it. This is done until we have collected the desired number of events, Nevents.

Les Houches Event File hep-ph/0609017

```
<LesHouchesEvents version="1.0">
<header>
#Additional information
</header>
<init>
     2212     2212  0.40000000000E+04  0.40000000000E+04 0 0 10042 10042 3  1
  0.13448000000E+02  0.11328000000E+00  0.26896000000E+01   0
</init>
<event>
 8    0   0.2689600E+01   0.1000000E+04   0.7957747E-01   0.9421117E-01
        2    -1     0     0   501     0  0.00000000000E+00  0.00000000000E+00  0.12216473395E+04  0.12216473395E+04  0.30000000261E-02 0.  1.
       -2    -1     0     0     0   501  0.00000000000E+00  0.00000000000E+00 -0.95840193959E+03  0.95840193960E+03  0.30000000261E-02 0. -1.
  6100002     2     1     2   502     0  0.12085632485E+03 -0.21778312976E+03  0.82072277461E+03  0.11732307109E+04  0.80000000000E+03 0.  0.
 -6100002     2     1     2     0   502 -0.12085632485E+03  0.21778312976E+03 -0.55747737471E+03  0.10068185682E+04  0.80000000000E+03 0.  0.
        2     1     3     3   502     0 -0.84181441025E+02 -0.27383300132E+03  0.36569663377E+03  0.46454822740E+03  0.30000000261E-02 0.  1.
  5100022     1     3     3     0     0  0.20503776588E+03  0.56049871558E+02  0.45502614084E+03  0.70868248348E+03  0.50000000000E+03 0.  1.
       -2     1     4     4     0   502  0.10854022679E+03  0.26478799687E+03 -0.18273879961E+03  0.33953958975E+03  0.30000000261E-02 0. -1.
  5100022     1     4     4     0     0 -0.22939655164E+03 -0.47004867115E+02 -0.37473857510E+03  0.66727897847E+03  0.50000000000E+03 0.  1.
</event>
<event>
...
```



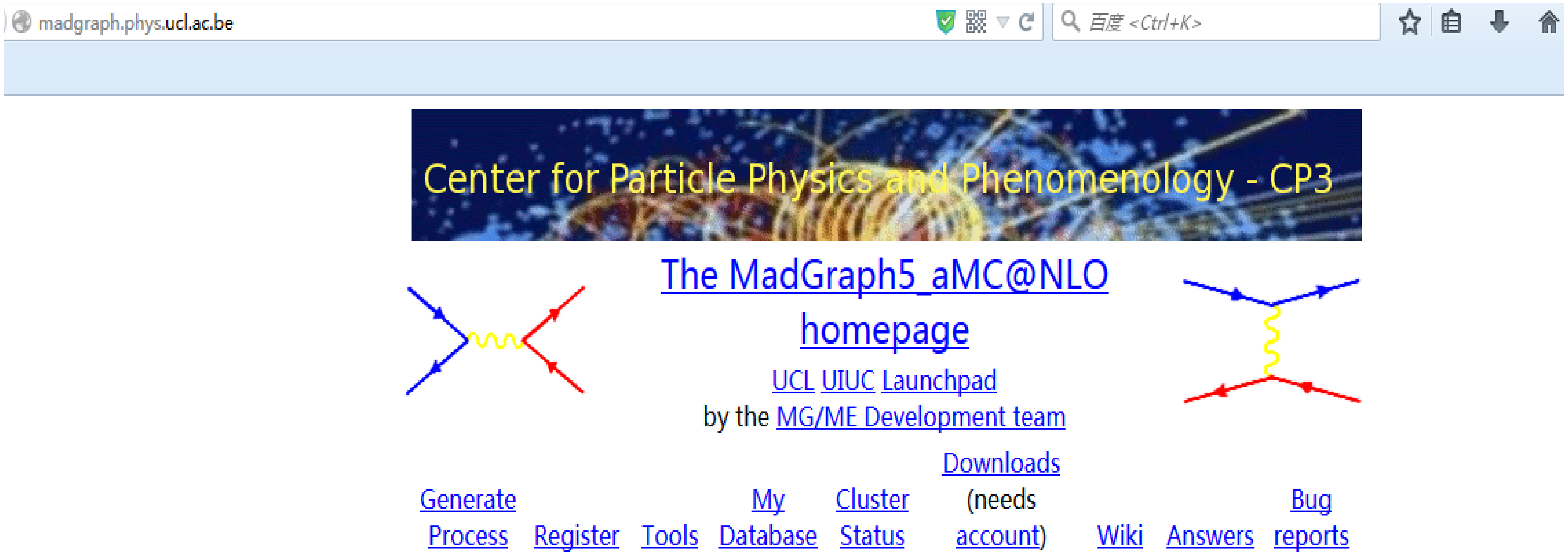


Weight: $\frac{13.448\ pb}{\#events}$

Mass Array:
$[[800\ GeV, 500\ GeV], [800\ GeV, 500\ GeV]]$

Example: MG_aMC@NLO

PP > Z LO & NLO



madgraph.phys.ucl.ac.be

Center for Particle Physics and Phenomenology - CP3

The MadGraph5_aMC@NLO homepage

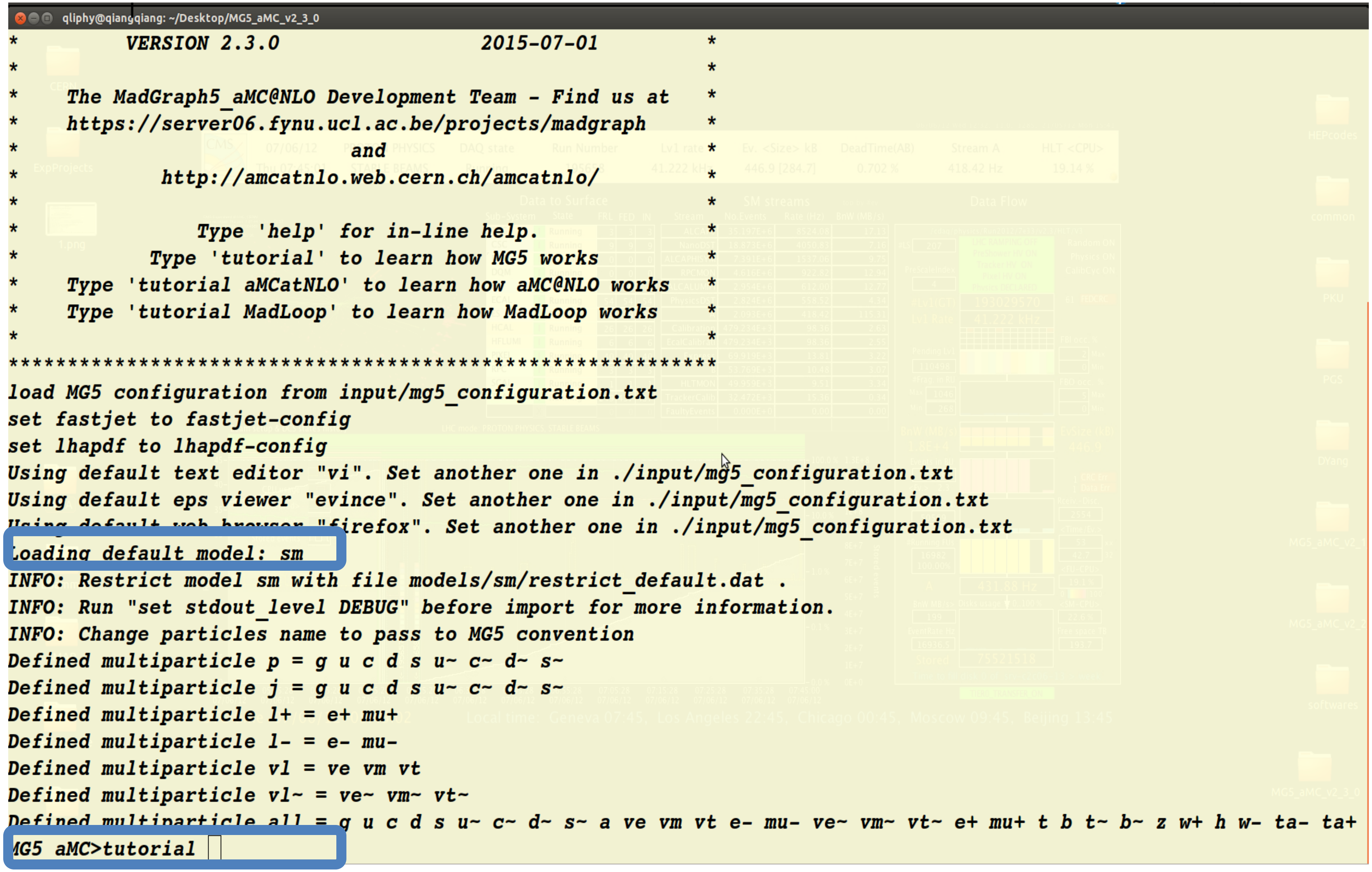
UCL UIUC Launchpad

by the MG/ME Development team

Generate Process	Register	Tools	My Database	Cluster Status	Downloads (needs account)	Wiki	Answers	Bug reports

