# D. Monte Carlo methods

V. Blobel
Universität Hamburg

**Sommersemester 2002**

The Monte Carlo method: a numerical technique for calculating probabilities and related quantities using sequences of random numbers (it is equivalent to random experiments and to integration in perhaps many dimensions).

1. Random number generation
2. Generators for general distributions
3. Simulation
4. Integration

Note: Statistical accuracy depends on $\sigma \propto 1/\sqrt{n}$.



## D.1 Random number generation

Basis for Monte Carlo simulations are uniformly distributed random numbers in $[0,1]$.

$u_j \in U(0,1)$ uniform on (0,1) `u = ran()`

$$g(u) = \begin{cases} 1 & \text{if } 0 \leq u < 1 \\ 0 & \text{else} \end{cases} \qquad G(u) = \begin{cases} 0 & \text{if } u < 0 \\ u & \text{if } 0 \leq u < 1 \\ 1 & \text{if } u \leq 1 \end{cases} .$$

Methods:

- toss 32 coins for e.g. a 32 bit number $\rightarrow$ to tiring
- use computer algorithm to generate sequence of random numbers

## *Example: Linear congruential generator (LCG)*

Based on the recurrence relation

$$I_j = (a \cdot I_{j-1} + c) \mod m \qquad j = 1, 2, \ldots$$

integer random numbers $I_j$ between 0 and $m-1$ are generated.

Parameters: factor $a$ increment $c$ modulus $m$

Integer $I_j$ is converted to a real number $u = I_j/m$.
Values $u = 0$ and $u = 1$ usually excluded.
The maximum value $m$ of the period is reached under the following conditions:

1. increment $c \neq 0$;
2. the increment $c$ and the modulus $m$ do not have a common divisor;
3. $(a-1)$ is a multiple of the prime number $p$ for each prime number $p$, which is a divisor of the modulus $m$;
4. $(a-1)$ is a multiple of 4, if modulus $m$ is a multiple of 4.



## *Numerical example*

factor $a = 3$ increment $c = 0$ modulus $m = 7$

$$\begin{aligned}
I_0 &= 1 \quad \text{(start value = seed)} \\
I_1 &= (3 \cdot 1 + 0) \mod 7 = 3 \\
I_2 &= (3 \cdot 3 + 0) \mod 7 = 2 \\
I_3 &= (3 \cdot 2 + 0) \mod 7 = 6 \\
I_4 &= (3 \cdot 6 + 0) \mod 7 = 4 \\
I_5 &= (3 \cdot 4 + 0) \mod 7 = 5 \\
I_6 &= (3 \cdot 5 + 0) \mod 7 = 1 \quad \text{sequence repeats}
\end{aligned}$$

- start with seed $I_0$, which determines the whole sequence
- use different seeds in continued or repeated calculation (e.g. seed derived from daytime)
- use same seed during program development

## *Test of randomness*

Are these numbers really *random* numbers? $\rightarrow$ tests!

In a given calculation only a small fraction of the total period should be used ($\rightarrow$ large period required).

Deficiencies of the LCG:

- Low-order bits are less "random" than high-order bits.
- Sequential correlations exist: $d$ subsequent random numbers in $d$-dimensional space lie on $(d-1)$-dimensional hyperplanes
- there are at most $m^{1/d}$ such planes (Example $139968^{1/10} = 3.27$)
- maximum period reached only for certain parameter sets.

Good parameter sets (with $m$ = period):

| | | | | |
|---|---|---|---|---|
| $a$ | 205 | 281 | 1021 | 421 |
| $c$ | 29573 | 28411 | 25673 | 54773 |
| $m$ | 139968 | 134456 | 121500 | 259200 |

Some sets result in integer overflows (machine-dependence).



## *Improvements*

How to improve random number generation? Use *good* random number generators and combine $> 1$ generator!

Combination of two generators:

- fill array with random numbers of one random generator;
- use second random generator to select from array.

Good random number generators:

- lagged-Fibonacci based generators efficient and long period;
- RANLUX, based on the method of Lüscher, has a solid theoretical basis in chaos theory;
- Gutbrods generator (used in examples)

### *Test for generators*

Basic test is for uniform distribution (histogram).

