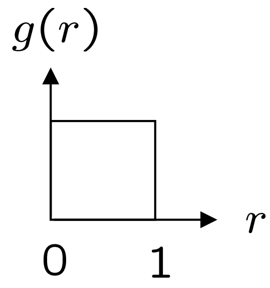
The Monte Carlo method

What it is: a numerical technique for calculating probabilities and related quantities using sequences of random numbers.

The usual steps:



(1) Generate sequence $r_1, r_2, \ldots, r_m$ uniform in [0, 1].

(2) Use this to produce another sequence $x_1, x_2, \ldots, x_n$ distributed according to some pdf $f(x)$ in which we're interested (x can be a vector).

(3) Use the x values to estimate some property of $f(x)$, e.g., fraction of x values with $a < x < b$ gives $\int_a^b f(x)\,dx$.

→ MC calculation = integration (at least formally)

MC generated values = 'simulated data'

→ use for testing statistical procedures

Random number generators

Goal: generate uniformly distributed values in [0, 1].

Toss coin for e.g. 32 bit number... (too tiring).

→ 'random number generator'

= computer algorithm to generate $r_1, r_2, ..., r_n$.

Example: multiplicative linear congruential generator (MLCG)

$n_{i+1} = (a\, n_i) \bmod m$, where

n_i = integer

a = multiplier

m = modulus

n_0 = seed (initial value)

N.B. mod = modulus (remainder), e.g. 27 mod 5 = 2.

This rule produces a sequence of numbers $n_0, n_1, ...$

Random number generators (2)

The sequence is (unfortunately) periodic!

Example (see Brandt Ch 4): $a = 3$, $m = 7$, $n_0 = 1$

$$\begin{aligned}
n_1 &= (3 \cdot 1) \bmod 7 = 3 \\
n_2 &= (3 \cdot 3) \bmod 7 = 2 \\
n_3 &= (3 \cdot 2) \bmod 7 = 6 \\
n_4 &= (3 \cdot 6) \bmod 7 = 4 \\
n_5 &= (3 \cdot 4) \bmod 7 = 5 \\
n_6 &= (3 \cdot 5) \bmod 7 = 1 \qquad \leftarrow \text{ sequence repeats}
\end{aligned}$$

Choose a, m to obtain long period (maximum = $m - 1$); m usually close to the largest integer that can represented in the computer.

Only use a subset of a single period of the sequence.

Random number generators (3)

$r_i = n_i/m$ are in [0, 1] but are they 'random'?

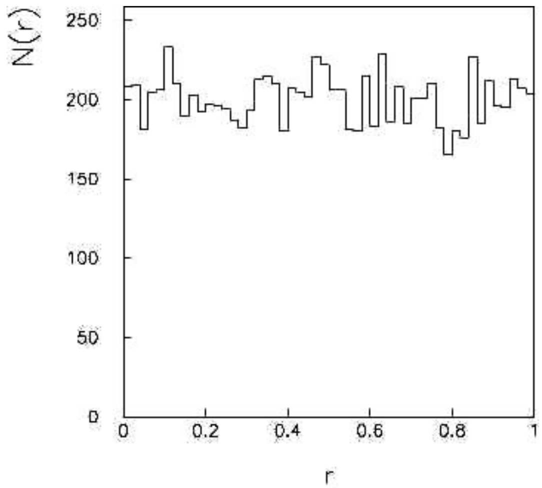
Choose a, m so that the r_i pass various tests of randomness:

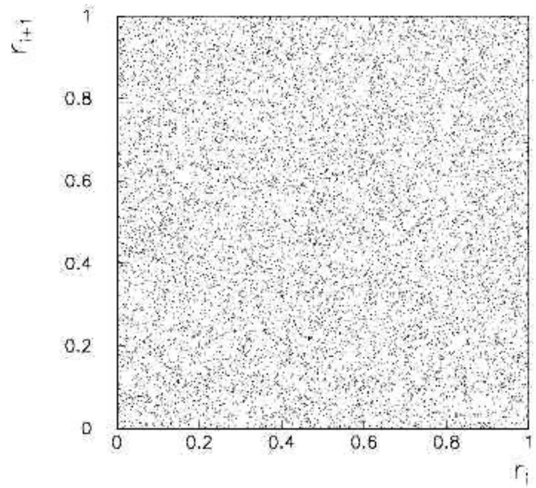
uniform distribution in [0, 1],

all values independent (no correlations between pairs),

e.g. L'Ecuyer, Commun. ACM **31** (1988) 742 suggests

$a = 40692$
$m = 2147483399$





Far better generators available, e.g. `TRandom3`, based on Mersenne twister algorithm, period = $2^{19937} - 1$ (a "Mersenne prime").