Generate processes online using MadGraph5_aMC@NLO

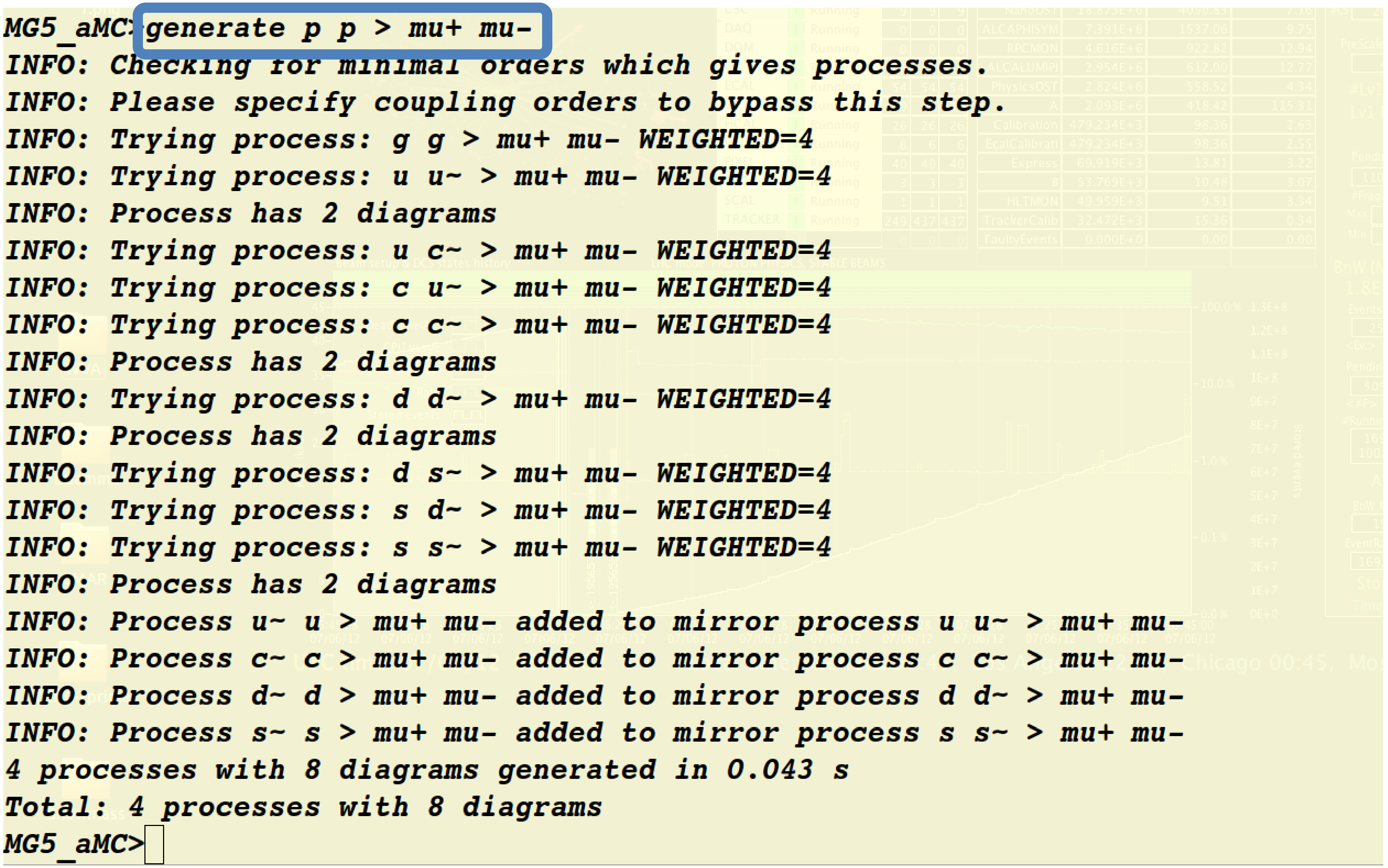
Example: PP > Z LO & NLO



qliphy@qiangqiang: ~/Desktop/MG5_aMC_v2_3_0

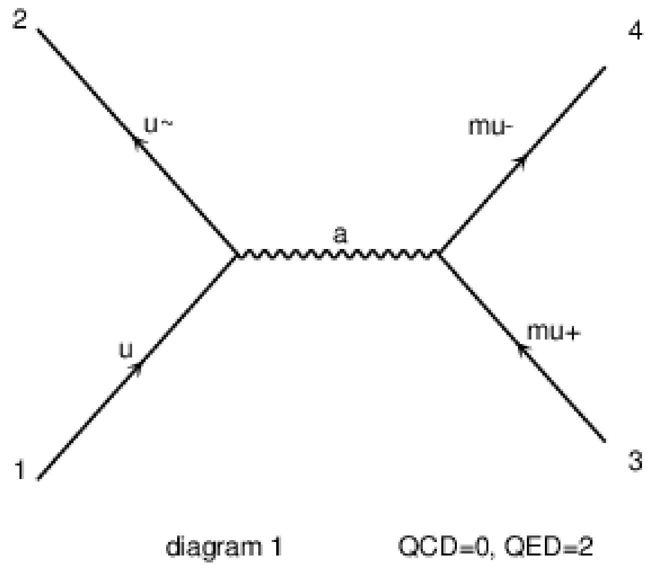
```
*         VERSION 2.3.0                 2015-07-01          *
*                                                           *
*    The MadGraph5_aMC@NLO Development Team - Find us at    *
*    https://server06.fynu.ucl.ac.be/projects/madgraph      *
*                            and                            *
*            http://amcatnlo.web.cern.ch/amcatnlo/          *
*                                                           *
*               Type 'help' for in-line help.               *
*           Type 'tutorial' to learn how MG5 works          *
*    Type 'tutorial aMCatNLO' to learn how aMC@NLO works    *
*    Type 'tutorial MadLoop' to learn how MadLoop works     *
*                                                           *
************************************************************
load MG5 configuration from input/mg5_configuration.txt
set fastjet to fastjet-config
set lhapdf to lhapdf-config
Using default text editor "vi". Set another one in ./input/mg5_configuration.txt
Using default eps viewer "evince". Set another one in ./input/mg5_configuration.txt
Using default web browser "firefox". Set another one in ./input/mg5_configuration.txt
Loading default model: sm
INFO: Restrict model sm with file models/sm/restrict_default.dat .
INFO: Run "set stdout_level DEBUG" before import for more information.
INFO: Change particles name to pass to MG5 convention
Defined multiparticle p = g u c d s u~ c~ d~ s~
Defined multiparticle j = g u c d s u~ c~ d~ s~
Defined multiparticle l+ = e+ mu+
Defined multiparticle l- = e- mu-
Defined multiparticle vl = ve vm vt
Defined multiparticle vl~ = ve~ vm~ vt~
Defined multiparticle all = g u c d s u~ c~ d~ s~ a ve vm vt e- mu- ve~ vm~ vt~ e+ mu+ t b t~ b~ z w+ h w- ta- ta+
MG5_aMC>tutorial
```

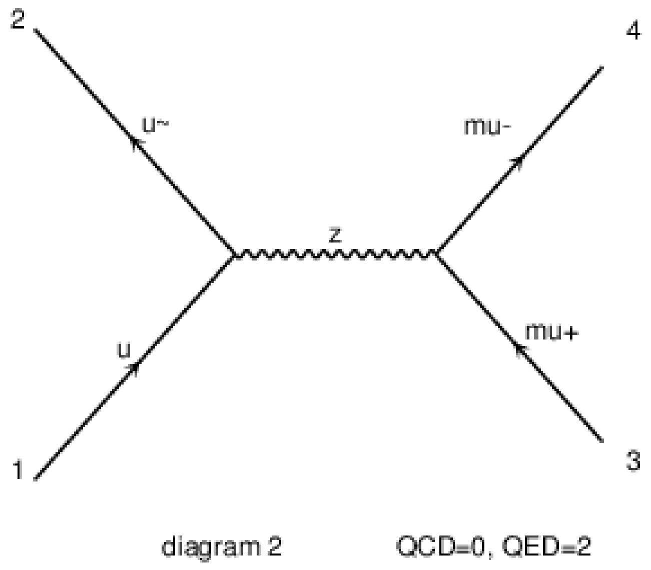
Example: PP > Z LO & NLO



```
MG5_aMC>generate p p > mu+ mu-
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying process: g g > mu+ mu- WEIGHTED=4
INFO: Trying process: u u~ > mu+ mu- WEIGHTED=4
INFO: Process has 2 diagrams
INFO: Trying process: u c~ > mu+ mu- WEIGHTED=4
INFO: Trying process: c u~ > mu+ mu- WEIGHTED=4
INFO: Trying process: c c~ > mu+ mu- WEIGHTED=4
INFO: Process has 2 diagrams
INFO: Trying process: d d~ > mu+ mu- WEIGHTED=4
INFO: Process has 2 diagrams
INFO: Trying process: d s~ > mu+ mu- WEIGHTED=4
INFO: Trying process: s d~ > mu+ mu- WEIGHTED=4
INFO: Trying process: s s~ > mu+ mu- WEIGHTED=4
INFO: Process has 2 diagrams
INFO: Process u~ u > mu+ mu- added to mirror process u u~ > mu+ mu-
INFO: Process c~ c > mu+ mu- added to mirror process c c~ > mu+ mu-
INFO: Process d~ d > mu+ mu- added to mirror process d d~ > mu+ mu-
INFO: Process s~ s > mu+ mu- added to mirror process s s~ > mu+ mu-
4 processes with 8 diagrams generated in 0.043 s
Total: 4 processes with 8 diagrams
MG5_aMC>
```

