Random number generators and application

Marcin Chrząszcz
mchrzasz@cern.ch

University of Zurich[UZH]

Experimental Methods in Particle Physics,
19 November, 2015

Random and pseudorandom numbers

John von Neumann:

”Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number — there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method.”

$\Rrightarrow$ Random number: a given value that is taken by a random variable

$\twoheadrightarrow$ by definition cannot be predicted.

$\Rrightarrow$ Sources of truly random numbers:

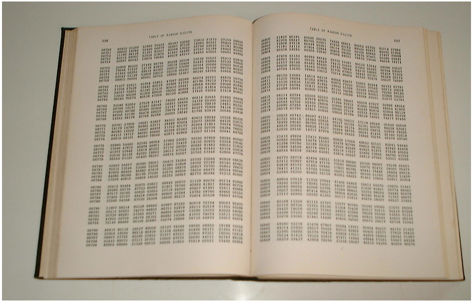
- Mechanical
- Physical

$\Rrightarrow$ Disadvantages of physical generators:

- To slow for typical applications, especially the mechanical ones!
- Not stable; small changes in boundary conditions might lead to completely different results!

Random numbers - history remark

⇛ In the past there were books with random numbers:



⇛ It's obvious that they didn't become very popular ;)

⇛ This methods are comming back!

↠ Storage device are getting more cheap and bigger (CD, DVD).

↠ 1995: G. Marsaglia, $650\mathrm{MB}$ of random numbers, "White and Black Noise".

Pseudorandom numbers

$\Rrightarrow$ Pseudorandom numbers are numbers that are generated accordingly to strict mathematical formula.
$\looparrowright$ Strictly speaking they are non random numbers, how ever they have all the statistical properties of random numbers.
$\looparrowright$ Discussing those properties is a wide topic so let's just say that without knowing the formula they are generated by one cannot say if those numbers are random or not.
$\Rrightarrow$ Mathematical methods of producing pseudorandom numbers:

- Good statistical properties of generated numbers.
- Easy to use and fast!
- Reproducible!

$\Rrightarrow$ Since mathematical pseudorandom genrators are dominantly: pseudorandom $\rightarrowtail$ random.

Middle square generator; von Neumann

⇛ The first mathematical generator (middle square) was proposed by von Neumann (1964).

↬ Formula: $$X_n = \lfloor X_{n-1}^2 \cdot 10^{-m} \rfloor - \lfloor X_{n-1}^2 \cdot 10^{-3m} \rfloor$$

↬ where X_0 is a constant (seed), $\lfloor \cdot \rfloor$ is the cut-off of a number to integer.

⇛ Example:

Let's put $m = 2$ and $X_0 = 2045$:

↬ $X_0^2 = \underbrace{04}_{\text{rej}}1820\underbrace{25}_{\text{rej}}$ $\Rightarrow X_1 = 1820$

↬ $X_1^2 = \underbrace{03}_{\text{rej}}3124\underbrace{00}_{\text{rej}}$ $\Rightarrow X_1 = 3124$

↬ Simple generator but unfortunately quite bad generator. Firstly the sequences are very short and strongly dependent on the X_0 number.

Linear generators Lecture2/Linear_gen1

⇛ This was a first generator written and it's a good example how to not write generators.

⇛ It's highly non stable!

```
mchrzasz-ThinkPad-W530% python gen.py 14714 4
21650.0
46872.0
219698.0
4826721.0
2329723538.0
5.42761170924e+14
2.94589685716e+25
8.67830820626e+46
7.53130325698e+89
Traceback (most recent call last):
  File "gen.py", line 29, in <module>
    sys.exit(main())
  File "gen.py", line 22, in main
    tmp=X0**2
OverflowError: (34, 'Numerical result out of range')
```

Linear generators

⇛ General equation:

$$X_n = (a_1 X_{n-1} + a_2 X_{n-2} + ... + a_k X_{n-k} + c) \text{ mod } m,$$

↬ where a_i, c, m are parameters of a generator(integer numbers).

↬ Generator initialization ⇄ setting those parameters.