See F. James, Comp. Phys. Comm. 60 (1990) 111; Brandt Ch. 4

Randon number generators (3.1)

- Von Neumann algorithm
  r_0=0.9876 seed (initial number arbitrarily chosen); a change in the seed changes the random sequence
  $r_0^2=0.97535376$

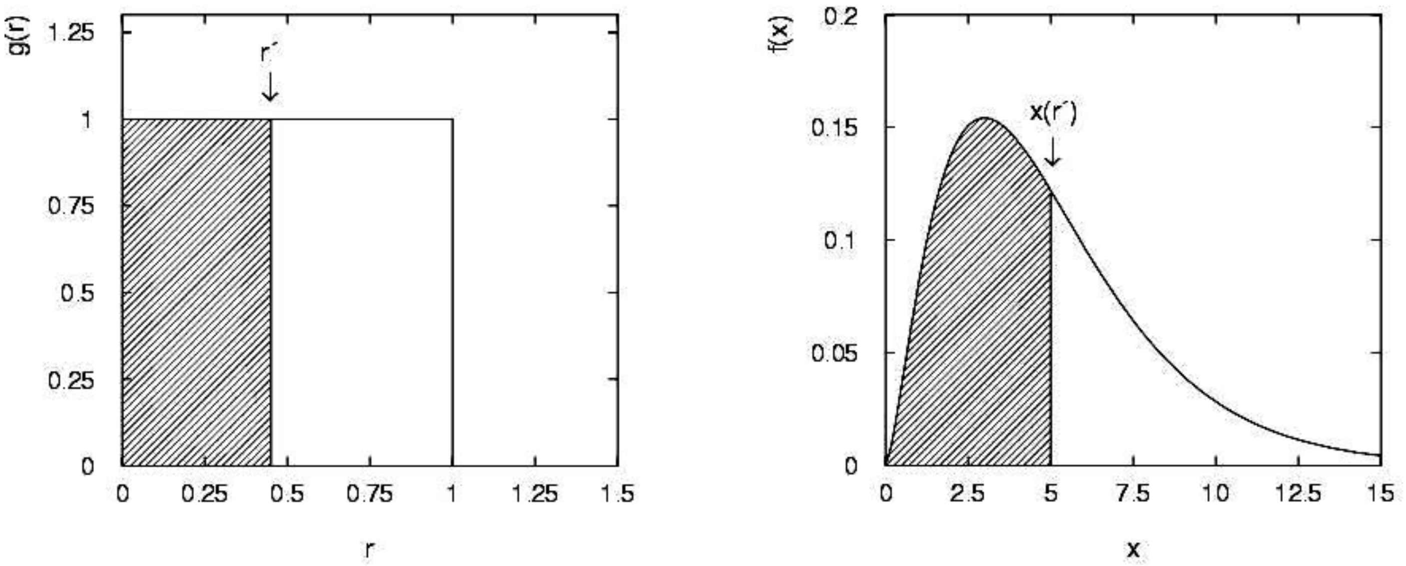
  r_1=0.5353
  $r_1^2=0.28654609$

  r_2=0.6546
  $r_1^2=\ldots\ldots$

- In general $r_{n+1}=F(r_n)$
- The algorithm produces a "deterministic" sequence
  => Pseudo random numbers

The transformation method

Given $r_1, r_2, \ldots, r_n$ uniform in [0, 1], find $x_1, x_2, \ldots, x_n$ that follow $f(x)$ by finding a suitable transformation $x(r)$.



Require: $P(r \leq r') = P(x \leq x(r'))$

i.e. $\displaystyle\int_{-\infty}^{r'} g(r)\,dr = r' = \int_{-\infty}^{x(r')} f(x')\,dx' = F(x(r'))$

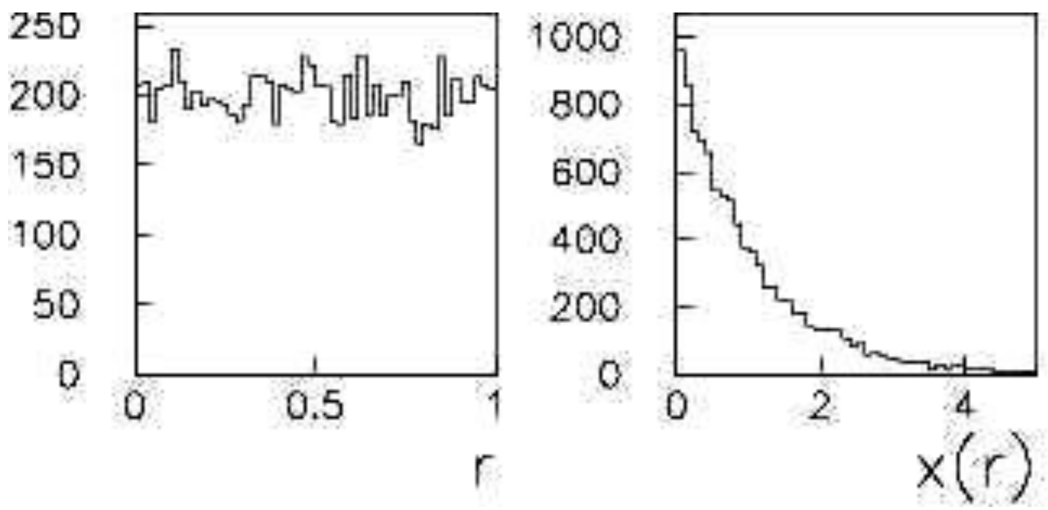
That is, $\boxed{\text{set } F(x) = r \text{ and solve for } x(r).}$

Example of the transformation method

Exponential pdf: $\quad f(x;\xi) = \frac{1}{\xi} e^{-x/\xi} \quad (x \geq 0)$

Set $\int_0^x \frac{1}{\xi} e^{-x'/\xi}\, dx' = r$ and solve for $x(r)$.

