Statistical Methods in Particle Physics

4. Monte Carlo Methods

Heidelberg University, WS 2023/24

Klaus Reygers, Martin Völkl (lectures)
Ulrich Schmidt, (tutorials)

Monte Carlo method

- Any method which solves a problem by generating suitable random numbers
- Useful for obtaining numerical solutions to problems which are too complicated to solve analytically
- The most common application of the Monte Carlo method is Monte Carlo integration
- Pioneers
  - Enrico Fermi
  - Stanislaw Ulam
  - John von Neumann
  - Nicholas Metropolis

https://en.wikipedia.org


Enrico Fermi


Stanislaw Ulam


J. von Neumann


N. Metropolis

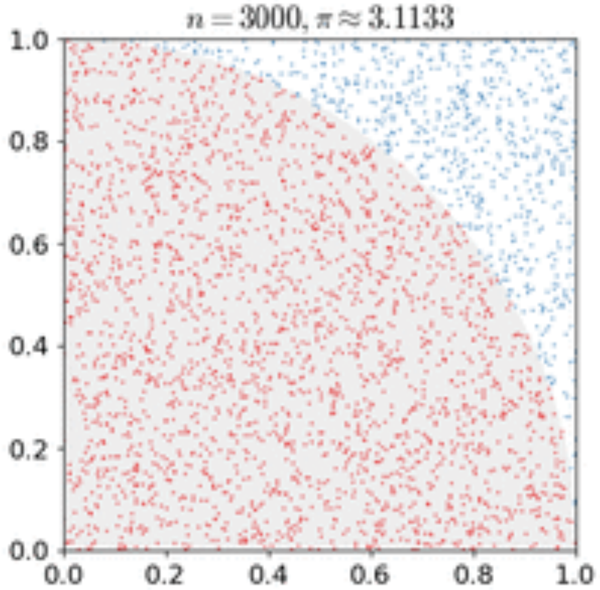
http://mathworld.wolfram.com/MonteCarloMethod.html

Monte Carlo method: Examples

[from Bohm, Zech: Introduction to Statistics and Data Analysis for Physicists]

- Area of a circle
- Volume of the intersection of a cone and a torus
  - Hard to solve analytically
  - Easy to solve by scattering points homogeneously inside a cuboid containing the intersect
- Efficiency of particle detection with a scintillator
  - Produced photons are reflected at the surfaces and sometime absorbed
  - Almost impossible to calculate analytically for different parameters like incident angle, particle energy, …
  - Monte Carlo simulation is the only sensible approach
- Complicated function $f(x_1, x_2, \ldots, x_n)$, what is the marginal

$$f_j(x_j) = \int \ldots \int f \, \mathrm{d}x_1 \ldots \mathrm{d}x_{j-1} \, \mathrm{d}x_{j+1} \ldots \mathrm{d}x_n \, ?$$



These problems are easy if we can just sample from the relevant distributions.

Pseudo-random numbers

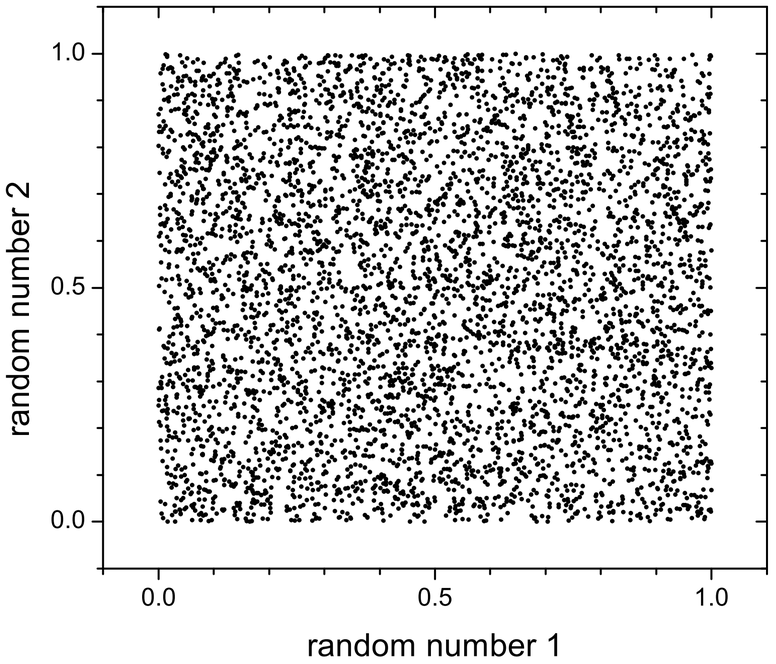
- Principle: Use insignificant digits of an operation to generate next number
  - choose large integers λ and m, $\lambda < m$
  - choose integer $n_0 < m$ (“seed”)
  - uniformly distributed random numbers r_i:

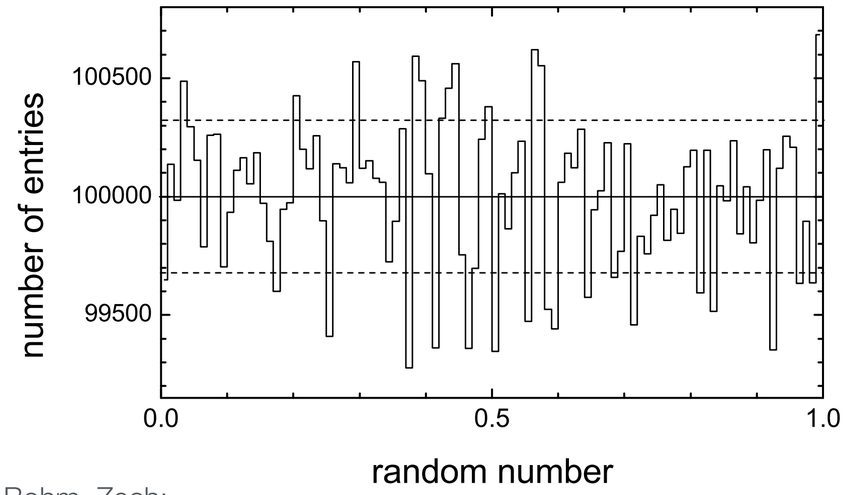
$$n_{i+1} = \lambda n_i \bmod m$$

$$r_i = n_i / m, \quad r_i \in [0, 1]$$

“Multiplicative linear congruential algorithm“
(period at maximum m – 1)

- *Mersenne twister*
  - Invented 1997 by M. Matsomoto and T. Nishimura
  - Sequence repeats after 2^{19937} calls, i.e., never …
- Quality checks
  - Frequency of occurrence
  - Plot correlations between consecutive random numbers





Bohm, Zech:
http://www-library.desy.de/preparch/books/vstatmp_engl.pdf

Random Numbers from distributions: Inverse transform method

Consider a distribution f from which we want to draw random numbers. Let $u(r)$ be the uniform distribution in [0, 1]:

$$\int_{-\infty}^{x} f(x')\,\mathrm{d}x' = \int_{0}^{r(x)} u(r')\,\mathrm{d}r' = r(x)$$

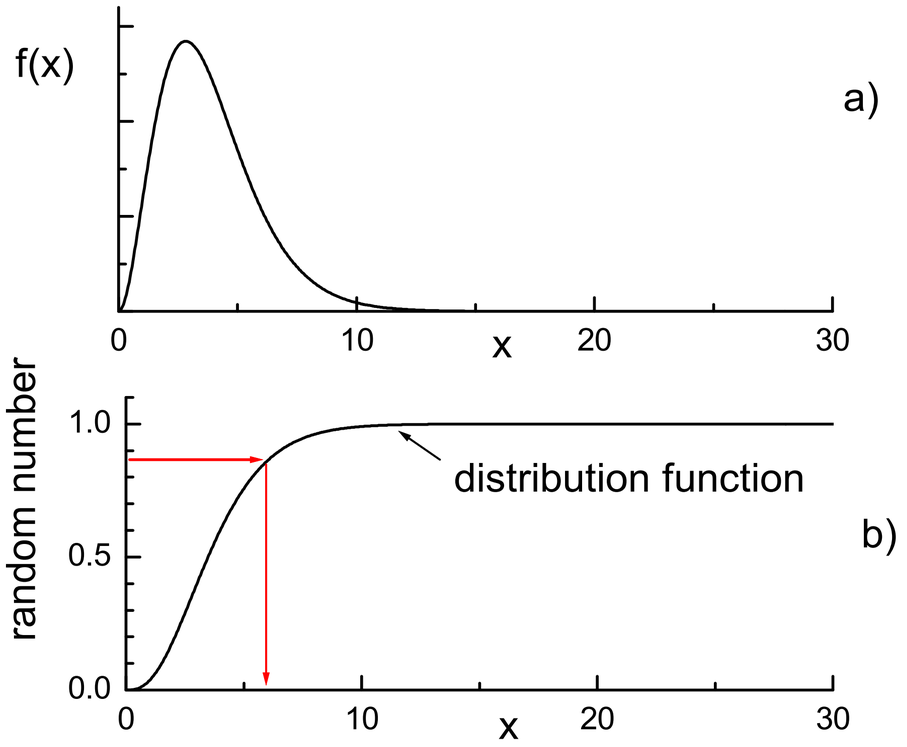
With $F(x)$ = cumulative distr.:

$$F(x) = r$$

We get the random number x from the inverse of the cumulative distribution:

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

Bohm, Zech:
http://www-library.desy.de/preparch/books/vstatmp_engl.pdf



Cross check:

$$\frac{\mathrm{d}p}{\mathrm{d}x} = \underbrace{\frac{\mathrm{d}p}{\mathrm{d}r}}_{=1}\frac{\mathrm{d}r}{\mathrm{d}x} = \frac{\mathrm{d}F(x)}{\mathrm{d}x} = f(x) \qquad \square$$

Example I

Linear function:

$$f(x) = 2x, \qquad 0 \leq x \leq 1$$
$$F(x) = x^2 \quad \rightarrow \quad x = \sqrt{r}$$

Exponential:

$$f(x) = \gamma e^{-\gamma x}, \qquad x \geq 0$$
$$F(x) = 1 - e^{-\gamma x} \quad \rightarrow \quad x = -\frac{\ln(1-r)}{\gamma}$$

One can store *F*(x) as a histogram if there is no analytical solution, cf. root's `GetRandom()` function:

```
root [0] TF1 f("f", "x^3/(exp(x)-1)", 0., 15.);
root [1] cout << f.GetRandom() << endl;
13.9571
```

Box-Muller algorithm for creating Gaussian distributed random numbers

1. Generate two uniformly distributed random numbers u_1 and u_2 in the range [0,1]

2. Set

$$\phi = 2\pi u_1, \qquad r = \sqrt{-2 \ln u_2}$$

3. Then

$$z_1 = r \cos\phi \quad \text{and} \quad z_2 = r \sin\phi$$

are two independent rv's following a standard normal distribution

Why?

$$u_2(r) = e^{-\frac{r^2}{2}}$$

$$\frac{dp}{dr} = \frac{dp}{du_2} \cdot \left|\frac{du_2}{dr}\right| = e^{-\frac{r^2}{2}} r$$

$$dp = \underbrace{\frac{1}{2\pi} e^{-\frac{r^2}{2}} r \, dr \, d\phi}_{\text{2d standard normal distribution in polar coordinates}} = \underbrace{\frac{1}{2\pi} e^{-\frac{z_1^2+z_2^2}{2}} dz_1 \, dz_2}_{\text{2d standard normal distribution in cartesian coordinates}}$$

Inverse transform method using histograms in Python

```python
def get_random(f, xmin, xmax, n_samples):
    """Generate n_samples random numbers within range [xmin, xmax]
    from arbitrary continuous function f
    using inverse transform sampling
    """

    # number of points for which we evaluate F(x)
    nbins = 10000

    # indefinite integral F(x), normalize to unity at xmax
    x = np.linspace(xmin, xmax, nbins+1)
    F = integrate.cumtrapz(f(x), x, initial=0)
    F = F / F[-1]

    # interpolate F^{-1} and evaluate it for
    # uniformly distributed rv's in [0,1[
    inv_F = interpolate.interp1d(F, x, kind="quadratic")
    r = np.random.rand(n_samples)
    return inv_F(r)
```

[random_numbers_from_distribution.ipynb]

Example II: Uniform points on a sphere

$$\frac{\mathrm{d}p}{\mathrm{d}\Omega} = \frac{\mathrm{d}p}{\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi} = \mathrm{const} \equiv k \qquad \frac{\mathrm{d}p}{\mathrm{d}\theta\,\mathrm{d}\phi} = k\sin\theta \equiv f(\phi)g(\theta)$$