⇛ Very old generators. (often used in Pascal, or first C versions):

$$k = 1: \;\; X_n = (aX_{n-1} + c) \text{ mod} m,$$

$$c = \begin{cases} = 0, \text{multiplicativegeneator} \\ \neq 0, \text{mixgeneator} \end{cases}$$

⇛ The period can be achieved by tuning the seed parameters:

$$P_{\max} = \begin{cases} 2^{L-2}; \text{ for } m = 2^L \\ m - 1; \text{ for } m = \text{prime number} \end{cases}$$

Shift register generator

⇛ General equation:

$$b_n = (a_1 X_{n-1} + a_2 X_{n-2} + ... + a_k X_{n-k} + c) \bmod 2,$$

where $a_i \subset (\{0, 1\})$

⇛ Super fast and easy to implement due to: $(a + b) \bmod 2 = a \text{ xor } b$

a	b	a xor b
0	0	0
1	0	1
0	1	1
1	1	0

⇛ Maximal period is $2^k - 1$.

⇛ Example (Tausworths generator):

$a_p = a_q = 1$, other $a_i = 0$ and $p > q$. Then: $b_n = b_{n-p} \text{ xor } b_{n-q}$

⇛ How to get numbers from bits (for example):

$U_i = \sum_{j=1}^{L} 2^{-j} b_{is+j},\ \ s < L$.

Fibonacci generator

⇛ In 1202 Fibonacci with Leonardo in Piza:

$$f_n = f_{n-2} + f_{n-1},\ n \geqslant 2$$

⇛ Based on this first generator was created (Taussky and Todd, 1956):

$$X_n = (X_{n-2} + X_{n-1}) \bmod m,\ n \geqslant 2$$

This generator isn't so good in terms of statistics tests.
⇛ Generalization:

$$X_n = (X_{n-r} \odot X_{n-s}) \bmod m,\ n \geqslant r,\ s \geqslant 1$$

$\odot$	P_{max}	Stat. properties
$+, -$	$(2^r - 1)2^{L-1}$	good
x	$(2^r - 1)2^{L-13}$	very good
xor	$(2^r - 1)$	poor

Multiply with carry, generator

⇛ We start from:

$$b_n = (a_1 X_{n-1} + a_2 X_{n-2} + ... + a_k X_{n-k} + c) \bmod m,$$

where $a_1, .., a_k \in \mathbb{N}$ are constant parameters.

⇛ The c parameters is calculated foe each step:

$$c = \lfloor (a_1 X_{n-1} + a_2 X_{n-2} + ... + a_k X_{n-k} + c)/m \rfloor,$$

⇛ Initialization: $a_1, .., a_k, c$.

⇛ Advantages:

- Fast and easy to implement.
- Large period.
- Good statistical properties.
- First proposed by Marsaglia.

Subtract with borrow, generator

⇛ Created again by Marsaglia (1991):

$$X_n = (X_{n-r} \ominus X_{n-s}) \bmod m,\ r, s \in \mathbb{N},$$

where :

$$x \ominus y = \begin{cases} x - y - c + m,\ c = 1,\ \text{when x} - \text{y} - \text{c} < 0 \\ x - y - c,\ c = 0,\ \text{when x} - \text{y} - \text{c} \geqslant 0 \end{cases}$$

⇛ Initialization: $X_1, ..., X_{n-r}$ and $c = 0$.

⇛ Fast and easy :)

⇛ Fails some of the basic statistics tests.

Non linear generators

⇛ The natural solutions to problems of linear generators are the non-linear generators (second part of 1980s).
⇛ Eichenauera i Lehna (1986):

$$X_n = (aX_{n1}^{-1} + b) \bmod m,$$

⇛ Eichenauera-Hermanna (1993)

$$X_n = [a(n + n_0) + b]^{-1} \bmod m,$$

⇛ L. Blum, M. Blum, Shub (1986):

$$X_n = X_{n-1}^2 \bmod m,$$

↣ Very popular in cryptography.
⇛ Pros and cons:

- They all pass all statistical tests.
- Much slower then linear generators.

RANLUX generator

⇛ All described generators are based on some mathematical algorithms and recursion. The typical scheme is of constructing a MC generator:

- Think of a formula that takes some initial values.
- Generate large number of random numbers and put them through statistical tests.
- If the test are positive we accept the the generator.