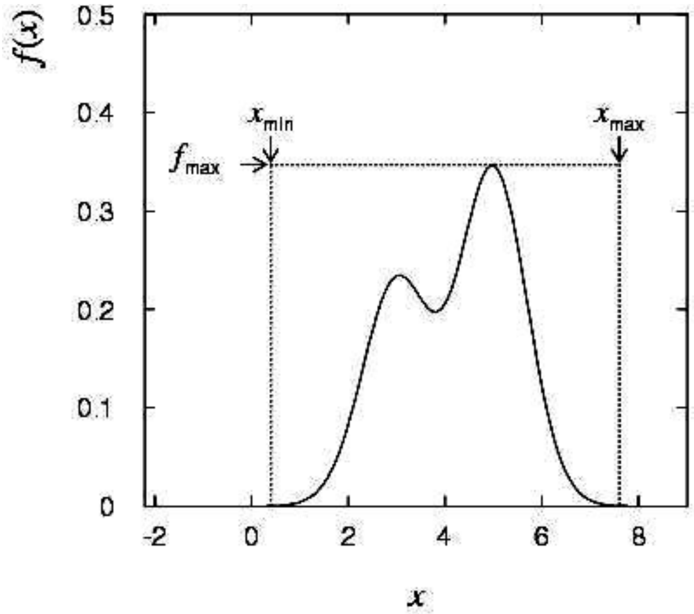
$\rightarrow \quad x(r) = -\xi \ln(1-r) \qquad$ ($x(r) = -\xi \ln r$ works too.)



1. Generation of random,
   Pseudo-random numbers
2. Random variable r uniformly distributed between 0 and 1
3. Sampling of a discrete random variable
   Example:
   A discrete random variable x with 3 values, x1, x2, x3 with probabilities P1, P2 and P3 respectively (Σ Pi=1).
   Extract y=r
   if $0<y<P1 \Rightarrow x=x1$
   if $P1<y<(P1+P2) \Rightarrow x=x2$
   if $(P2+P3)<y<1 \Rightarrow x=x3$
4. Sampling of a continuous random variable x with arbitrary pdf f(x)
   Extract y=r
   $x=F^{-1}(y)$ with $y=F(x)=\int_0^x f(x')dx'$
   Examples:
   $f(x)=1/(b-a) \Rightarrow x=a+(b-a)r$
   $f(\theta)=\sin\theta/2 \Rightarrow \cos\theta = 1-2r \Rightarrow \theta=\mathrm{acos}(1-2r)$
   $f(x)=\mu\exp(-\mu x) \Rightarrow x=-\ln(1-r)/\mu \Rightarrow x=-\ln(r)/\mu$

The acceptance-rejection method

Enclose the pdf in a box:



(1) Generate a random number x, uniform in $[x_{min}, x_{max}]$, i.e.
$x = x_{\min} + r_1(x_{\max} - x_{\min})$, r_1 is uniform in [0,1].