Up-down- or sequence test: generate $n+1$ random numbers and determine number of sequences $N(k)$ of increasing or decreasing random numbers, satisfying

$$\sum_{k=1}^{n} k \cdot N(k) = n \qquad \sum_{k=1}^{n} N(k) = \frac{(2n-1)}{3} \, .$$

For truely uncorrelated random numbers $u_j$ the following values are expected:

$$N(1) = \frac{5n+1}{12} \qquad N(2) = \frac{11n-14}{60}$$

$$N(k) = \frac{2\left[\left(k^2+3k+1\right)n - \left(k^3+3k^2-k-4\right)\right]}{(k+3)!}$$

Especially the numbers $N(k)$ from larger values of $k$ allow sensitive tests of the random number generator.



## D.2 Generators for general distributions

MC simulation/integration requires random numbers from different distributions,

e.g. normal (also many-dimensional), exponential, Poisson ...

Methods

- inverse transform for continuous variable
- special methods
- inverse transform for discrete variable
- acceptance/rejection method

## *Inverse transform for continuous variable*

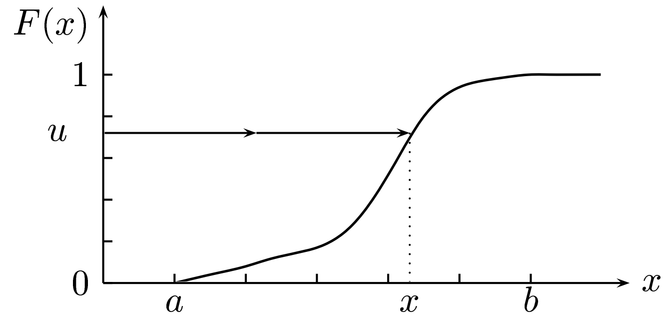
If $x$ is chosen from density $f(x)$, then the cumulative probability $F(x)$ is itself a rn which will occur with uniform density $g(u)$ on $[0,1]$:

$$f(x)\mathrm{d}x \;=\; g(u)\,\mathrm{d}u \qquad \int_{-\infty}^{x} f(x')\,\mathrm{d}x' \;= F(x) \;= u.$$

The equation $u = F(x)$ is solved for the variable $x$ by

$$x = F^{(-1)}(u)$$

with the inverse function $F^{(-1)}(u)$, provided the inverse function can be found.






## *Exponential distribution*

$f(x)\,\mathrm{d}x = \lambda e^{-\lambda x}\,\mathrm{d}x$ exponential distribution $x \in (0, +\infty)$

Inverse transform method:

$$\int_0^x f(x')\,\mathrm{d}x' = -e^{-\lambda x'}\Big|_0^x = F(x) = \boxed{1 - e^{-\lambda x} = u}$$

$$e^{-\lambda x} = 1 - u \qquad x = -\frac{1}{\lambda}\ln(1-u) \quad \text{or} \quad \boxed{x = -\frac{1}{\lambda}\ln u}$$

## *Random (uniform) angle and direction*

Azimuthal angle $\phi$: $\sin\phi$ and $\cos\phi$ of an uniformly distributed random angle $\phi$ is generated by the algorithm (without function call!)

1. Generate $u_1$ and $u_2 \in U(0,1)$; then $v_1 = 2u_1 - 1$, $v_2 = u_2$;
2. calculate $r^2 = v_1^2 + v_2^2$; if $r^2 > 1$, goto step 1 (i.e. accept points within a unit circle: $v_1/\sqrt{r^2} = \cos(\phi/2)$, $v_2/\sqrt{r^2} = \sin(\phi/2)$);
3. calculate

$$\boxed{\sin\phi = 2\frac{v_1 v_2}{r^2}} \qquad \boxed{\cos\phi = \frac{v_1^2 - v_2^2}{r^2}}$$

based on

$$\sin\phi = 2\cos\frac{\phi}{2}\cdot\sin\frac{\phi}{2} \qquad \cos\phi = \cos^2\frac{\phi}{2} - \sin^2\frac{\phi}{2}$$

Polar angle $\theta$: $\cos\theta = 2u - 1$ corresponds to distribution uniform in $\cos\theta$.

Random unit vector: $(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$ (note: all components follow the same distribution – test!)



## *Normal distribution*

A simple approximate method based on Central limit theorem:

$$z = \sum_{i=1}^{12} u_i - 6 \qquad \text{add 12 random numbers } \; u_i \in U(0,1)$$

Box-Muller method: starts from $\chi^2$ variable with 2 degrees of freedom, which has an exponential distribution (using ln and sqrt).