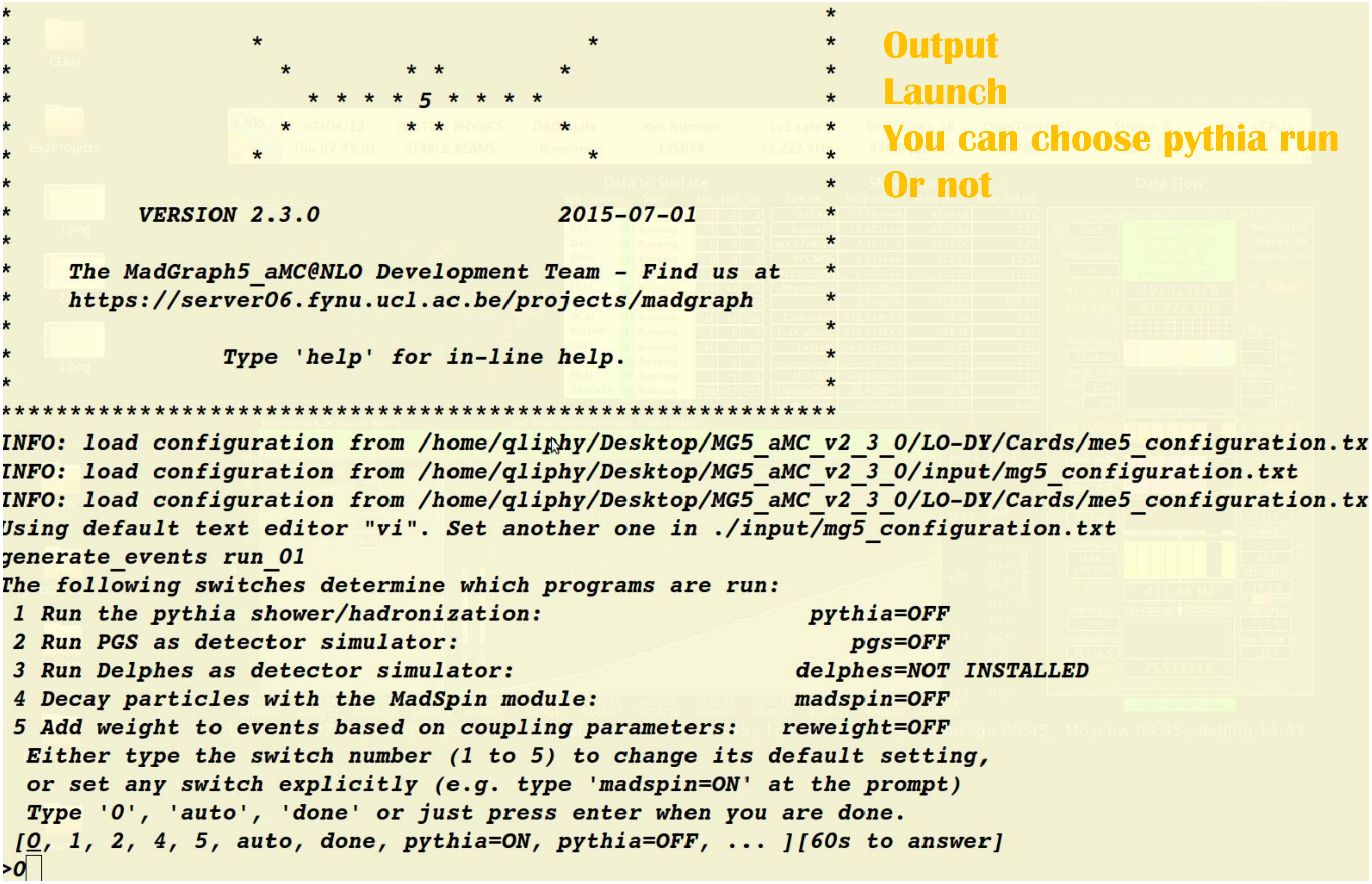
Example: PP > Z LO & NLO





You can choose QCD or QED vertex number

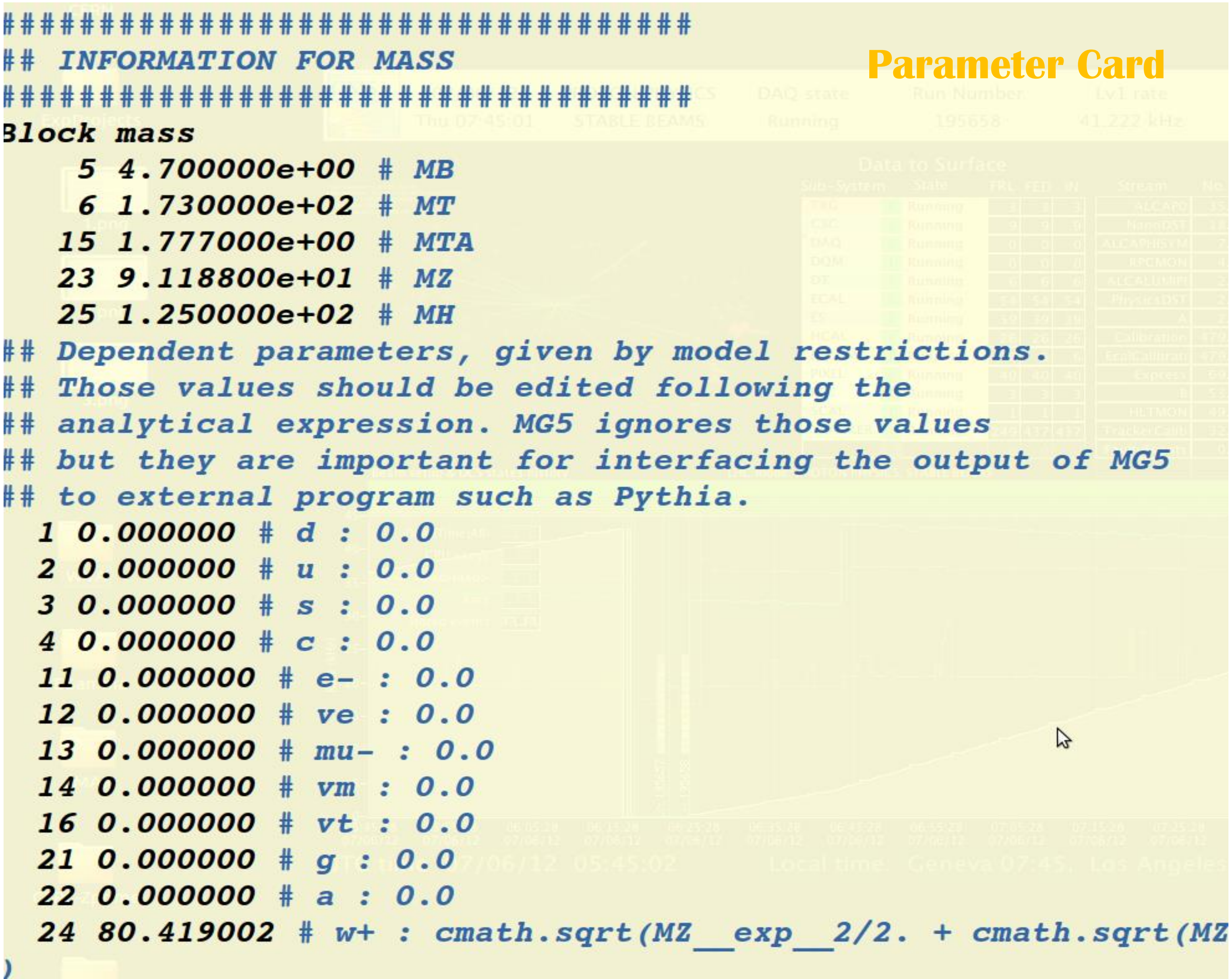
Example: PP > Z LO & NLO



```
*                                                          *
*                  *                       *               *
*                    *        * *          *               *
*                      * * * * 5 * * * *                   *
*                    *        * *          *               *
*                  *                       *               *
*                                                          *
*          VERSION 2.3.0                2015-07-01         *
*                                                          *
*     The MadGraph5_aMC@NLO Development Team - Find us at  *
*     https://server06.fynu.ucl.ac.be/projects/madgraph    *
*                                                          *
*                Type 'help' for in-line help.             *
*                                                          *
************************************************************
INFO: load configuration from /home/qliphy/Desktop/MG5_aMC_v2_3_0/LO-DY/Cards/me5_configuration.tx
INFO: load configuration from /home/qliphy/Desktop/MG5_aMC_v2_3_0/input/mg5_configuration.txt
INFO: load configuration from /home/qliphy/Desktop/MG5_aMC_v2_3_0/LO-DY/Cards/me5_configuration.tx
Using default text editor "vi". Set another one in ./input/mg5_configuration.txt
generate_events run_01
The following switches determine which programs are run:
 1 Run the pythia shower/hadronization:                  pythia=OFF
 2 Run PGS as detector simulator:                           pgs=OFF
 3 Run Delphes as detector simulator:                   delphes=NOT INSTALLED
 4 Decay particles with the MadSpin module:             madspin=OFF
 5 Add weight to events based on coupling parameters:  reweight=OFF
  Either type the switch number (1 to 5) to change its default setting,
  or set any switch explicitly (e.g. type 'madspin=ON' at the prompt)
  Type '0', 'auto', 'done' or just press enter when you are done.
 [0, 1, 2, 4, 5, auto, done, pythia=ON, pythia=OFF, ... ][60s to answer]
>0
```