Distributions for θ and ϕ:

$$f(\phi) \equiv \frac{\mathrm{d}p}{\mathrm{d}\phi} = \mathrm{const} = \frac{1}{2\pi}, \qquad 0 \leq \phi \leq 2\pi$$

$$g(\theta) \equiv \frac{\mathrm{d}p}{\mathrm{d}\theta} = \frac{1}{2}\sin\theta, \qquad 0 \leq \theta \leq \pi$$

Calculating the inverse of the cumulative distribution we obtain:

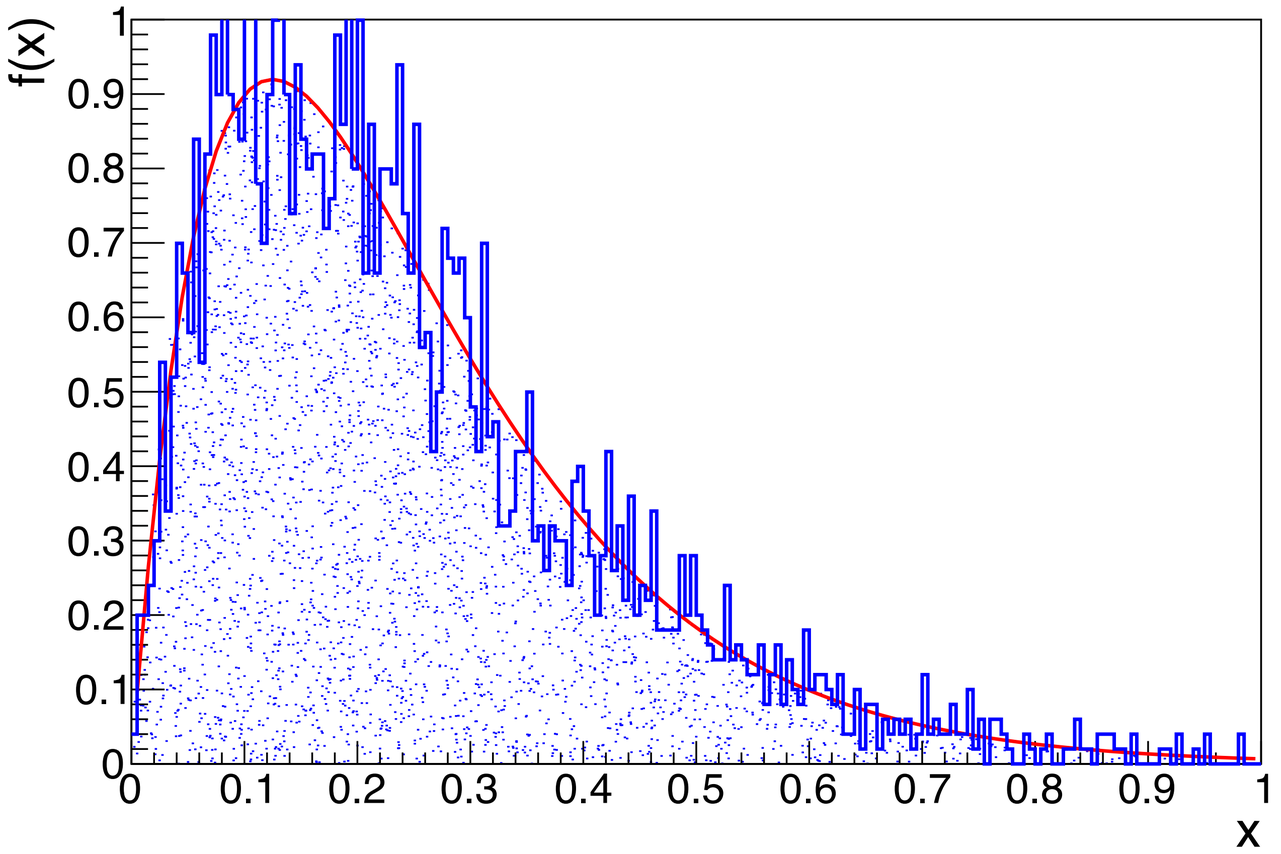
$$\phi = 2\pi r_1$$

$$\theta = \arccos(1 - 2r_2) \qquad [\text{as } G(\theta) = \frac{1}{2}(1 - \cos\theta)]$$