1. Generate two random numbers $u_1$ and $u_2$ and construct $v_1 := 2u_1 - 1$ and $v_2 := 2u_2 - 1$.
2. calculate $r^2 = v_1^2 + v_2^2$; if $r^2 > 1$, goto step 1 (unit circle);
3. calculate the two independent random numbers

$$z_1 = v_1\sqrt{\frac{-2\ln r^2}{r^2}} \qquad z_2 = v_2\sqrt{\frac{-2\ln r^2}{r^2}}$$

The random number $z$ is transformed to a random number $x$ from the normal distribution with mean $\mu$ and standard deviation $\sigma$ by $x = \mu + \sigma z$.

## *Normal distribution in n dimensions*

Density

$$f(\boldsymbol{x}) = \frac{|\boldsymbol{W}|^{1/2}}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_x})^T\, \boldsymbol{W}\, (\boldsymbol{x}-\boldsymbol{\mu_x})\right)$$

with $\boldsymbol{W} = \boldsymbol{V}^{-1}$ and $|\boldsymbol{W}| = \det\ \boldsymbol{W} = |\boldsymbol{V}^{-1}|$.

Random number generation based on Cholesky decomposition of $\boldsymbol{V} = \boldsymbol{R}^T\boldsymbol{R}$ or $\boldsymbol{W} = \boldsymbol{R}^{-1}(\boldsymbol{R}^{-1})^T$ ($\boldsymbol{R}$ = right-triangular metrix).

$$\boldsymbol{x} = (\boldsymbol{\mu}+)\ \boldsymbol{R}^T\, \boldsymbol{z}$$

where $\boldsymbol{z}$ = vector of independent random numbers from the normal distribution $N(0,1)$.

---

Proof, assuming $\boldsymbol{\mu} = 0$: $\quad \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{R}^{-1}(\boldsymbol{R}^{-1})^T\boldsymbol{x} = \boldsymbol{z}^T\mathbf{1}\boldsymbol{z}$

$$\boldsymbol{z} = (\boldsymbol{R}^{-1})^T\boldsymbol{x} \quad \rightarrow \quad \boldsymbol{x} = \boldsymbol{R}^T\boldsymbol{z}$$

---



## *Normal distribution in 2 dimensions*

Covariance matrix $\boldsymbol{V}$ and right triangular matrix $\boldsymbol{R}$:

$$\boldsymbol{V} = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} = \boldsymbol{R}^T\boldsymbol{R} \qquad \boldsymbol{R} = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

Result of decomposition:

$$R_{11} = \sqrt{V_{11}} = \sigma_1 \qquad R_{12} = \frac{V_{11}}{\sqrt{V_{11}}} = \rho\sigma_2 \qquad R_{22} = \sqrt{V_{22} - R_{12}^2} = \sigma_2\sqrt{1-\rho^2}$$

Generate $z_1$, $z_2 \in N(0,1)$ (uncorrelated) and transform:

$$\begin{aligned} x_1 &= \mu_1 + \sigma_1 z_1 \\ x_2 &= \mu_2 + \sigma_2\left(\rho z_1 + \sqrt{1-\rho^2}\, z_2\right) \end{aligned}$$

resulting random values $x_1$ and $x_2$ are correlated.

---

## *Example: Multiple scattering through small angles*

Charged particle traversing a medium of thickness $x/X_0$ is deflected by many small-angle scatters. Projected (plane) angular distribution approximately given by

$$\begin{aligned} f(\theta_{\text{plane}}) &= \frac{1}{\sqrt{2\pi}\theta_0} \exp\left(-\frac{\theta^2_{\text{plane}}}{2\theta_0^2}\right) \mathrm{d}\theta_{\text{plane}} \\ \text{with} \quad \theta_0 &= \frac{13.6\,\text{MeV}}{\beta c p} \sqrt{\frac{x}{X_0}} \left[1 + 0.038 \ln\left(\frac{x}{X_0}\right)\right] \end{aligned}$$

Transverse displacement $y$ is correlated to angle $\theta$ with correlation coefficient $\rho_{y\theta} = \sqrt{3}/2 \approx 0.87$.