Output
Launch
You can choose pythia run
Or not

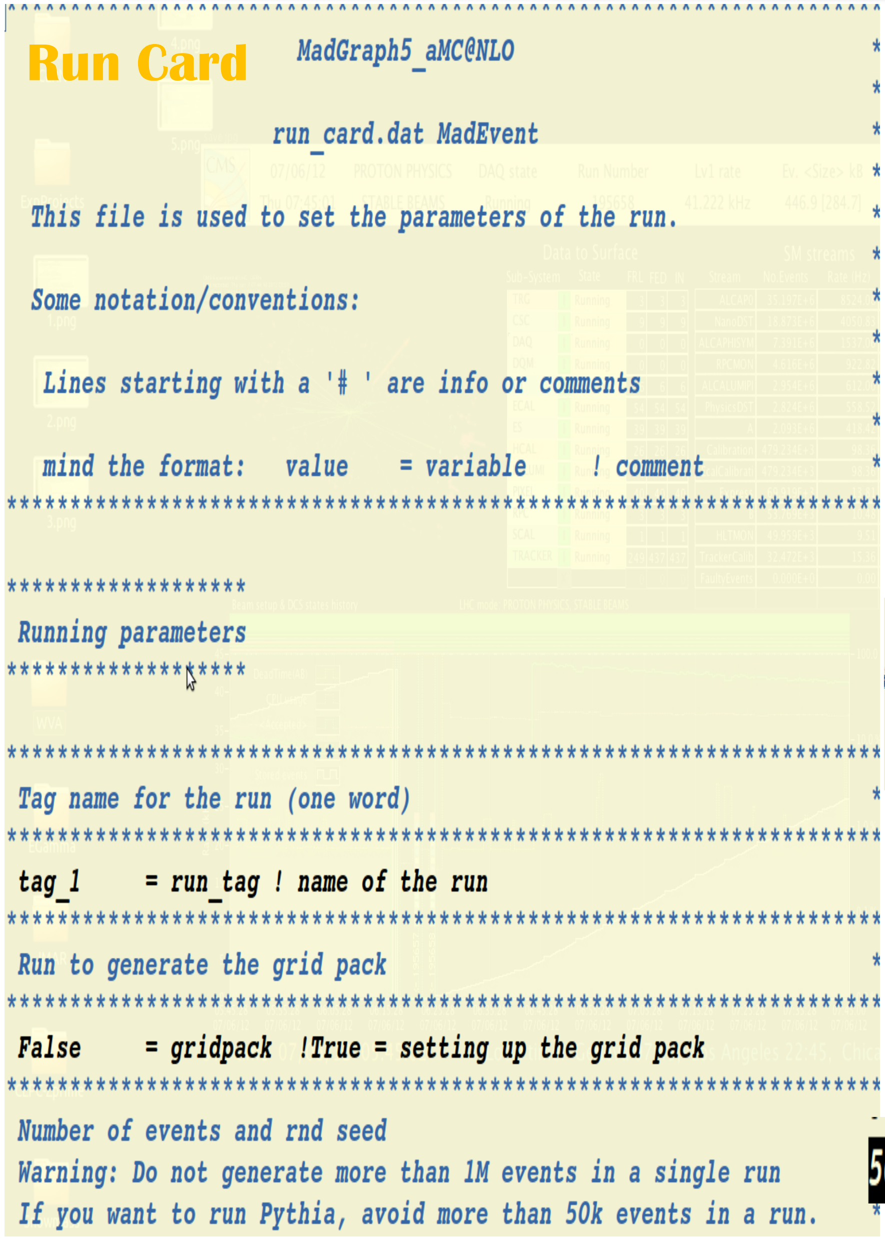
Example: PP > Z LO & NLO



Parameter Card

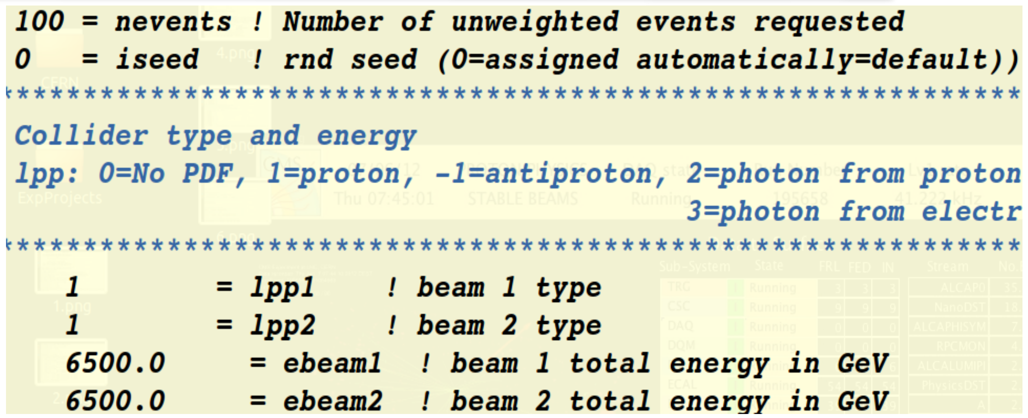
```
####################################
## INFORMATION FOR MASS
####################################
Block mass
    5 4.700000e+00 # MB
    6 1.730000e+02 # MT
   15 1.777000e+00 # MTA
   23 9.118800e+01 # MZ
   25 1.250000e+02 # MH
## Dependent parameters, given by model restrictions.
## Those values should be edited following the
## analytical expression. MG5 ignores those values
## but they are important for interfacing the output of MG5
## to external program such as Pythia.
  1 0.000000 # d : 0.0
  2 0.000000 # u : 0.0
  3 0.000000 # s : 0.0
  4 0.000000 # c : 0.0
  11 0.000000 # e- : 0.0
  12 0.000000 # ve : 0.0
  13 0.000000 # mu- : 0.0
  14 0.000000 # vm : 0.0
  16 0.000000 # vt : 0.0
  21 0.000000 # g : 0.0
  22 0.000000 # a : 0.0
  24 80.419002 # w+ : cmath.sqrt(MZ__exp__2/2. + cmath.sqrt(MZ
)
```

Example: PP > Z LO & NLO

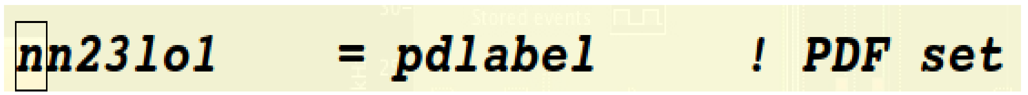


Run Card

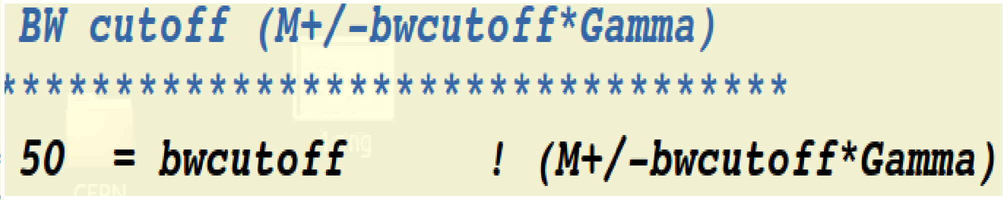
```
**********************************************************************
                 MadGraph5_aMC@NLO                                   *
                                                                     *
                       run_card.dat MadEvent                         *
                                                                     *
  This file is used to set the parameters of the run.                *
                                                                     *
  Some notation/conventions:                                         *
                                                                     *
   Lines starting with a '# ' are info or comments                   *
                                                                     *
   mind the format:   value    = variable     ! comment              *
**********************************************************************

******************
 Running parameters
******************

**********************************************************************
 Tag name for the run (one word)                                     *
**********************************************************************
 tag_1     = run_tag ! name of the run
**********************************************************************
 Run to generate the grid pack                                       *
**********************************************************************
 False     = gridpack  !True = setting up the grid pack
**********************************************************************
 Number of events and rnd seed
 Warning: Do not generate more than 1M events in a single run
 If you want to run Pythia, avoid more than 50k events in a run.
```



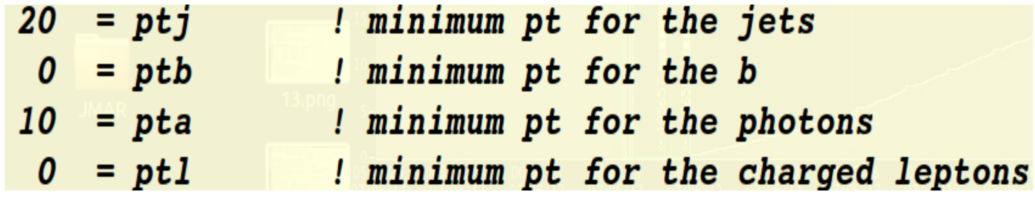
```
 100 = nevents ! Number of unweighted events requested
 0   = iseed   ! rnd seed (0=assigned automatically=default))
*****************************************************************
 Collider type and energy
 lpp: 0=No PDF, 1=proton, -1=antiproton, 2=photon from proton
                                         3=photon from electr
*****************************************************************
    1         = lpp1    ! beam 1 type
    1         = lpp2    ! beam 2 type
    6500.0     = ebeam1  ! beam 1 total energy in GeV
    6500.0     = ebeam2  ! beam 2 total energy in GeV
```



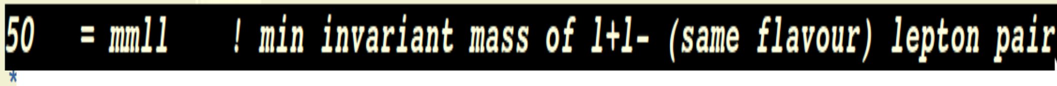
```
nn23lo1     = pdlabel      ! PDF set
```



```
 BW cutoff (M+/-bwcutoff*Gamma)
*********************************
 50  = bwcutoff      ! (M+/-bwcutoff*Gamma)
```

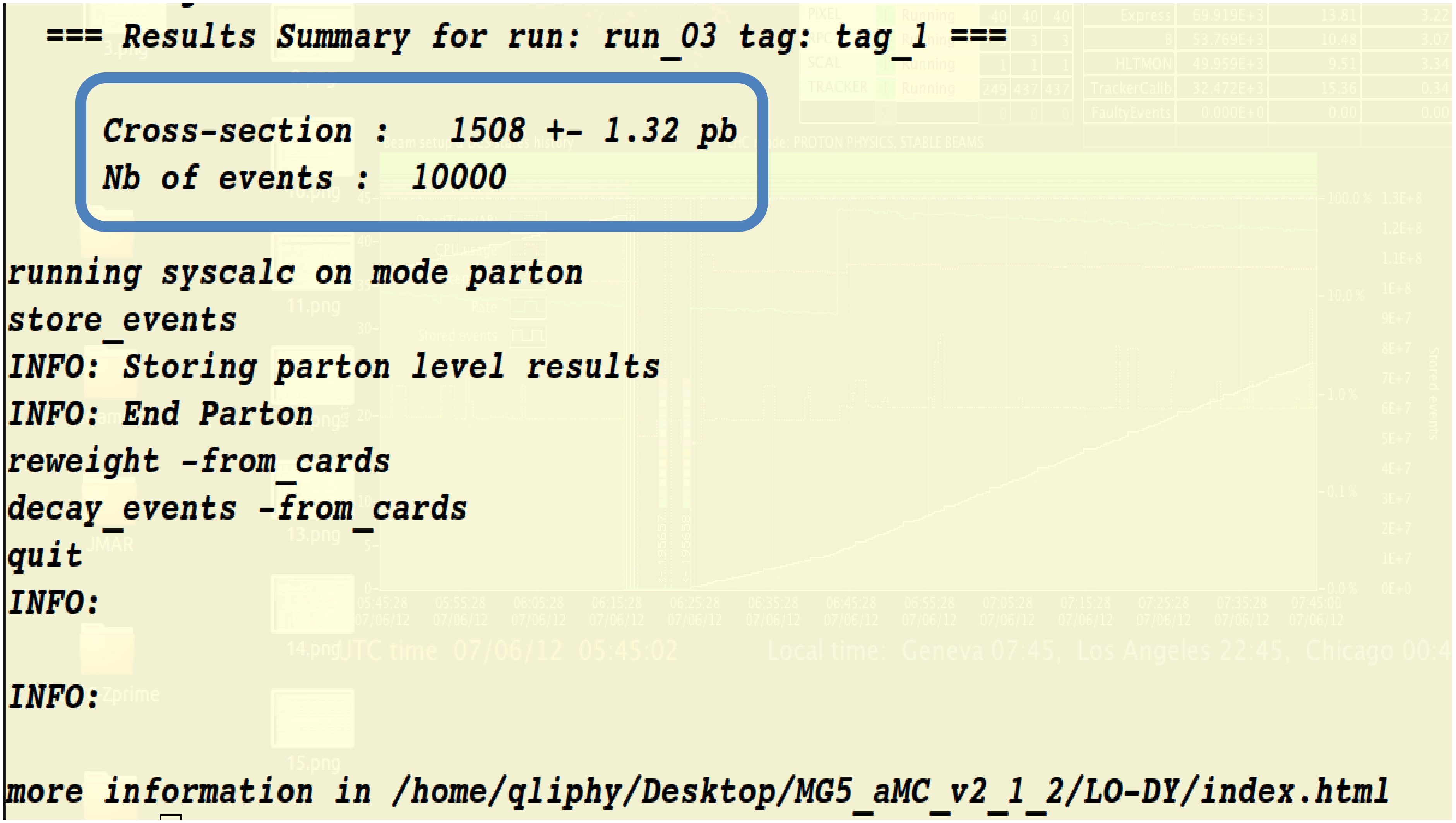


```
 20  = ptj       ! minimum pt for the jets
  0  = ptb       ! minimum pt for the b
 10  = pta       ! minimum pt for the photons
  0  = ptl       ! minimum pt for the charged leptons
```