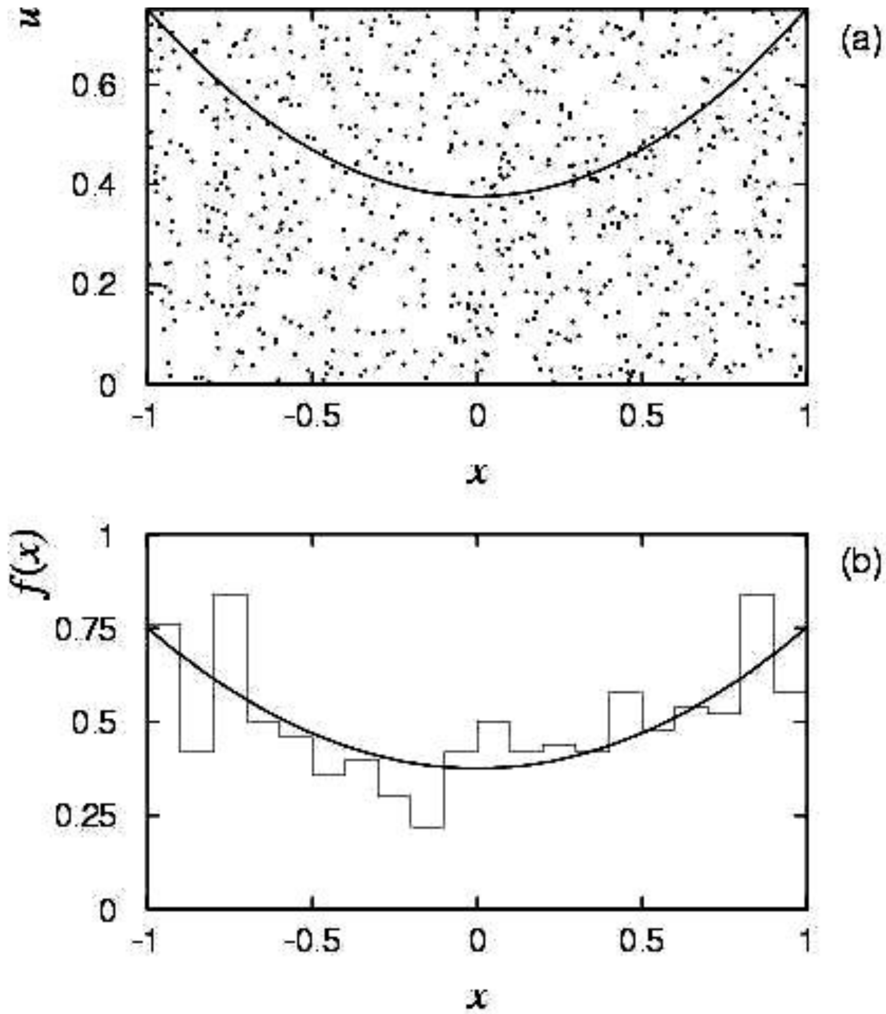
(2) Generate a 2nd independent random number u uniformly distributed between 0 and f_{max}, i.e. $u = r_2 f_{\max}$.

(3) If $u < f(x)$, then accept x. If not, reject x and repeat.

Example with acceptance-rejection method

$$f(x) = \frac{3}{8}(1 + x^2)$$

$$(-1 \leq x \leq 1)$$



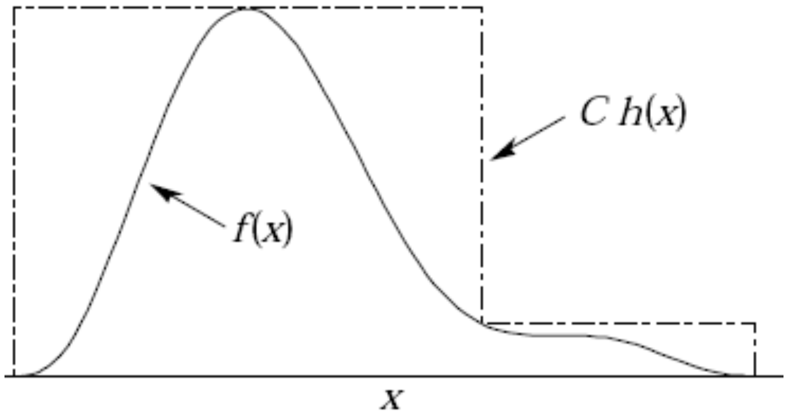
If dot below curve, use x value in histogram.

Improving efficiency of the acceptance-rejection method

The fraction of accepted points is equal to the fraction of the box's area under the curve.

For very peaked distributions, this may be very low and thus the algorithm may be slow.

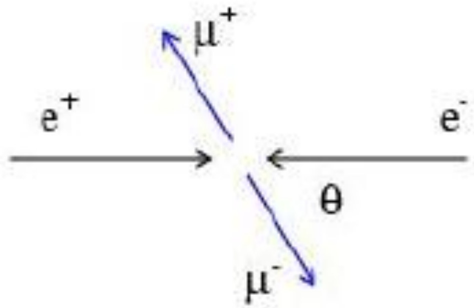
Improve by enclosing the pdf $f(x)$ in a curve $C\,h(x)$ that conforms to $f(x)$ more closely, where $h(x)$ is a pdf from which we can generate random values and C is a constant.



Generate points uniformly over $C\,h(x)$.

If point is below $f(x)$, accept x.

Monte Carlo event generators



Simple example: $e^+e^- \to \mu^+\mu^-$

Generate $\cos\theta$ and ϕ:

$$f(\cos\theta; A_{\mathrm{FB}}) \propto (1 + \frac{8}{3} A_{\mathrm{FB}} \cos\theta + \cos^2\theta) ,$$

$$g(\phi) = \frac{1}{2\pi} \quad (0 \le \phi \le 2\pi)$$

Less simple: 'event generators' for a variety of reactions:

$e^+e^- \to \mu^+\mu^-$, hadrons, ...

$pp \to$ hadrons, D-Y, SUSY,...

e.g. PYTHIA, HERWIG, ISAJET...

Output = 'events', i.e., for each event we get a list of generated particles and their momentum vectors, types, etc.