Monte Carlo generation of a joint $(y, \theta)$ distribution using $z_1$, $z_2 \in N(0,1)$:

$$\begin{aligned} y_{\text{plane}} &= z_1\, x\, \theta_0/\sqrt{12} + z_2\, \rho_{y\theta}\, x\, \theta_0/\sqrt{3} \\ \theta_{\text{plane}} &= z_2 \theta_0 \end{aligned}$$



## *Cholesky decomposition*

The matrix $\boldsymbol{V}$ is transformed to the matrix $\boldsymbol{R}$ in steps. Starting with $\boldsymbol{V}^{(0)} = \boldsymbol{V}$ the transformation is

$$V_{ij}^{(k)} = V_{ij}^{(k-1)} - R_{ki} R_{kj} \qquad i,\, j = k+1,\, k+2,\, \dots\, n\,.$$

for $k = 1,\, 2,\, \dots\, (n-1)$. First $k = 1$ and the elements with $i,\, j = 2,\, 3, \dots\, n$ are transformed, then $k = 2$ and the elements with $i,\, j = 3,\, 4, \dots\, n$ are transformed by

$$R_{ii} = \sqrt{V_{ii}^{(i-1)}} \qquad R_{ij} = \frac{V_{ij}^{(i-1)}}{R_{ii}} \quad \text{für} \quad j > i\,.$$

The relation

$$V_{jj} = \sum_{i=1}^{j} R_{ij}^2 \qquad j = 1,\, 2, \dots\, n$$

shows that the elements of the Cholesky factors $\boldsymbol{R}$ are limited by the positive diagonal elements of the matrix $\boldsymbol{V}$.

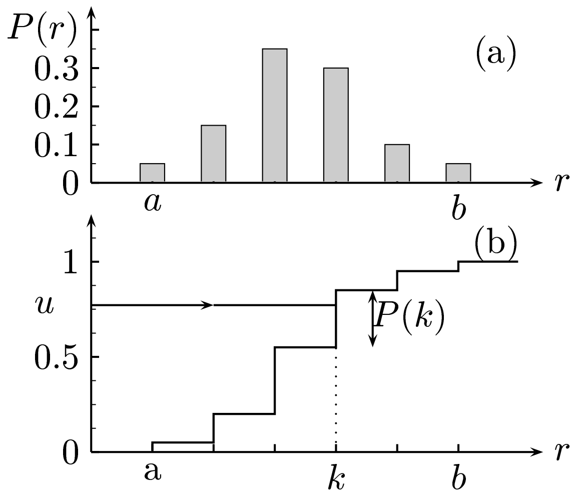
## *Inverse transformation method for discrete variable*

In a discrete distribution an index $k$ is assigned to the discrete random values with $k = 1, 2, \ldots$. The discrete values $x_k$ occurs with probability $P(x_k)$.

Chose random number $u \in U(0,1)$ and find index $k$ with

$$P_{k-1} < u \leq P_k$$

where $P_0 = 0$.






## *Poisson distribution (e.g simulation of histogram entries)*

$$P_\mu(k) = \frac{\mu^k}{k!} e^{-\mu} \qquad k = 0, 1, \ldots \qquad \text{parameter} = \mu$$

Method based on sum of exponentially distributed variables:

$$e^{-\mu} < \prod_{i=1}^{k+1} u_i \quad \leftarrow \quad \mu < -\sum_{i=1}^{k+1} \ln u_i$$

Algorithm for random number generation of integer $k$:

1. Set $k := 0$ and $A := 1.0$;
2. generate a random number $u$ and replace $A$ by $u \cdot A$: $A := u \cdot A$; if $A \leq e^{-\mu}$, accept $k$ and stop
3. increase $k$ by 1: $k := k + 1$ and goto step 2

## *Poisson variable, contnd.*

For larger values of $\mu$ like $\mu = 100$ the method makes no sense, because $e^{-\mu}$ may be already outside the floating point range of the computer.

Already for $\mu > 10 \ldots 20$ it is often better to use the normal approximation of the Poisson distribution and to generate a random variable $z$ from $N(0,1)$; the real random number is then converted to the random integer $k$ by $k = \max\left(0, \left[\mu + z\sqrt{\mu} + 0.5\right]\right)$.



## *Acceptance-rejection method*

Obtaining the inverse function $F^{(-1)}(u)$ is often impossible, although the density function $f(x)$ can be computed for any possible value of $x$. The method due to von Neumann is based on the relation

$$f(x) \leq C \cdot h(x) \qquad \text{for all } x$$

where $h(x)$ is a density, where the random values $x$ are easily generated; the factor $C$ is a (positive) constant. The algorithm is as follows:

1. Choose a random number $x$ according to distribution $h(x)$;
2. generate $u \in U(0,1)$ (uniformly distributed);
3. if the product $u \cdot (C \cdot h(x)) > f(x)$, reject $x$ and repeat step 1; otherwise accept $x$.