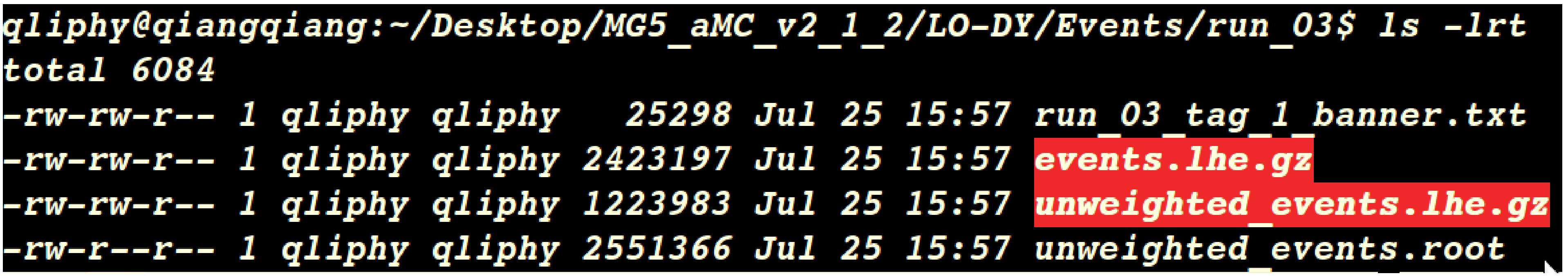
```
50   = mmll    ! min invariant mass of l+l- (same flavour) lepton pair
```

Example: PP > Z   LO & NLO

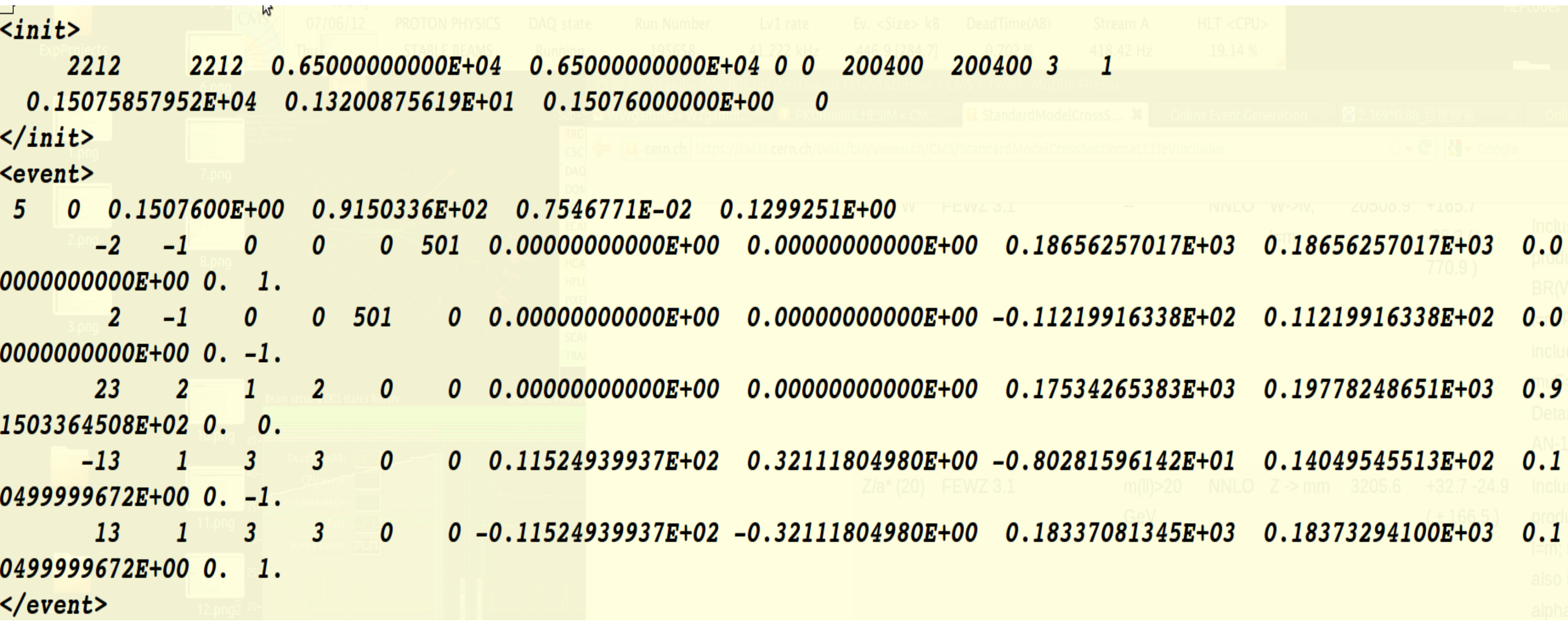


```
  === Results Summary for run: run_03 tag: tag_1 ===

     Cross-section :   1508 +- 1.32 pb
     Nb of events :  10000

running syscalc on mode parton
store_events
INFO: Storing parton level results
INFO: End Parton
reweight -from_cards
decay_events -from_cards
quit
INFO:

INFO:

more information in /home/qliphy/Desktop/MG5_aMC_v2_1_2/LO-DY/index.html
```

Example: PP > Z LO & NLO

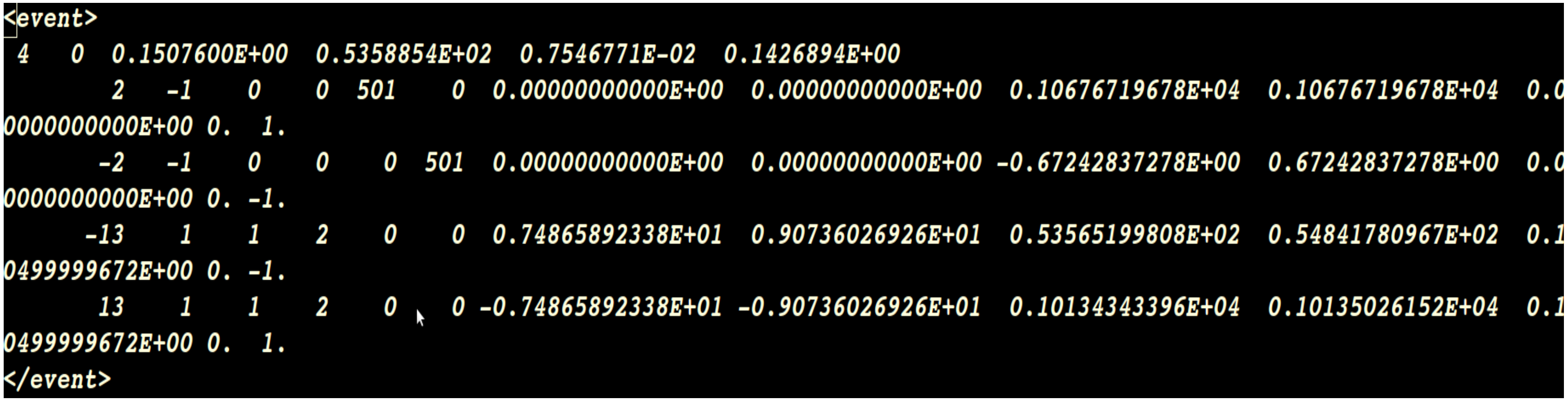


```
qliphy@qiangqiang:~/Desktop/MG5_aMC_v2_1_2/LO-DY/Events/run_03$ ls -lrt
total 6084
-rw-rw-r-- 1 qliphy qliphy   25298 Jul 25 15:57 run_03_tag_1_banner.txt
-rw-rw-r-- 1 qliphy qliphy 2423197 Jul 25 15:57 events.lhe.gz
-rw-rw-r-- 1 qliphy qliphy 1223983 Jul 25 15:57 unweighted_events.lhe.gz
-rw-r--r-- 1 qliphy qliphy 2551366 Jul 25 15:57 unweighted_events.root
```



```
<init>
     2212      2212  0.65000000000E+04  0.65000000000E+04 0 0  200400  200400 3   1
  0.15075857952E+04  0.13200875619E+01  0.15076000000E+00    0
</init>
<event>
 5   0  0.1507600E+00  0.9150336E+02  0.7546771E-02  0.1299251E+00
       -2    -1     0     0     0   501  0.00000000000E+00  0.00000000000E+00  0.18656257017E+03  0.18656257017E+03  0.0
0000000000E+00 0.  1.
        2    -1     0     0   501     0  0.00000000000E+00  0.00000000000E+00 -0.11219916338E+02  0.11219916338E+02  0.0
0000000000E+00 0. -1.
       23     2     1     2     0     0  0.00000000000E+00  0.00000000000E+00  0.17534265383E+03  0.19778248651E+03  0.9
1503364508E+02 0.  0.
      -13     1     3     3     0     0  0.11524939937E+02  0.32111804980E+00 -0.80281596142E+01  0.14049545513E+02  0.1
0499999672E+00 0. -1.
       13     1     3     3     0     0 -0.11524939937E+02 -0.32111804980E+00  0.18337081345E+03  0.18373294100E+03  0.1
0499999672E+00 0.  1.
</event>
```