A simulated event

⋮

Event listing (summary)

I	particle/jet	KS	KF	orig	p_x	p_y	p_z	E	m
1	!p+!	21	2212	0	0.000	0.000	7000.000	7000.000	0.938
2	!p+!	21	2212	0	0.000	0.000	-7000.000	7000.000	0.938
3	!g!	21	21	1	0.863	-0.323	1739.862	1739.862	0.000
4	!ubar!	21	-2	2	-0.621	-0.163	-777.415	777.415	0.000
5	!g!	21	21	3	-2.427	5.486	1487.857	1487.	
6	!g!	21	21	4	-62.910	63.357	-463.274	471.	
7	!~g!	21	1000021	0	314.363	544.843	498.897	979.	
8	!~g!	21	1000021	0	-379.700	-476.000	525.686	980.	
9	!~chi_1-!	21	-1000024	7	130.058	112.247	129.860	263.	
10	!sbar!	21	-3	7	259.400	187.468	83.100	330.	
11	!c!	21	4	7	-79.403	242.409	283.026	381.	
12	!~chi_20!	21	1000023	8	-326.241	-80.971	113.712	385.	
13	!b!	21	5	8	-51.841	-294.077	389.853	491.	
14	!bbar!	21	-5	8	-0.597	-99.577	21.299	101.	
15	!~chi_10!	21	1000022	9	103.352	81.316	83.457	175.	
16	!s!	21	3	9	5.451	38.374	52.302	65.	
17	!cbar!	21	-4	9	20.839	-7.250	-5.938	22.	
18	!~chi_10!	21	1000022	12	-136.266	-72.961	53.246	181.	
19	!nu_mu!	21	14	12	-78.263	-24.757	21.719	84.	
20	!nu_mubar!	21	-14	12	-107.801	16.901	38.226	115.	
21	gamma	1	22	4	2.636	1.357	0.125	2.	
22	(~chi_1-)	11	-1000024	9	129.643	112.440	129.820	262.	
23	(~chi_20)	11	1000023	12	-322.330	-80.817	113.191	382.	
24	~chi_10	1	1000022	15	97.944	77.819	80.917	169.	
25	~chi_10	1	1000022	18	-136.266	-72.961	53.246	181.	
26	nu_mu	1	14	19	-78.263	-24.757	21.719	84.	
27	nu_mubar	1	-14	20	-107.801	16.901	38.226	115.	
28	(Delta++)	11	2224	2	0.222	0.012	-2734.287	2734.	

397	pi+	1	211	209	0.006	0.398	-308.296	308.297	0.140
398	gamma	1	22	211	0.407	0.087	-1695.458	1695.458	0.000
399	gamma	1	22	211	0.113	-0.029	-314.822	314.822	0.000
400	(pi0)	11	111	212	0.021	0.122	-103.709	103.709	0.135
401	(pi0)	11	111	212	0.084	-0.068	-94.276	94.276	0.135
402	(pi0)	11	111	212	0.267	-0.052	-144.673	144.674	0.135
403	gamma	1	22	215	-1.581	2.473	3.306	4.421	0.000
404	gamma	1	22	215	-1.494	2.143	3.051	4.016	0.000
405	pi-	1	-211	216	0.007	0.738	4.015	4.085	0.140
406	pi+	1	211	216	-0.024	0.293	0.486	0.585	0.140
407	K+	1	321	218	4.382	-1.412	-1.799	4.968	0.494
408	pi-	1	-211	218	1.183	-0.894	-0.176	1.500	0.140
409	(pi0)	11	111	218	0.955	-0.459	-0.590	1.221	0.135
410	(pi0)	11	111	218	2.349	-1.105	-1.181	2.855	0.135
411	(Kbar0)	11	-311	219	1.441	-0.247	-0.472	1.615	0.498
412	pi-	1	-211	219	2.232	-0.400	-0.249	2.285	0.140
413	K+	1	321	220	1.380	-0.652	-0.361	1.644	0.494
414	(pi0)	11	111	220	1.078	-0.265	0.175	1.132	0.135
415	(K_S0)	11	310	222	1.841	0.111	0.894	2.109	0.498
416	K+	1	321	223	0.307	0.107	0.252	0.642	0.494
417	pi-	1	-211	223	0.266	0.316	-0.201	0.480	0.140
418	nbar0	1	-2112	226	1.335	1.641	2.078	3.111	0.940
419	(pi0)	11	111	226	0.899	1.046	1.311	1.908	0.135
420	pi+	1	211	227	0.217	1.407	1.356	1.971	0.140
421	(pi0)	11	111	227	1.207	2.336	2.767	3.820	0.135
422	n0	1	2112	228	3.475	5.324	5.702	8.592	0.940
423	pi-	1	-211	228	1.856	2.606	2.808	4.259	0.140
424	gamma	1	22	229	-0.012	0.247	0.421	0.489	0.000
425	gamma	1	22	229	0.025	0.034	0.009	0.043	0.000
426	pi+	1	211	230	2.718	5.229	6.403	8.703	0.140
427	(pi0)	11	111	230	4.109	6.747	7.597	10.961	0.135
428	pi-	1	-211	231	0.551	1.233	1.945	2.372	0.140
429	(pi0)	11	111	231	0.645	1.141	0.922	1.608	0.135
430	gamma	1	22	232	-0.383	1.169	1.208	1.724	0.000
431	gamma	1	22	232	-0.201	0.070	0.060	0.221	0.000

⋮

PYTHIA Monte Carlo

pp $\rightarrow$ gluino-gluino

Monte Carlo detector simulation

Takes as input the particle list and momenta from generator.