Both $f(x)$ and $h(x)$ are normalized $\rightarrow$ factor $C > 1$. The acceptance probability (= efficiency) in the last step is equal to the ratio of the areas (under $f(x)$ and $h(x)$) which is $1/C < 1$. The factor $C$ will not be much larger than 1, if the shape of $h(x)$ is similar to the shape of $f(x)$; in this case the efficiency of the algorithm will be close to 100 %.

## *Acceptance-rejection method contnd.*

The distribution $h(x)$ may be even a *uniform* distribution between the limits $a$ and $b$ of $x$ and in this case one has to choose the constant $C$ such that $f(x) \leq C$ for all $x$.

Then the algorithm is:

1. Choose $x_i$ uniformly distributed between $a$ und $b$:
   $x = a + u_1 \cdot (b - a)$ with $u_1 \in U(0,1)$;
2. choose another random number $u_2 \in U(0,1)$.
3. if the product $u_2 \cdot C > f(x)$, goto step 1, otherwise accept $x$.

The random value $x$ is accepted with probability $\propto f(x)$.



## *Acceptance-rejection method for weighted events*

Assume events are generated with statistical *non-constant weights* $w$.

Further event calculation requires some time and will be inefficient for events with small weight.

Method: determine upper limit $C$ for weight, i.e. $C \geq w_{\max}$.

1. Choose random number $u \in U(0,1)$.
2. Reject event, if $w < u \cdot C$, otherwise accept and assign weight 1.

Events are accepted with probability $\propto w$.

## D.3 Integration

Many excellent methods exist for numerical integration with high accuracy in one dimension:

- trapezoidal rule
- ...
- Gauss quadrature

Numerical integration in space of high dimension is often difficult because of complicated integration region.

Monte Carlo methods become relatively efficient in high dimension and allow to work with complicated integration regions.



*Integral by trapezoidal rule and by MC*

Approximation of the integral of the function $g(x)$

$$\mathrm{I} = \int_a^b g(x)\,\mathrm{d}x$$

- Numerical integration: trapezoidal rule with $(n+1)$ points

$$I_{TR} = \frac{b-a}{n} \sum_{j=0}^{n} w_j g(x_j) + R_n \qquad R_n = -\frac{1}{12}\frac{(b-a)^3}{n^2}\tilde{g}''$$

- Monte Carlo method: random $x \in U(a,b)$

$$I = (b-a)\int_a^b g(x)\cdot\frac{1}{(b-a)}\,\mathrm{d}x = (b-a)\,E[g(x)]$$

Monte-Carlo estimate $I_{\mathrm{MC}}$ with $n$ random values $x_i$

$$I_{\mathrm{MC}} = \frac{b-a}{n}\sum_{i=1}^{n} g(x_i) \qquad x_i \in U(a,b)$$

## *Accuracy of Monte Carlo estimate*

Variance of MC estimate by standard statistical methods:

$$V[I_{MC}] = V\left[\frac{b-a}{n}\sum_{i=1}^{n} g_i\right] = \left(\frac{b-a}{n}\right)^2 V\left[\sum_{i=1}^{n} g_i\right] = \frac{(b-a)^2}{n} V\left[g_i\right] \propto 1/n$$

The standard deviation has, in agreement with the law of large numbers, a dependence $\propto 1/\sqrt{n}$.

Factor $V\left[g_i\right]$ may be large:

$$V\left[g_i\right] = E\left[(g_i - \bar{g})^2\right] \approx \left[\frac{\sum g_i^2}{n} - \left(\frac{\sum g_i}{n}\right)^2\right] \qquad \text{estimate}$$

Variance reducing methods discussed later.



## *Numerical integration in d dimensions*

- Numerical integration on unit hypercube $(0,1)^d$ with trapezoidal rule in $d$ dimensions, applying trapezoidal rule with $n+1$ points in each dimension:

  $$I_{TR} = \frac{1}{n^d}\sum_{j_1=0}^{n}\sum_{j_2=0}^{n}\cdots\sum_{j_d=0}^{n} w_{j_1}w_{j_2}\cdots w_{j_d}\cdot g\left(\frac{j_1}{n}, \frac{j_2}{n}, \ldots \frac{j_d}{n}\right) + \mathcal{O}\left(\frac{1}{n^2}\right)$$

  where the function $g(x_1, x_2, \ldots x_d)$ has to be evaluated $N = (n+1)^d \approx n^d$ times. Although the deviation is still proportional to $1/n^2$, the dependence on the number $N$ of function evaluations is $N^{-2/d}$.
- Monte Carlo method: $\sigma \propto 1/\sqrt{N}$ for $N$ function evaluations

Cross-over point: $N^{-2/d} = N^{-1/2}$ $\rightarrow$ Monte Carlo "better" for $d > 4$

Complicated integration boundaries are simpler for the Monte Carlo method than for numerical methods.