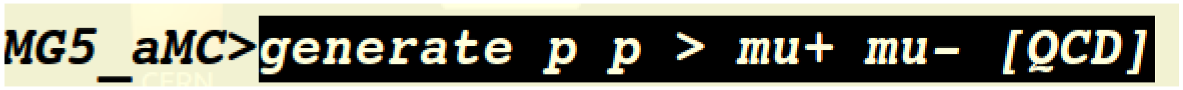
Example: PP > Z LO & NLO



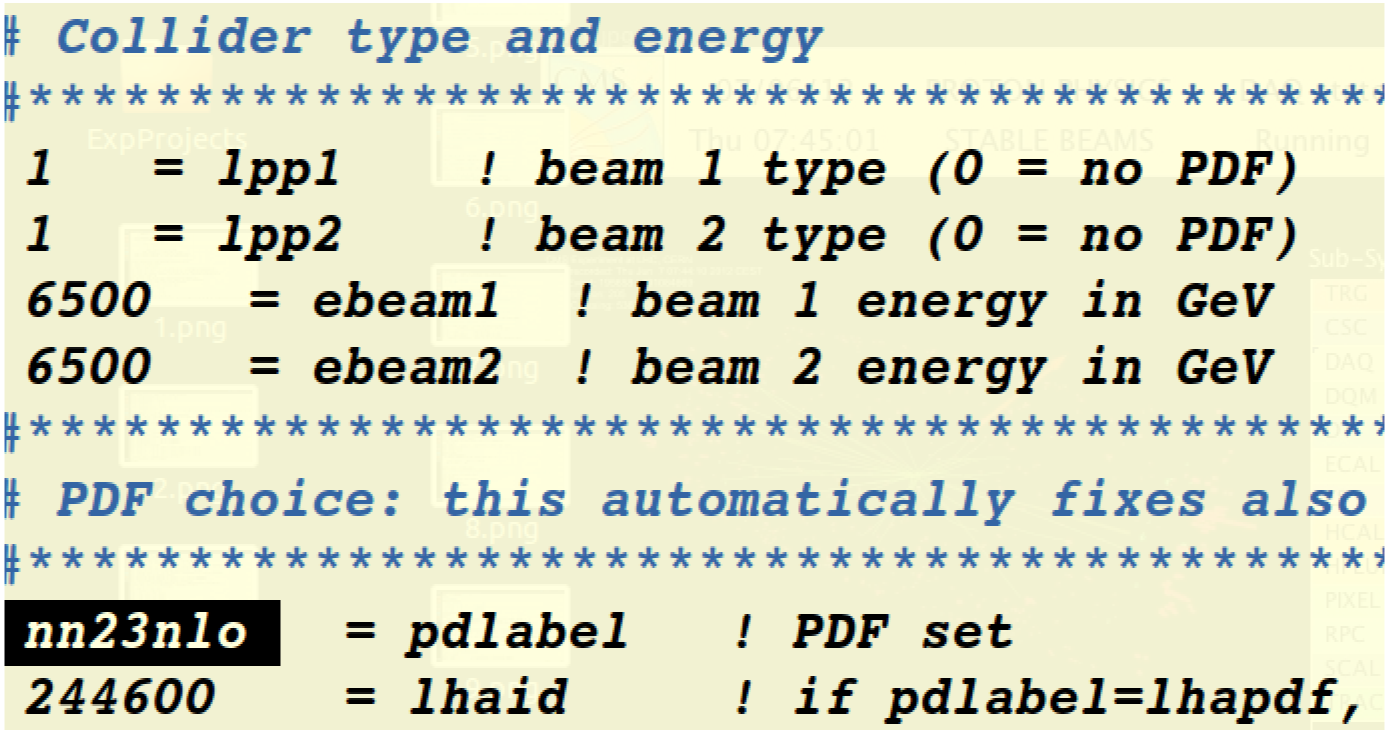
```
<event>
 4    0  0.1507600E+00  0.5358854E+02  0.7546771E-02  0.1426894E+00
        2    -1     0     0   501     0  0.00000000000E+00  0.00000000000E+00  0.10676719678E+04  0.10676719678E+04  0.0
0000000000E+00 0.  1.
       -2    -1     0     0     0   501  0.00000000000E+00  0.00000000000E+00 -0.67242837278E+00  0.67242837278E+00  0.0
0000000000E+00 0. -1.
      -13     1     1     2     0     0  0.74865892338E+01  0.90736026926E+01  0.53565199808E+02  0.54841780967E+02  0.1
0499999672E+00 0. -1.
       13     1     1     2     0     0 -0.74865892338E+01 -0.90736026926E+01  0.10134343396E+04  0.10135026152E+04  0.1
0499999672E+00 0.  1.
</event>
```

PP to Photon to mumubar

Example: PP > Z LO & NLO

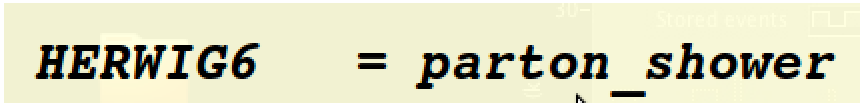


```
MG5_aMC>generate p p > mu+ mu- [QCD]
```



```
# Collider type and energy
#*****************************************
 1     = lpp1     ! beam 1 type (0 = no PDF)
 1     = lpp2     ! beam 2 type (0 = no PDF)
 6500     = ebeam1   ! beam 1 energy in GeV
 6500     = ebeam2   ! beam 2 energy in GeV
#*****************************************
# PDF choice: this automatically fixes also
#*****************************************
nn23nlo    = pdlabel    ! PDF set
244600     = lhaid      ! if pdlabel=lhapdf,
```

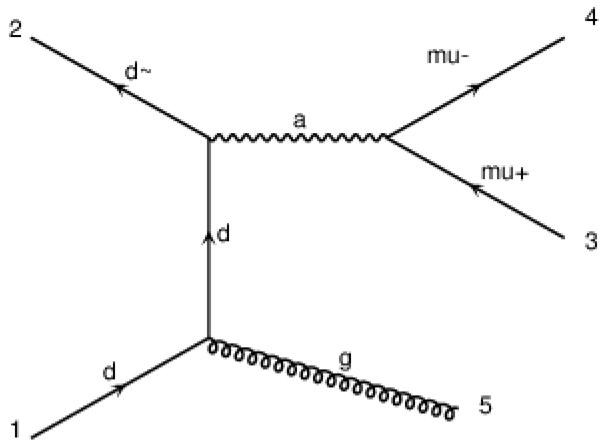
NLO PDF for NLO, LO PDF for LO



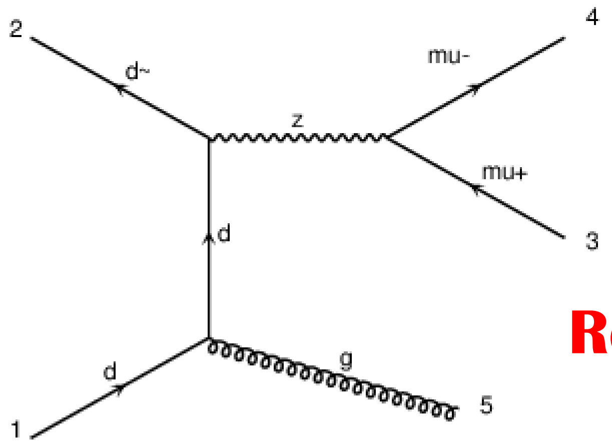
```
HERWIG6    = parton_shower
```

ME + PS, to be mentioned later

Example: PP > Z LO & NLO



real diagram 1 QCD=1, QED=2



real diagram 2 QCD=1, QED=2

Real emission

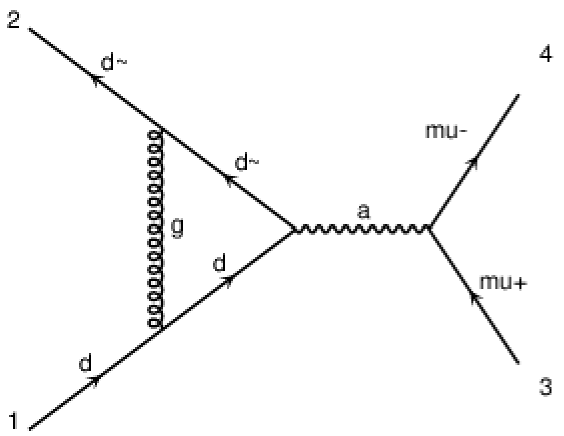


diagram 1 QCD=2, QED=2

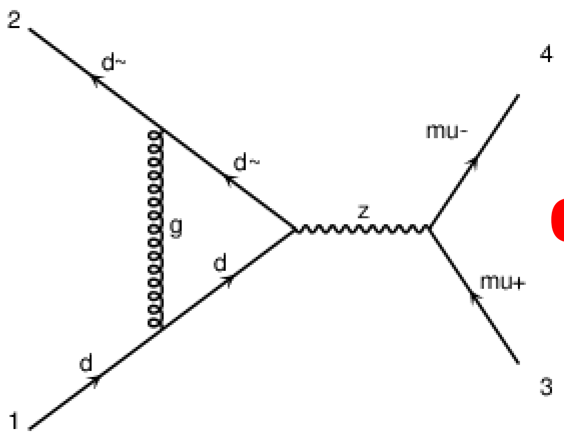
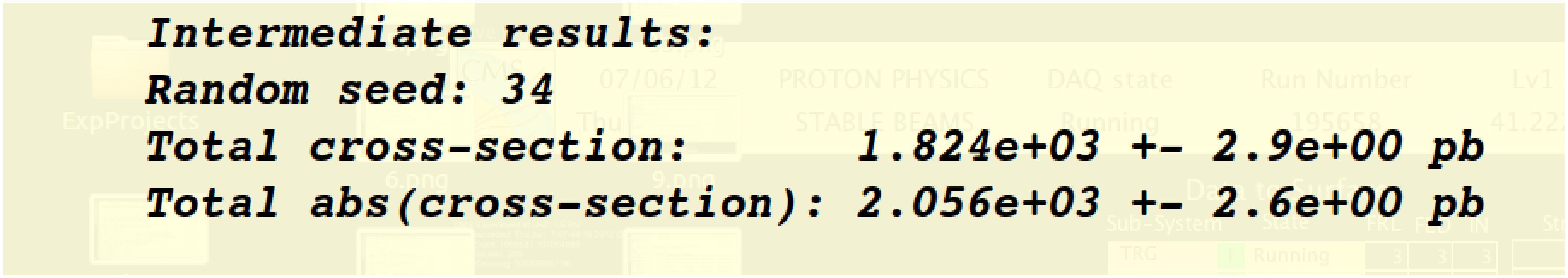


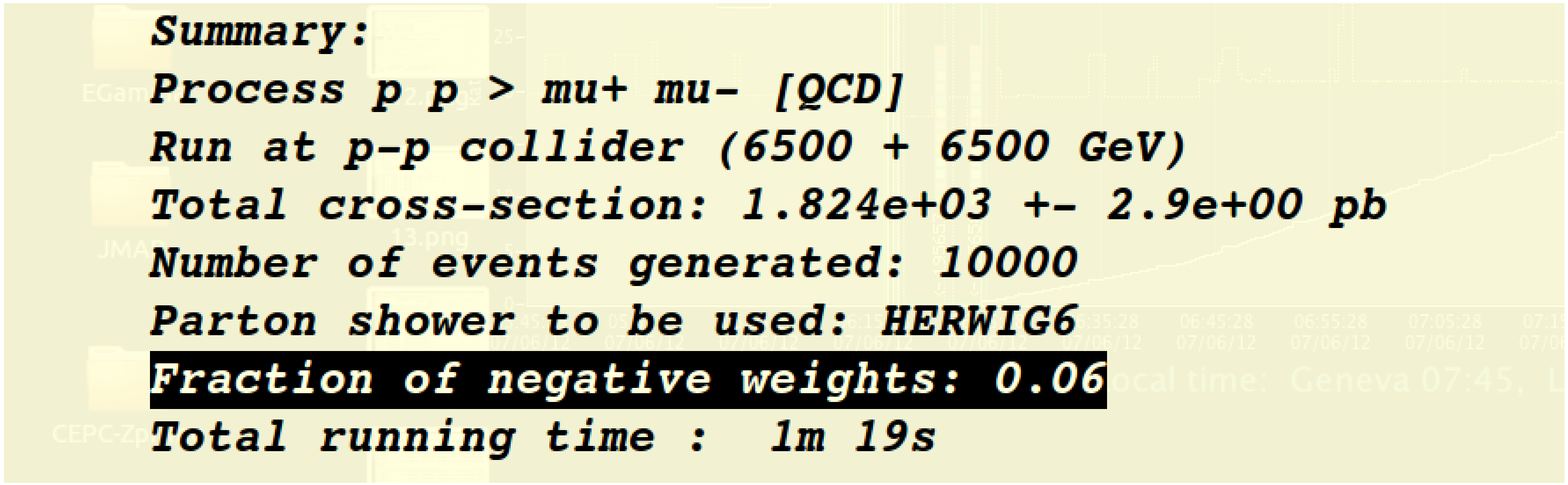
diagram 2 QCD=2, QED=2

One Loop virtual

Example: PP > Z LO & NLO



```
Intermediate results:
Random seed: 34
Total cross-section:       1.824e+03 +- 2.9e+00 pb
Total abs(cross-section): 2.056e+03 +- 2.6e+00 pb
```