Upshot: ϕ and cos θ need to be distributed uniformly

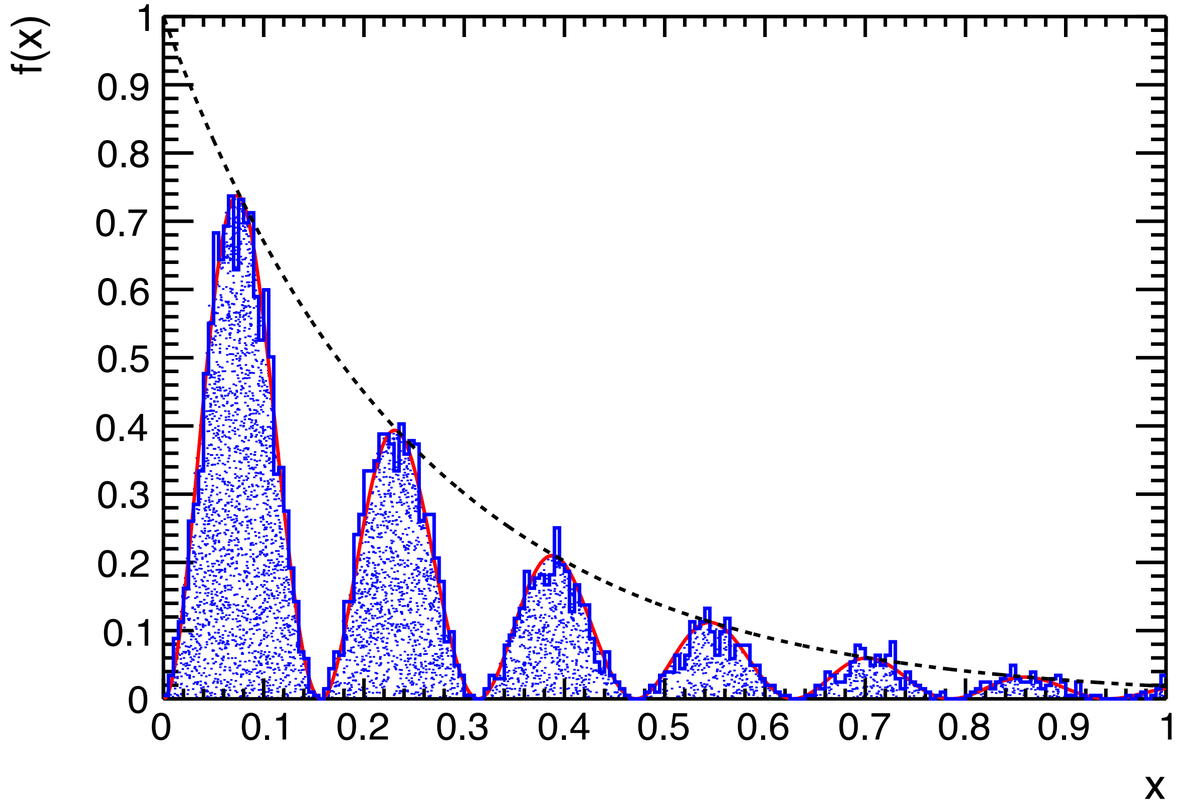
Random numbers from distributions: Acceptance-rejection method

- Idea:
  - Create flat distribution in x,y
  - Use points below function value
  - X values are distributed like the distribution
- Algorithm
  - Generate random number *x* uniformly between *a* and *b*
  - Generate another number $y \in [0,1]$
  - For some number A larger than the maximum of the function: accept if $y < f(x)/A$
  - Repeat many times



Random numbers from distributions: Acceptance-rejection method (II)

- The efficiency of this algorithm can be quite small (e.g. long, small tails)
- Find another distribution p_m that is easy to sample, and where $f_m = \alpha p_m$ is always larger than the target distribution for some α
- This is called a *majorant*
- Now sample from p_m and accept points with the condition $y < f(x)/f_m(x)$