## *Variance reducing methods*

However, the convergence of the Monte Carlo estimate $I_{\text{MC}}$ by a rate of $1/\sqrt{N}$ is slow.

The accuracy of results depends in addition on the variance $V[g]$ of the function and techniques to reduce the variance are important.

- Importance sampling
- Control variates
- Stratified sampling



## *Importance sampling*

The method corresponds to a change of the variable. The integral of the function $g(x)$ in the intervall $(a, b)$ is written in the form

$$I = \int_a^b g(x)\,\mathrm{d}x = \int_a^b \left[\frac{g(x)}{f(x)}\right] f(x)\,\mathrm{d}x = E\left[\frac{g(x)}{f(x)}\right] ,$$

with a probability density $f(x)$, normalized in the interval $(a, b)$. Assuming that the density $f(x)$ can be integrated ($F(x)$) and the inverse $F^{(-1)}(u)$ can be found, the expectation value of the function $r(x) = g(x)/f(x)$ for probability density $f(x)$ can easily be determined with Monte Carlo methods.

$$I_{\text{MC}} = \frac{1}{n}\sum_{i=1}^{n}\frac{g(x_i)}{f(x_i)} \quad \text{with} \quad x_i \in (a,\, b) \quad \text{from density } f(x) \,.$$

Importance sampling improves the accuracy of a Monte Carlo integration, if a density $f(x)$ can be found with a small variance of $r(x) = g(x)/f(x)$

$$V[r] = E\left[(r(x) - \overline{r})^2\right]$$

## *Control variates*

In this method for variance reduction a function $h(x)$ is subtracted from the function $g(x)$, which can be integrated analytically:

$$\int_a^b g(x)\,\mathrm{d}x = \int_a^b h(x)\,\mathrm{d}x + \int_a^b \left[g(x) - h(x)\right]\,\mathrm{d}x\,,$$

The first integral is calculated analytically, the second one by Monte Carlo methods.

The function $h(x)$ should be chosen to reduce the variance of the second integral.



## *Stratified sampling*

The full integration space is divided into subspaces and a Monte Carlo integration is performed in each subspace; the partial results are added up in the end. The method is based on the property

$$\int_a^b g(x)\,\mathrm{d}x = \int_a^c g(x)\,\mathrm{d}x + \int_c^b g(x)\,\mathrm{d}x \qquad a < c < b$$

for the Riemann-Integral.

For a $d$-**dimensional integral** the integration region is divided into $k$ subregions $M_j$ with $j = 1, 2, \ldots k$. A subregion has a volume $V(M_j)$ and in each subregion the Monte Carlo integration is done with $N_j$ points. The Monte Carlo estimate of the complete integral is

$$I_{\mathrm{MC}} = \sum_{j=1}^{k} \frac{V(M_j)}{N_j} \sum_{i=1}^{N_j} g(\boldsymbol{x}_{ji})\,.$$

*Stratified sampling contnd.*

The variance of the integral is, instead of $\sigma^2(g)/N$, given by the expression

$$V[I_{\mathrm{MC}}] = \sum_{j=1}^{k} \frac{V^2(M_j)}{N_j} \, \sigma^2(g)\big|_{M_j} \; ,$$

where the $\sigma^2(g)\big|_{M_j}$ are the variances in the subregions.

If the subregions and the numbers $N_j$ are carefully chosen, the variance can be reduced compared to the simple Monte Carlo integral. The variances in the subregions should be small, and the number of points in each subregion should be proportional to the product of volume $V(M_j)$ and standard deviation $\sigma(g)|_{M_j}$ in each subregion.

In *adaptive* Monte Carlo methods the subregions are optimized during the calculation, and the number of points $N_j$ in each subregion is optimized (e.g $\rightarrow$ CERN program library).



## D.4 Simulation

Simulation of physical processes by Monte Carlo methods

Example from High Energy Physics: Generators for *ep*-interactions

Results are "events", i.e. list of final state particles and their momentum vectors.

**ARIADNE** A program for simulation of QCD cascades implementing the color dipole model.

**AROMA** Heavy quark production in boson-gluon fusion using full electroweak LO cross-sections (with quark masses) in ep collisions, DIS and photoproduction. Parton showers and Lund hadronization gives full events.