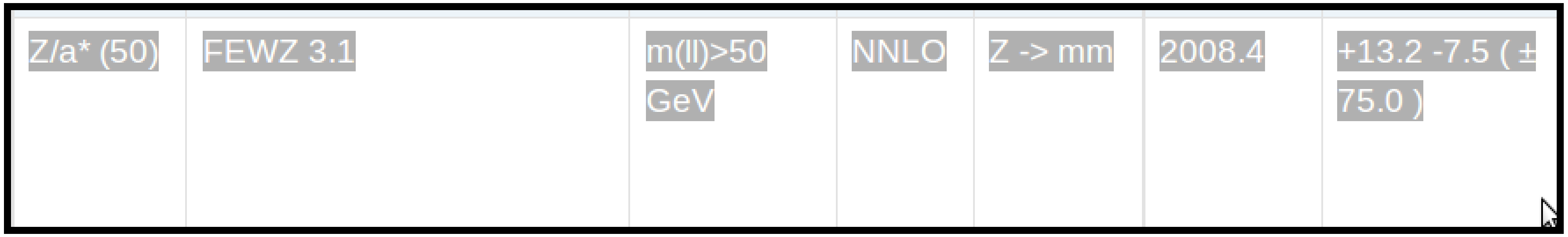


```
Summary:
Process p p > mu+ mu- [QCD]
Run at p-p collider (6500 + 6500 GeV)
Total cross-section: 1.824e+03 +- 2.9e+00 pb
Number of events generated: 10000
Parton shower to be used: HERWIG6
Fraction of negative weights: 0.06
Total running time :  1m 19s
```

K Facor: 1824/1508 ~ 1.21

2.056*(0.94-0.06)=1.81

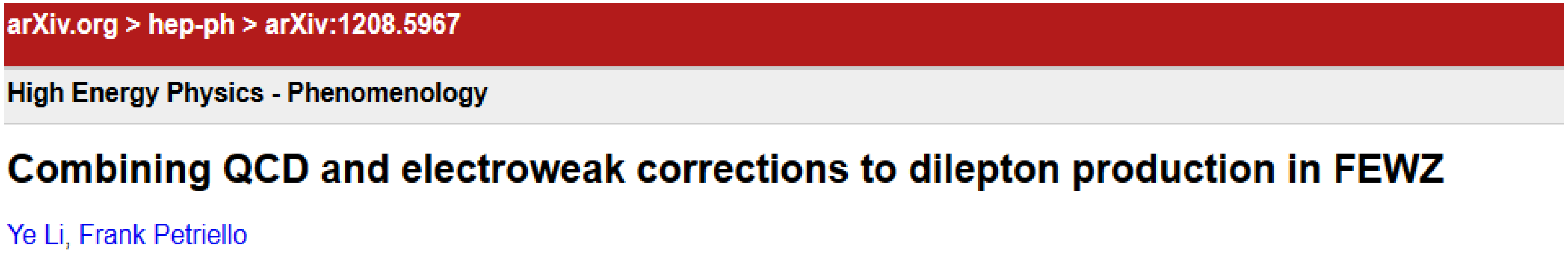
Example: PP > Z LO & NLO



Z/a* (50)	FEWZ 3.1	m(ll)>50 GeV	NNLO	Z -> mm	2008.4	+13.2 -7.5 (± 75.0)

NLO/LO 1824/1508 ~ 1.21

NNLO/NLO 2008.4/1824 ~ 1.1 NLO EWK also included



arXiv.org > hep-ph > arXiv:1208.5967

High Energy Physics - Phenomenology

Combining QCD and electroweak corrections to dilepton production in FEWZ

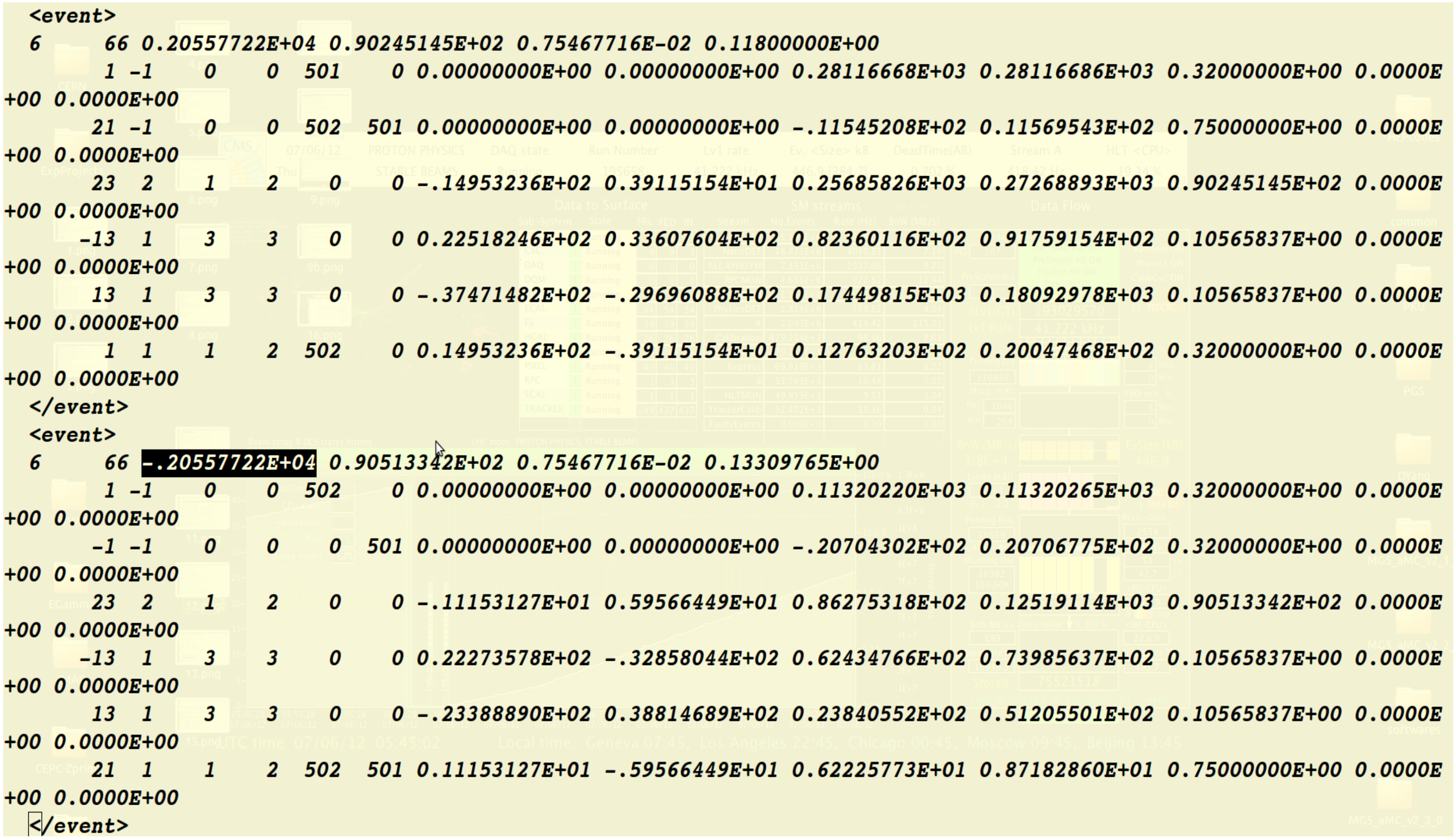
Ye Li, Frank Petriello

Example: PP > Z LO & NLO

NLO events: additional parton in the final state

```
 <event>
 6    66 0.20557722E+04 0.88575911E+02 0.75467716E-02 0.11800000E+00
        2 -1    0    0  501    0 0.00000000E+00 0.00000000E+00 0.32758644E+02 0.32760207E+02 0.32000000E+00 0.0000E
+00 0.0000E+00
       21 -1    0    0  502  501 0.00000000E+00 0.00000000E+00 -.25056521E+03 0.25056633E+03 0.75000000E+00 0.0000E
+00 0.0000E+00
       23  2    1    2    0    0 0.12823333E+02 0.44733748E+01 -.29224945E+02 0.94256237E+02 0.88575911E+02 0.0000E
+00 0.0000E+00
      -13  1    3    3    0    0 -.28120157E+02 0.10814566E+02 -.41280973E+02 0.51106046E+02 0.10565837E+00 0.0000E
+00 0.0000E+00
       13  1    3    3    0    0 0.40943489E+02 -.63411912E+01 0.12056028E+02 0.43150191E+02 0.10565837E+00 0.0000E
+00 0.0000E+00
        2  1    1    2  502    0 -.12823333E+02 -.44733748E+01 -.18858162E+03 0.18907030E+03 0.32000000E+00 0.0000E
+00 0.0000E+00
 </event>
 <event>
 5    66 0.20557722E+04 0.90465747E+02 0.75467716E-02 0.11800000E+00
       -1 -1    0    0    0  501 0.00000000E+00 0.00000000E+00 0.21814416E+01 0.22047874E+01 0.32000000E+00 0.0000E
+00 0.0000E+00
        1 -1    0    0  501    0 0.00000000E+00 0.00000000E+00 -.93290233E+03 0.93290239E+03 0.32000000E+00 0.0000E
+00 0.0000E+00
       23  2    1    2    0    0 0.00000000E+00 0.00000000E+00 -.93072089E+03 0.93510717E+03 0.90465747E+02 0.0000E
+00 0.0000E+00
      -13  1    3    3    0    0 -.69025294E+01 0.30106640E+02 -.12379180E+03 0.12758713E+03 0.10565837E+00 0.0000E
+00 0.0000E+00
       13  1    3    3    0    0 0.69025294E+01 -.30106640E+02 -.80692909E+03 0.80752005E+03 0.10565837E+00 0.0000E
+00 0.0000E+00
 </event>
```