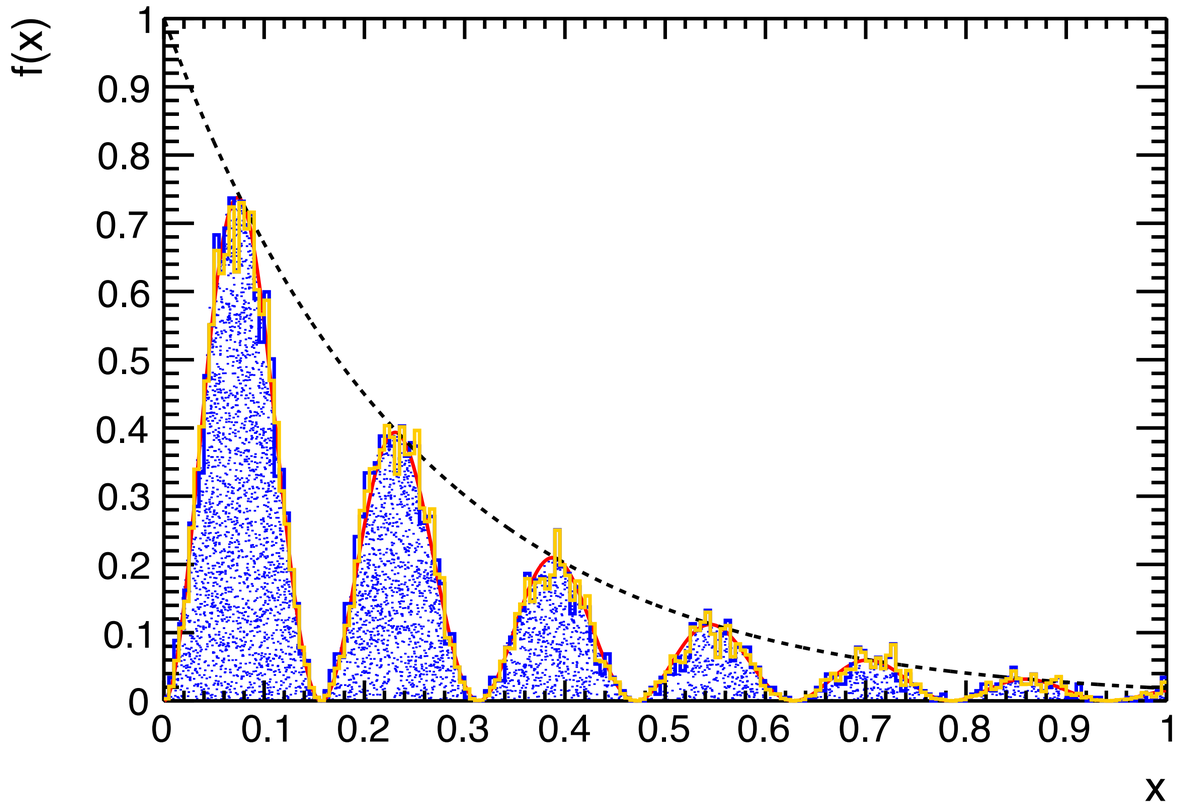
- (Example: $\sin^2(c_1 x)\ \exp(-c_2 x)$ is hard to sample from, but the majorant $\exp(-c_2 x)$ is easy to sample)

Importance Sampling

- Almost the same as accept-reject
- Instead of accepting point with probability $f(x)/f_m(x)$, always accept, but with weight

$$w = f(x)/f_m(x)$$

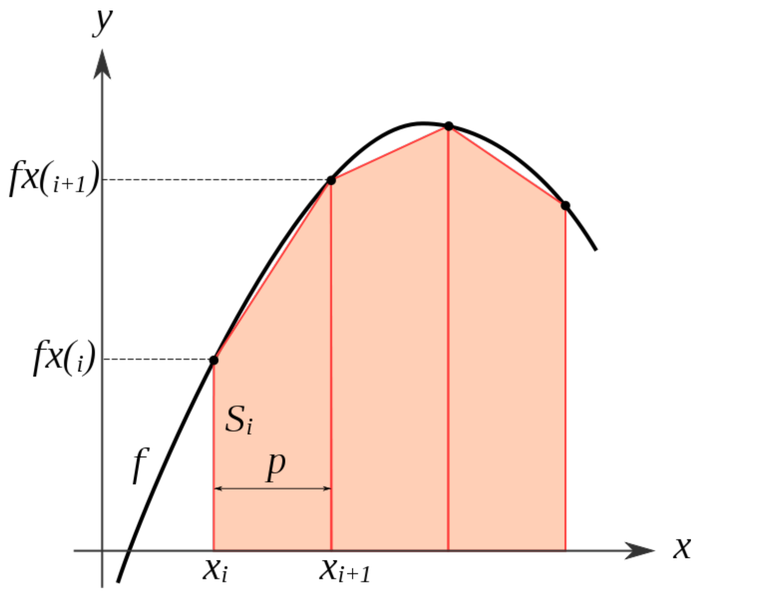
- Comparison function does not need to be a majorant