Simulates detector response:

multiple Coulomb scattering (generate scattering angle),
particle decays (generate lifetime),
ionization energy loss (generate Δ),
electromagnetic, hadronic showers,
production of signals, electronics response, ...

Output = simulated raw data $\rightarrow$ input to reconstruction software:

track finding, fitting, etc.

Predict what you should see at 'detector level' given a certain hypothesis for 'generator level'. Compare with the real data.

Estimate 'efficiencies' = #events found / # events generated.

Programming package: `GEANT`

Monte Carlo integration method

- x uniform random variable in [a,b] (mean value technique):

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} f\left(x_i\right)=\frac{1}{b-a}\int_a^b f\left(x\right)dx$$

- Hit or miss method, x,y u.r.v., x in [a.b], y in [0,c]:

$$\int_a^b f\left(x\right)dx=\frac{N_{hit}}{N_{TOT}}c(b-a)$$

Monte Carlo integration method

- x uniform random variable in [a,b] (mean value technique):

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

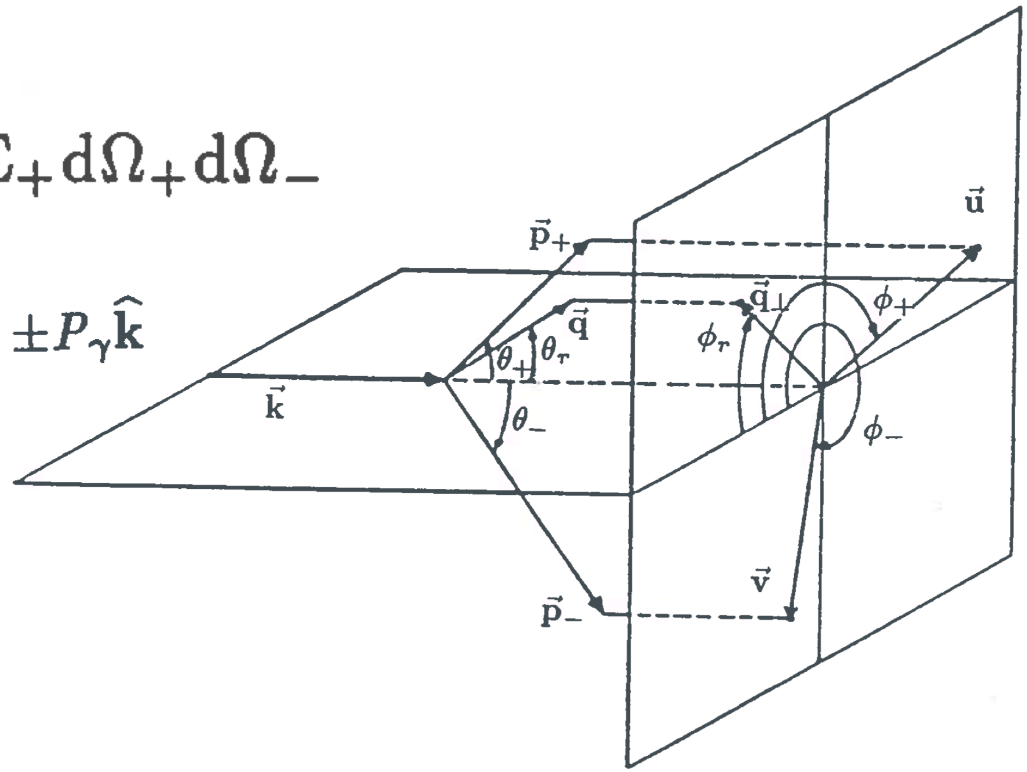
- Hit or miss method, x,y u.r.v., x in [a.b], y in [0,c]:

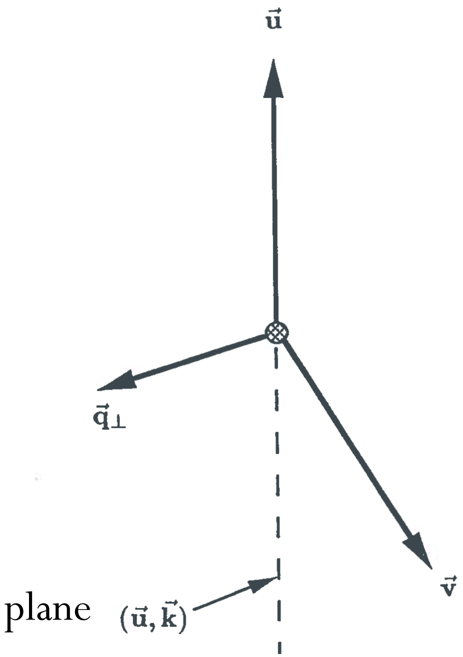
$$\int_a^b f(x)dx = \frac{N_{hit}}{N_{TOT}} c(b-a)$$