⇛ Now let's think: why the hell numbers obtained that way are showing some random number properties?

RANLUX generator

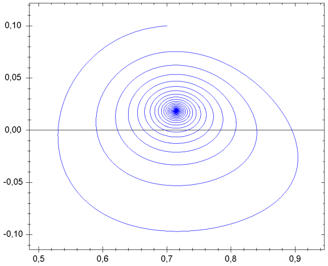
⇛ All described generators are based on some mathematical algorithms and recursion. The typical scheme is of constructing a MC generator:

- Think of a formula that takes some initial values.
- Generate large number of random numbers and put them through statistical tests.
- If the test are positive we accept the the generator.

⇛ Now let's think: why the hell numbers obtained that way are showing some random number properties? There is no science behind it, it's pure luck!
⇛ M.Luscher (1993) hep-lat/9309020
⇛ Generator RANLUX based on Kolomogorow entropy and Lyapunov exponent. Effectively we are building inside the generator the chaos theory.
⇛ RANLUX and Mersenne Twister (TRandom1, TRandom3) are the 2 most powerful generators in the world that passed every known statistical test.

Chaos theory in a nut shell

⇛ We know that the solution of classical systems is described by trajectory in phase spaces. Now the problem with this picture starts to be when arround one point in this phase space we are getting more and more trajectories that are drifting a part later on.

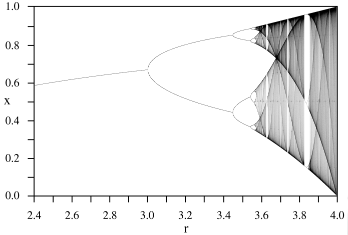


⇛ The Lyapunov exponent tells us how a two solutions drift apart with time:

$$|\delta X(t)| \approx e^{\lambda t} |\delta X_0|$$

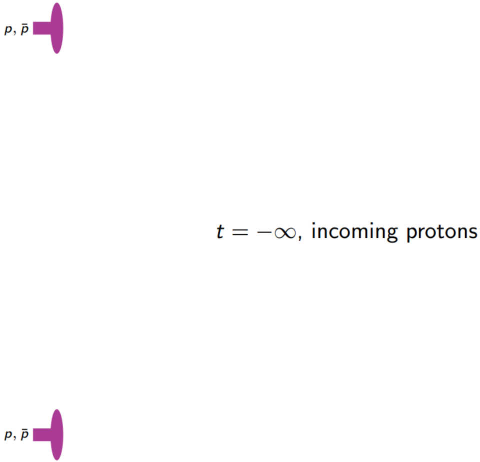
⇛ Kolomogorow entropy:

$$h_K = \int_P \lambda d\mu$$



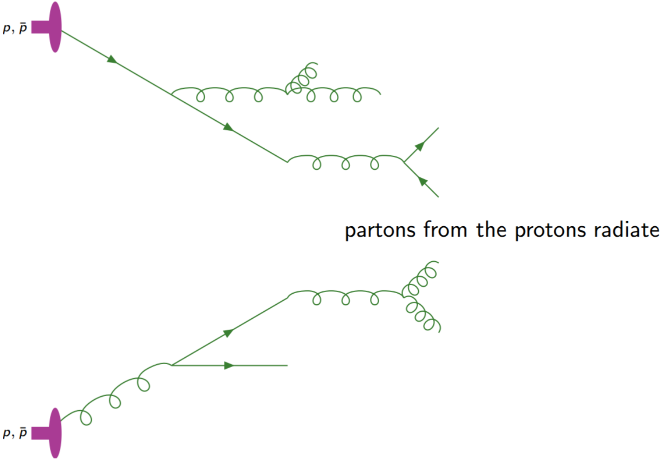
HEP simulation

⇛ There is some ambiguity what particle physicist call MC. Normally those are mathematical theorises but when we say MC we usually mean MC simulation of a physics process. ⇛ There are plenty of things that need to be simulated:



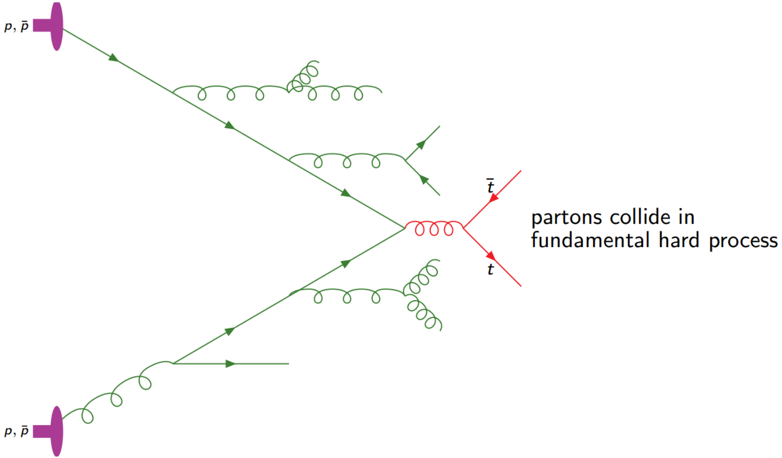
HEP simulation

⇛ There is some ambiguity what particle physicist call MC. Normally those are mathematical theorises but when we say MC we usually mean MC simulation of a physics process. ⇛ There are plenty of things that need to be simulated:



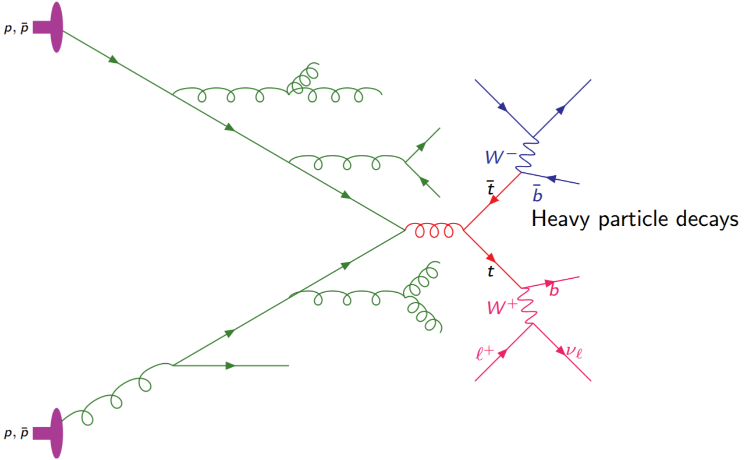
HEP simulation

⇛ There is some ambiguity what particle physicist call MC. Normally those are mathematical theorises but when we say MC we usually mean MC simulation of a physics process. ⇛ There are plenty of things that need to be simulated:



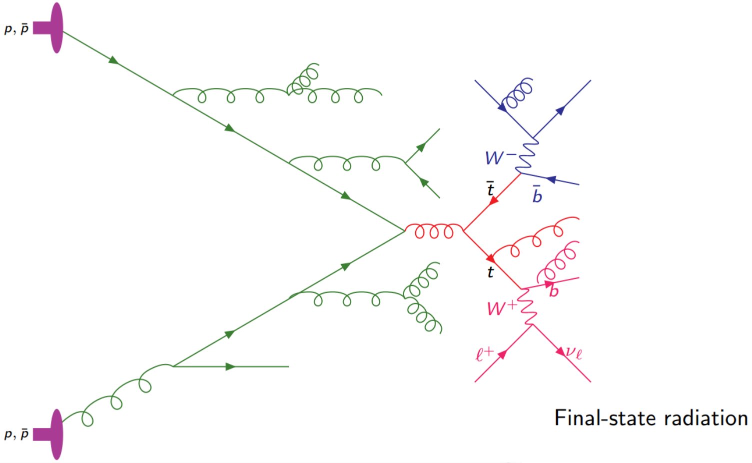
HEP simulation

⇛ There is some ambiguity what particle physicist call MC. Normally those are mathematical theorises but when we say MC we usually mean MC simulation of a physics process. ⇛ There are plenty of things that need to be simulated:



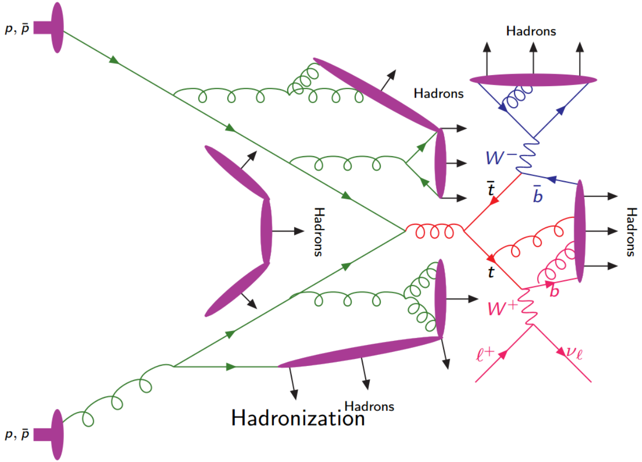
HEP simulation

⇛ There is some ambiguity what particle physicist call MC. Normally those are mathematical theorises but when we say MC we usually mean MC simulation of a physics process. ⇛ There are plenty of things that need to be simulated:



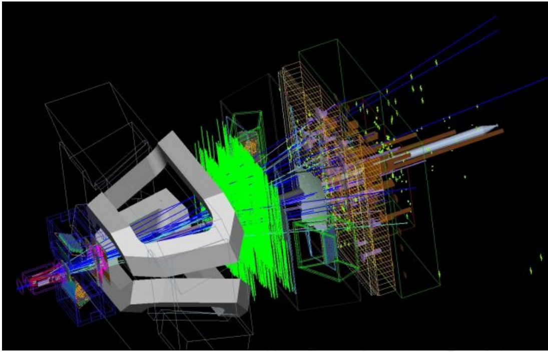
HEP simulation

$\Rightarrow$ There is some ambiguity what particle physicist call MC. Normally those are mathematical theorises but when we say MC we usually mean MC simulation of a physics process. $\Rightarrow$ There are plenty of things that need to be simulated:



Detector simulation

⇛ Things do not get simpler on the detector side simulation.
⇛ Lots of effects need to be taken into account:



⟶ Bremsstrahlung
⟶ Interactions with different detector materials
⟶ Particle identification
⟶ Showers

⇛ Example of generators:
⟶ FLUKA
⟶ Geant

Method of Moments

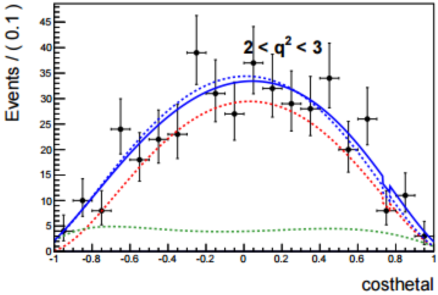
⇛ Now real cool things!

⇛ Let's consider we want to study a rare decay: $B^{\pm} \to K^{\pm}\mu\mu$. The decay is described by the following PDF:

$$\frac{1}{\Gamma}\frac{d^2\Gamma}{dq^2 d\cos\theta_l} = \frac{3}{4}(1-F_H)(1-\cos^2\theta_l) + F_H/2 + A_{FB}\cos\theta_l$$

⇛ PDF by construction is normalized: $\int_{-1}^{1}\frac{1}{\Gamma}\frac{d^2\Gamma}{dq^2 d\cos\theta_l} = 1$

- Normally we do a likelihood fit and we are done.
- There is a second way!



Method of Moments

⇛ Let's calculate the integrals:

$$\int_{-1}^{1} \frac{1}{\Gamma} \frac{d^2\Gamma}{dq^2 d\cos\theta_l} \cdot \cos\theta_l = \frac{2}{3} A_{FB}$$

$$\int_{-1}^{1} \frac{1}{\Gamma} \frac{d^2\Gamma}{dq^2 d\cos\theta_l} \cdot \cos^2\theta_l = \frac{1}{5} + \frac{2F_H}{15}$$

⇛ So we can get our parameters that we searched for by doing a integration. So now what?

Method of Moments

⇛ Let's calculate the integrals:

$$\int_{-1}^{1} \frac{1}{\Gamma} \frac{d^2\Gamma}{dq^2 d\cos\theta_l} \cdot \cos\theta_l = \frac{2}{3} A_{FB}$$

$$\int_{-1}^{1} \frac{1}{\Gamma} \frac{d^2\Gamma}{dq^2 d\cos\theta_l} \cdot \cos^2\theta_l = \frac{1}{5} + \frac{2F_H}{15}$$

⇛ So we can get our parameters that we searched for by doing a integration. So now what?