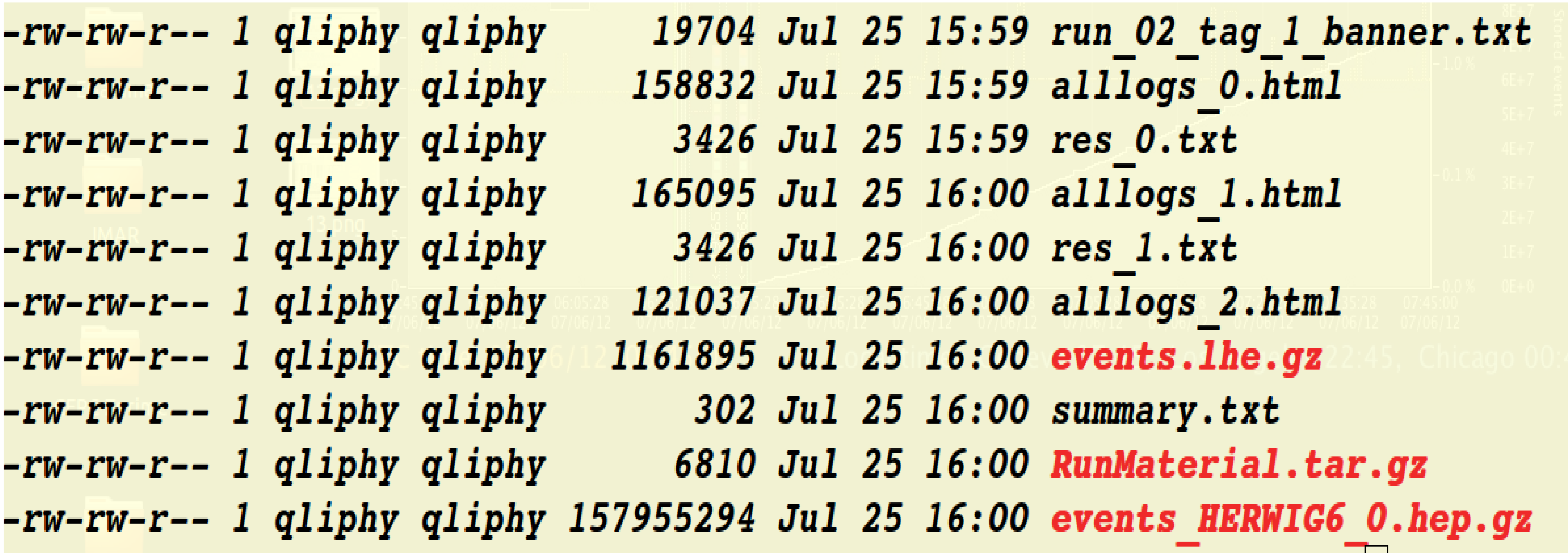
Example: PP > Z LO & NLO



```
  <event>
  6      66 0.20557722E+04 0.90245145E+02 0.75467716E-02 0.11800000E+00
            1 -1    0    0  501    0 0.00000000E+00 0.00000000E+00 0.28116668E+03 0.28116686E+03 0.32000000E+00 0.0000E
+00 0.0000E+00
           21 -1    0    0  502  501 0.00000000E+00 0.00000000E+00 -.11545208E+02 0.11569543E+02 0.75000000E+00 0.0000E
+00 0.0000E+00
           23  2    1    2    0    0 -.14953236E+02 0.39115154E+01 0.25685826E+03 0.27268893E+03 0.90245145E+02 0.0000E
+00 0.0000E+00
          -13  1    3    3    0    0 0.22518246E+02 0.33607604E+02 0.82360116E+02 0.91759154E+02 0.10565837E+00 0.0000E
+00 0.0000E+00
           13  1    3    3    0    0 -.37471482E+02 -.29696088E+02 0.17449815E+03 0.18092978E+03 0.10565837E+00 0.0000E
+00 0.0000E+00
            1  1    1    2  502    0 0.14953236E+02 -.39115154E+01 0.12763203E+02 0.20047468E+02 0.32000000E+00 0.0000E
+00 0.0000E+00
  </event>
  <event>
  6      66 -.20557722E+04 0.90513342E+02 0.75467716E-02 0.13309765E+00
            1 -1    0    0  502    0 0.00000000E+00 0.00000000E+00 0.11320220E+03 0.11320265E+03 0.32000000E+00 0.0000E
+00 0.0000E+00
           -1 -1    0    0    0  501 0.00000000E+00 0.00000000E+00 -.20704302E+02 0.20706775E+02 0.32000000E+00 0.0000E
+00 0.0000E+00
           23  2    1    2    0    0 -.11153127E+01 0.59566449E+01 0.86275318E+02 0.12519114E+03 0.90513342E+02 0.0000E
+00 0.0000E+00
          -13  1    3    3    0    0 0.22273578E+02 -.32858044E+02 0.62434766E+02 0.73985637E+02 0.10565837E+00 0.0000E
+00 0.0000E+00
           13  1    3    3    0    0 -.23388890E+02 0.38814689E+02 0.23840552E+02 0.51205501E+02 0.10565837E+00 0.0000E
+00 0.0000E+00
           21  1    1    2  502  501 0.11153127E+01 -.59566449E+01 0.62225773E+01 0.87182860E+01 0.75000000E+00 0.0000E
+00 0.0000E+00
  </event>
```

NLO events: negative weight

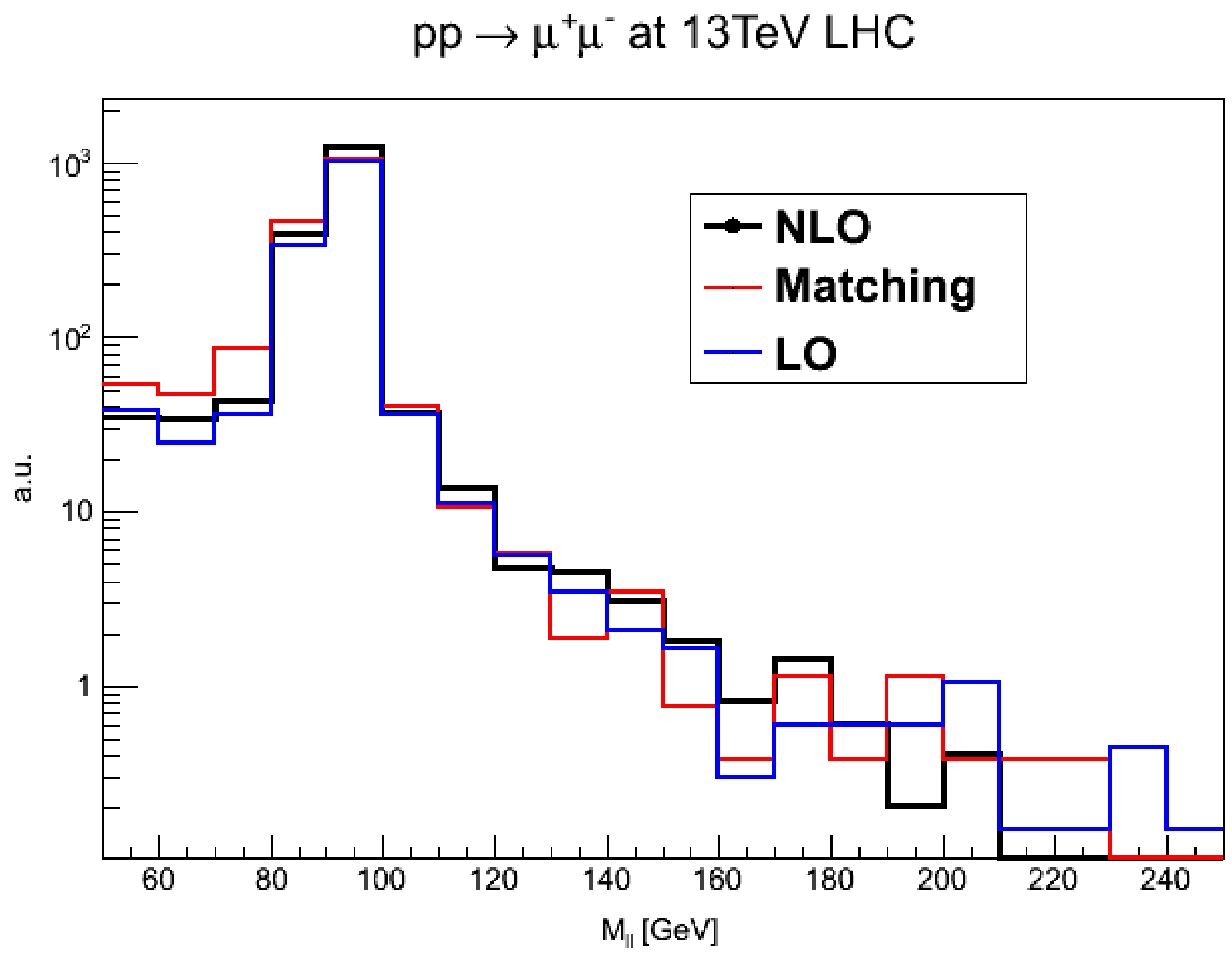
Example: PP > Z LO & NLO



```
-rw-rw-r-- 1 qliphy qliphy     19704 Jul 25 15:59 run_02_tag_1_banner.txt
-rw-rw-r-- 1 qliphy qliphy    158832 Jul 25 15:59 alllogs_0.html
-rw-rw-r-- 1 qliphy qliphy      3426 Jul 25 15:59 res_0.txt
-rw-rw-r-- 1 qliphy qliphy    165095 Jul 25 16:00 alllogs_1.html
-rw-rw-r-- 1 qliphy qliphy      3426 Jul 25 16:00 res_1.txt
-rw-rw-r-- 1 qliphy qliphy    121037 Jul 25 16:00 alllogs_2.html
-rw-rw-r-- 1 qliphy qliphy   1161895 Jul 25 16:00 events.lhe.gz
-rw-rw-r-- 1 qliphy qliphy       302 Jul 25 16:00 summary.txt
-rw-rw-r-- 1 qliphy qliphy      6810 Jul 25 16:00 RunMaterial.tar.gz
-rw-rw-r-- 1 qliphy qliphy 157955294 Jul 25 16:00 events_HERWIG6_0.hep.gz
```

hep file is after Parton Shower, huge size

Example: LO vs NLO vs Matching


$pp \to \mu^+\mu^-$ at 13TeV LHC
a.u.
10^3
10^2
M_{ll} [GeV]
NLO
Matching
LO