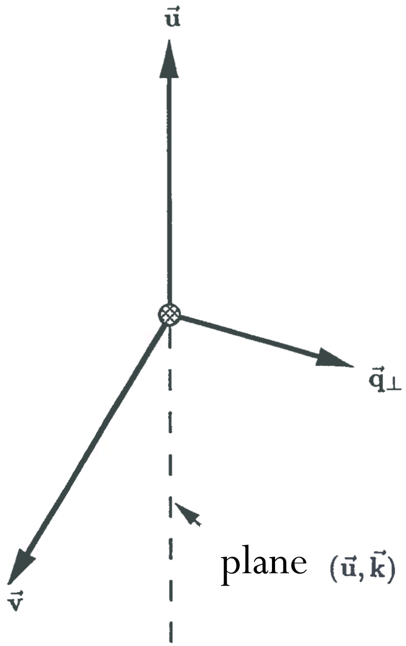
Differential pair production cross section from circularly polarized photons

$$d^5\sigma = (a + b\vec{\mathbf{P}}_\gamma \cdot \vec{\mathbf{n}})\mathrm{dE}_+\mathrm{d}\Omega_+\mathrm{d}\Omega_-$$

$$\vec{\mathbf{n}} = \vec{\mathbf{u}} \wedge \vec{\mathbf{v}} \qquad \vec{\mathbf{P}}_\gamma = \pm P_\gamma \hat{\mathbf{k}}$$





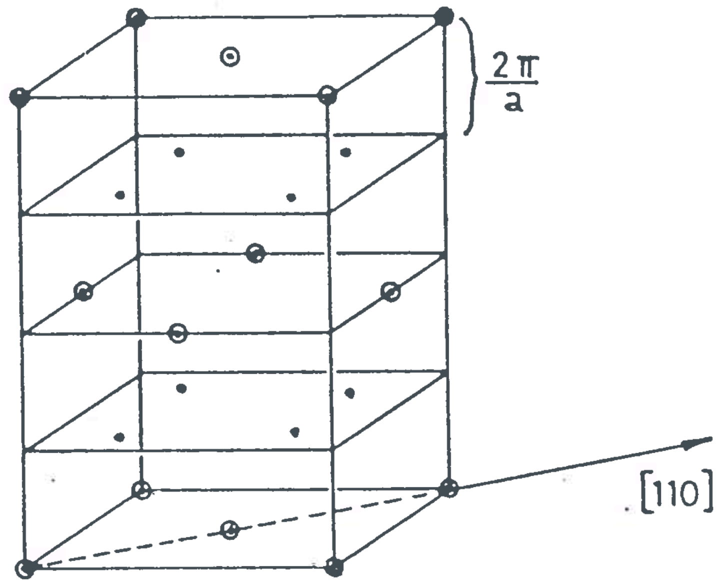


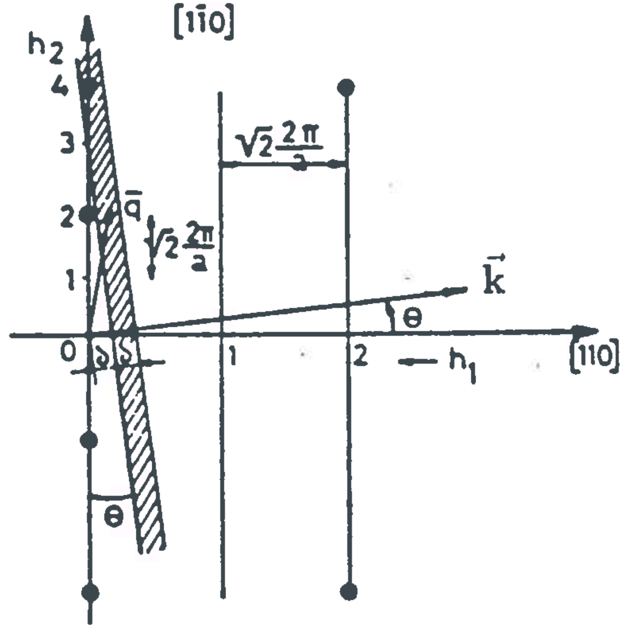
$$R = \frac{d^5\sigma(\vec{\mathbf{n}}) - d^5\sigma(-\vec{\mathbf{n}})}{d^5\sigma(\vec{\mathbf{n}}) + d^5\sigma(-\vec{\mathbf{n}})} = \vec{\mathbf{P}}_\gamma \cdot \vec{\mathbf{n}}\frac{a}{b}$$