⇛ Well nature is the best random number generator so let's take the data and treat and calculate the integral estimates:

$$\int_{-1}^{1} \frac{1}{\Gamma} \frac{d^2\Gamma}{dq^2 d\cos\theta_l} \cdot \cos\theta_l = \frac{2}{3} A_{FB} = \frac{1}{N} \sum_{i=1}^{N} \cos\theta_{l,i}$$

$$\int_{-1}^{1} \frac{1}{\Gamma} \frac{d^2\Gamma}{dq^2 d\cos\theta_l} \cdot \cos^2\theta_l = \frac{1}{5} + \frac{2F_H}{15} = \frac{1}{N} \sum_{i=1}^{N} \cos^2\theta_{l,i}$$

Method of Moments

⇛ So what did we do?

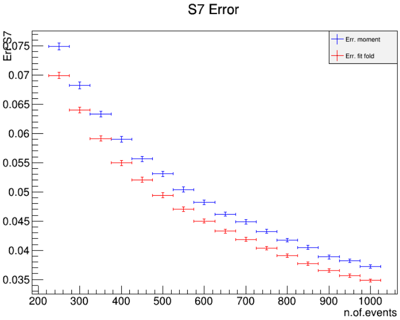
- We have just estimated a parameters of interests without using any fit!!

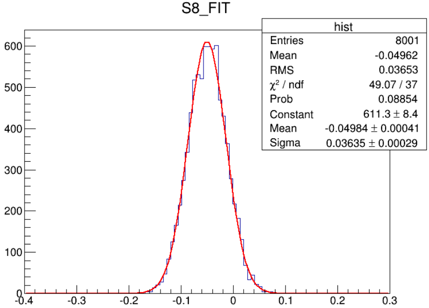
⇛ Pros and cones of method of moments:

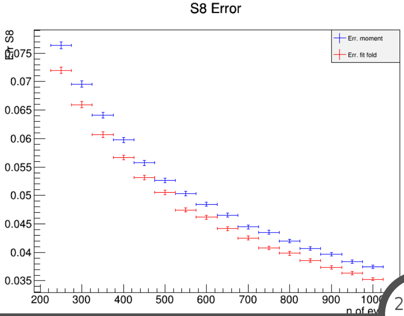
- Are very immune to bias.
- Do not suffer from boundary problems.
- Require less statistic to work then likelihood fit.
- They always have a Gaussian error.
- Estimator has a larger uncertainty.

Method of Moments, uncertainty estimator

$\Rrightarrow$ It can be proven that Method of Moments estimator converges slower then the maximum likelihood fit.







Other application of MC - testing your analysis

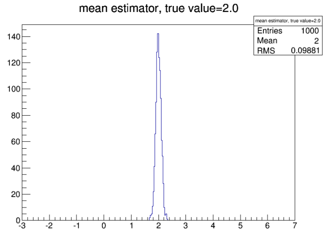
⇛ Probably the biggest application of MC methods in HEP are validations of your experimental methodology. The procedure is as follows:

- Define your analysis methodology: selection, efficiency corrections, parameters you want to measure.
- Simulate an assembly of simulation events for different values of parameters you want to measure.
- Do the analysis on this pseudo data.
- See if you are getting back what you have simulated.

Testing your analysis, Lecture2/Test_met

⇛ Probably the biggest application of MC methods in HEP are validations of your experimental methodology. The procedure is as follows:

- Define your analysis methodology: selection, efficiency corrections, parameters you want to measure.
- Simulate an assembly of simulation events for different values of parameters you want to measure.
- Do the analysis on this pseudo data.
- See if you are getting back what you have simulated.



Wrap up

$\Rrightarrow$ Things to remember:

- Computer cannot produce random numbers, only pseudorandom numbers.
- We use pseudorandon numbers as random numbers if they are statistically acting the same as random numbers.
- Linear generators are not commonly used nowadays.
- State of the art generators are the ones based on Kolomogorows theorem.
- MC methods used to simulate physics process, detector response and validating the estimators.

Backup