**COMPTON** Event generator of quasi-real QED Compton events in ep collisions.

## *Event generators contnd.*

**DIFFVM** DIFFVM is a program to simulate soft diffractive processes in ep-collisions in the framework of Regge phenomenology and the vector meson dominance model. It includes a detailed simulation of the diffractive dissociation of the proton.

**DIPSI** A Monte Carlo generator for elastic vector meson production in charged lepton-proton scattering.

**DISENT** DISENT is a program for (1+1)- and (2+1)-jet events in deep inelastic scattering in NLO.

**DISASTER** DISASTER is a program for (1+1)- and (2+1)-jet events in deep inelastic scattering in NLO.

**DJANGO** Interface between HERACLES (QED corrections to order alpha QED) and LEPTO/ARIADNE (QCD matrix elements and parton showers/colour dipole radiation) to give complete ep events.



## *Event generators contnd.*

**EPJPSI** EPJPSI is a program to simulate elastic and inelastic vector meson production in photo production and deep inelastic scattering.

**GRAPE-Dilepton** This program is an event generator for dilepton production in ep collisions. It is based on exact matrix elements calculated automatically by GRACE. The program is interfaced to PYTHIA to generate a complete final state.

**HERACLES** NC and CC ep interactions (using parameterizations of structure functions or parton densities) with radiative corrections: single photon emission from the lepton line, self energy corrections and the complete set of one-loop weak corrections can be included.

## *Event generators contnd.*

**HERWIG** General purpose generator for Hadron Emission Reactions With Interfering Gluons; based on matrix elements, parton showers including colour coherence within and between jets, and a cluster model for hadronization.

**HITVM** A Monte Carlo program of calculations of Forshaw and Ryskin for diffractive vector meson production using a gluon ladder in pQCD using BFKL formalism.

**JETSET** The Lund string model for hadronization of parton systems.



## *Event generators contnd.*

**JETVIP** NLO program for calculating one and two-jet cross sections for electron-protron and electron-positron collisions. The virtuality of the photon emitted by the electron can be chosen in a continuous region, ranging from on-shell photons (photoproduction) up to very large virtualities (deep-inelastic scattering region). The program enables to calculate the direct and resolved photon processes in NLO QCD. Special attention is given to the resolved virtual photon component.

**JBFKL** A Monte Carlo generator for J/psi meson production by off-shell gluons.

**LDCMC** A program which implements the Linked Dipole Chain (LDC) model for deeply inelastic scattering within the framework of ARIADNE. The LDC model is a reformulation of the CCFM model.

*Event generators contnd.*

**LEPTO** Deep inelastic lepton-nucleon scattering based on LO electroweak cross sections (incl. lepton polarization), first order QCD matrix elements, parton showers and Lund hadronization giving complete events. Soft colour interaction model gives rapidity gap events.

**LPAIR** This event generator is devoted to the process of electromagnetic production of lepton-pairs in lepton-lepton, lepton-hadron or hadron-hadron interactions via the two-photon process.

**MAJOR** Production and decay of heavy Majorana neutrinos in ep collisions, via mixing or exchange of righthanded W-bosons. Parton showers and Lund hadronization added to give complete events.

**MEPJET** NLO Monte Carlo program to study jet cross sections for arbitrary jet algorithms and arbitrary event shapes in deep inelastic e- p and e+ p scattering.



*Event generators contnd.*

**PHOJET** Multiparticle production in high energy hadron-hadron, photon-hadron, and photon-photon interactions (hadron = proton, antiproton, neutron, or pion).

**POMPYT** Diffractive hard scattering in pp, gamma-p and ep collisions, based on pomeron flux and pomeron parton densities (several options included). Also pion exchange is included. Parton showers and Lund hadronization to give complete events.

**PYTHIA** General purpose generator for e+e-, pp and ep-interactions, based on LO matrix elements, parton showers and Lund hadronization.

**QCDINS** QCDINS is a Monte Carlo package for simulating QCD-instanton induced scattering processes in deep-inelatic lepton-nucleon scattering. It is designed as an 'add-on' hard process generator for the Monte Carlo generator HERWIG. Optionally, an interface to JETSET is also available for the final hadronization step. This allows to examine the effects of different hadronization schemes on the instanton-induced final state.

*Event generators contnd.*

**RAPGAP** A full Monte Carlo suited to describe Deep Inelastic Scattering, including diffractive DIS and LO direct and resolved processes.