$$d\sigma = \left|\sum_{\vec{\mathbf{L}}} e^{i\vec{\mathbf{q}}\cdot\vec{\mathbf{L}}}\right|^2 d\sigma(\vec{\mathbf{q}})$$

$$\left|\sum_{\vec{\mathbf{L}}} e^{i\vec{\mathbf{q}}\cdot\vec{\mathbf{L}}}\right|^2 \propto \sum_{\vec{\mathbf{g}}} \delta(\vec{\mathbf{q}} - \vec{\mathbf{g}})$$

$$\vec{\mathbf{q}} = \vec{\mathbf{g}}$$

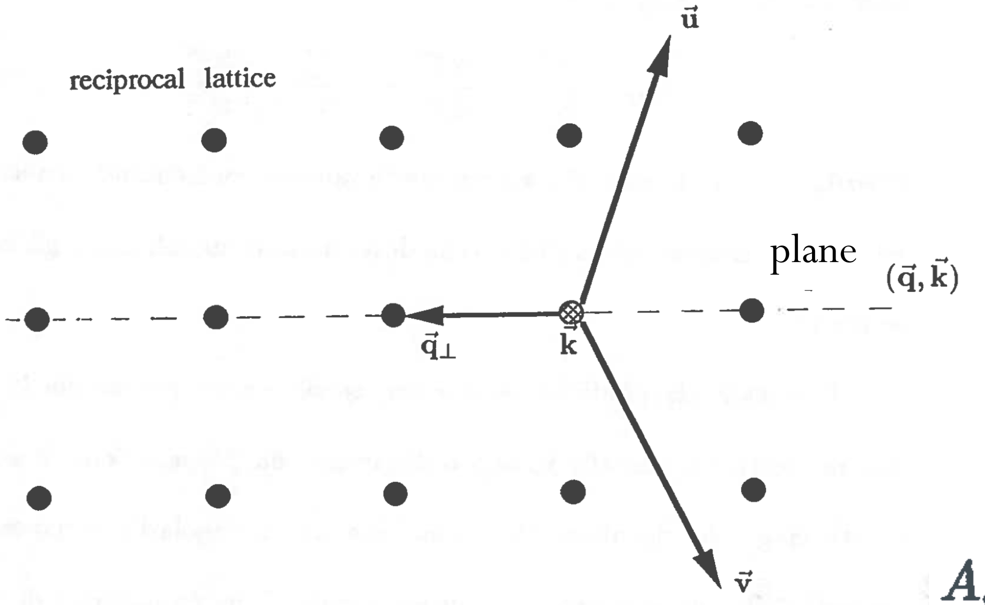


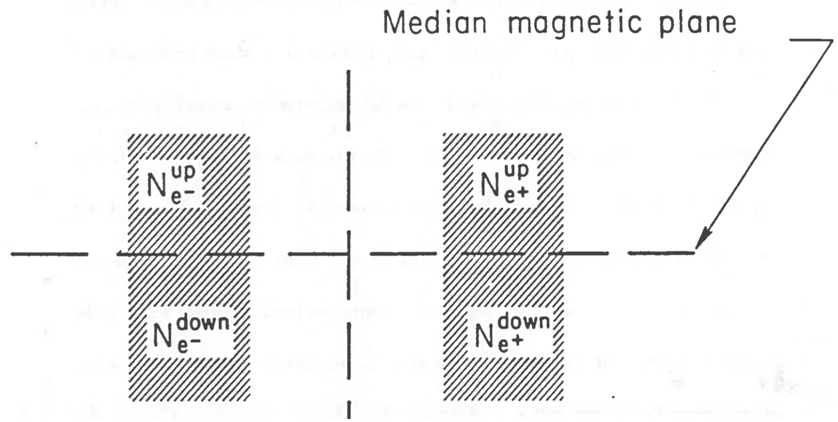


$$d^5\sigma(\vec{\mathbf{q}}) = \frac{d^5\sigma}{dE_+ d\Omega_+ d\Omega_-} |J| dq d\Omega_r dE_- d\phi_-$$

$$d\sigma_c^5 = \left|\sum_{\vec{\mathbf{L}}} \exp(i\vec{\mathbf{q}}\cdot\vec{\mathbf{L}})\right|^2 \exp(-A_T q^2) d\sigma^5(\vec{\mathbf{q}})$$

$$\left|\sum_{diam} \exp(i\vec{\mathbf{q}}\cdot\vec{\mathbf{L}})\right|^2 = \frac{1}{8} N \frac{(2\pi)^3}{\Delta} \sum_{\vec{\mathbf{g}}} D(\vec{\mathbf{g}}) \delta(\vec{\mathbf{q}} - \vec{\mathbf{g}})$$





$$A_{exp} = \frac{N^{sup}_{e^+} - N^{inf}_{e^+}}{N^{sup}_{e^+} + N^{inf}_{e^+}} = -\frac{N^{sup}_{e^-} - N^{inf}_{e^-}}{N^{sup}_{e^-} + N^{inf}_{e^-}}$$