**RIDI** A Monte Carlo generator for inelastic diffraction dissociation of virtual photon in ep scattering, based on a model by M. Ryskin. Transverse and longitudinal cross sections are simulated. Processes implemented: ep -¿ epX with X = q+qbar or X = q+qbar+gluon. Fragmentation and parton shower done with JETSET.

**SUSYGEN** Event Generator for SUSY processes in e+e-, mu+ mu-, ep and pp collisions. In ep collisions, allows to simulate the resonant production of R-parity violating squarks undergoing any (direct or cascade) decay. SUSY processes at hadronic colliders are not included in this pre-release of SUSYGEN 3.

**WABGEN** An event generator for the simulation of elastic and inelastic wide angle bremsstrahlung, including the simulation of the complete final state.



*Tools*

Last, but not least . . .

**PDFLIB** Nucleon, Pion and Photon Parton Density Functions and alpha(s) Calculations.

## *Monte Carlo detector simulation*

Aim:

- design and optimize the detectors
- develop and test the reconstruction and analysis programs, and
- interpret the experimental data

Takes as input the list of particles and their momenta from event generators.

Simulate the detector response

- description of geometry and material of the detector elements
- tracking of particles through the detector
- simulate their physical interactions in matter and the effects of fields and boundaries on their trajectory
- reproduce the electronic response of the sensitive detectors

Output is simulated "raw data" = input to reconstruction and analysis programs

**Example**: GEANT4 program for detector simulation, developed by a large collaboration of many institutes.



## *Example: MC generation of an electromagnetic shower*

Processes with electrons and photons:

$$\begin{array}{llll} \gamma\, X & \longrightarrow & e^+\, e^-\, X & \text{Pair creation} \\ e\, X & \longrightarrow & e\, \gamma\, X & \text{Bremsstrahlung} \end{array}$$

Cross sections, probabilities etc required and detector geometry.

Here simple example of homogeneous medium.

## *Pair creation – cross section*

$$\gamma X \longrightarrow e^+ e^- X$$

The probability for pair creation with positron energy between $E_+$ and $E_+ \mathrm{d}E_+$ by a photon of energy $k$ traversing a medium of thickness $\mathrm{d}x$, is given by

$$\Phi(k, E_+)\, \mathrm{d}E_+\, \mathrm{d}x = \frac{\mathrm{d}x}{X_0} \cdot \frac{\mathrm{d}E_+}{k} \cdot g(k, E_+)$$

with the function $g(k, E_+)$ given by

$$g(k, E_+) \simeq v^2 + (1-v)^2 + \frac{2}{3} v \cdot (1-v) \qquad \text{ratio} \quad v = \frac{E_+}{k}$$

... defines partition of photon energy between positron and electron.



## *Pair creation – MC generation*

Generation of actual photon path length and of electron/positron energy

$$\text{Mean free path } = \frac{7}{9} X_0 \quad \text{(from integration)}$$

Actual path follows exponential distribution:

$$\Delta s = -\frac{7}{9} X_0 \cdot \ln u \qquad u \in U(0,1)$$

Generate ratio $v = E_+/k$:

$$F(v) = \frac{1}{7/9} \int_0^v g(k, E_+)\, \mathrm{d}v = \frac{1}{7/9} \left( v - \frac{2}{3} v^2 + \frac{4}{9} v^3 \right) = u \qquad u \in U(0,1)$$

... and solve equation for $v$, which defines the electron/positron energies.

## *Bremsstrahlung – cross section*

$$e\,X \longrightarrow e\,\gamma\,X$$

The probability for emission of a photon with energy between $k$ and $k + \mathrm{d}k$ by an electron (energy $E$) traversing a medium of thickness $\mathrm{d}x$, is given by

$$\Phi(E,k)\,\mathrm{d}k\,\mathrm{d}x = \frac{\mathrm{d}x}{X_0} \cdot \frac{\mathrm{d}k}{k} \cdot F(E,k)$$

with the function $F(E,k)$ given by $E'$ =energy of outgoing electron, $k = E - E'$

$$F(E,k) \simeq 1 + \left(\frac{E'}{E}\right)^2 - \frac{2}{3}\frac{E'}{E}$$



## *Bremsstrahlung – MC generation*

Cross section $\propto 1/k$ diverging – make cut-off and transform:

$$y = \frac{k}{E} \qquad x = \ln y \qquad \frac{\mathrm{d}x}{\mathrm{d}y} = \frac{1}{y}$$

$$\frac{\mathrm{d}k}{k} = \frac{\mathrm{d}k/E}{k/E} = \frac{\mathrm{d}y}{y} = \mathrm{d}x$$