The curse of dimensionality

Trapezoidal rule in one dimension

- accuracy improves as $1/n^2$ with the number of points



https://en.wikipedia.org/wiki/Trapezoidal_rule

Monte Carlo integration in *d* dimensions:

- Averages, fractions from independent points all scale with $1/\sqrt{n}$
- n is accepted number of points

Trapezoidal rule in *d* dimension:

- accuracy improves as $1/n^{2/d}$ with the number of points
- for $d > 4$ the dependence on n is better for MC integration
- (However, fraction of accepted points tend to lower a bit with higher dimensions)

same as in 1d case

For multidimensional integrals MC integration outperforms other numerical integration methods

Metropolis-Hastings algorithm (1)

Bayesian inference often involves marginalization of a high-dimensional posterior distribution:

$$P(\theta_0|\text{data}) = \int P(\theta_0, \theta_1, ..., \theta_n|\text{data})\, d\theta_1 ... d\theta_n$$

Typically, the integral cannot be solved in closed form. Moreover, repeated one-dimensional integration becomes inefficient ("curse of dimensionality").

Idea: sample distribution many times and consider only parameter of interest.

Method: Markov Chain Monte Carlo (MCMC)

MCMC has revolutionized Bayesian analysis.

A sequence of random numbers is a Markov chain if the probability of the next number only depends on the previous one:

$$f(x_{n+1}|x_n, x_{n-1}, ..., x_0) = f(x_{n+1}|x_n)$$

Metropolis-Hastings algorithm (2)

Goal: sample from a distribution $f(\vec{x})$ known up to a normalization constant.

Take initial $\vec{x}_0$ with $f(\vec{x}_0) > 0$ and repeat the following steps many times:

1. Generate candidate $\vec{y}$ according to proposal distribution $q(\vec{y} \,|\, \vec{x}_k)$
2. Generate uniformly distributed random number r in [0, 1] and set

$$\vec{x}_{k+1} = \begin{cases} \vec{y}, & \text{if } r \leq \alpha(\vec{x}_k, \vec{y}) \\ \vec{x}_k, & \text{otherwise} \end{cases}$$

where

$$\alpha(\vec{x}, \vec{y}) = \min \left\{ 1, \frac{f(\vec{y})\, q(\vec{x} \,|\, \vec{y})}{f(\vec{x})\, q(\vec{y} \,|\, \vec{x})} \right\}$$

$\alpha(\vec{x}, \vec{y})$ is called the acceptance probability.

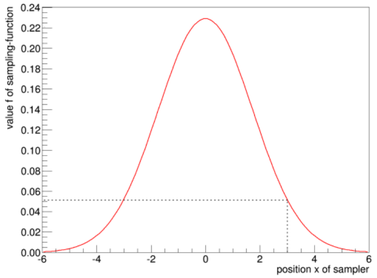
Metropolis-Hastings algorithm (3)

- The algorithm generates a *correlated* sequence of points (not suited for many applications, but okay for marginalization)
- If a finite initial sequence of points is discarded, the remaining points can be shown to follow $f(\vec{x})$
- Not easy to figure out when the sequence has started to converge to $f(\vec{x})$
- The proposal function $q(\vec{y}\,|\,\vec{x})$ can be almost anything. Often, a multi-dimensional Gaussian is used.
- Often the proposal function is symmetric, i.e., $q(\vec{y}\,|\,\vec{x}) = q(\vec{x}\,|\,\vec{y})$. Then the acceptance probability reduces to

$$\alpha(\vec{x},\vec{y}) = \min\left\{1, \frac{f(\vec{y})}{f(\vec{x})}\right\}$$

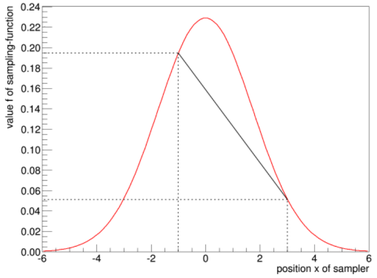
  and a step to a higher $f(\vec{y})$ is always taken.
- Original Metropolis algorithm suggested symmetric proposal functions, Hastings modified original rules by using non-symmetric functions.

Example



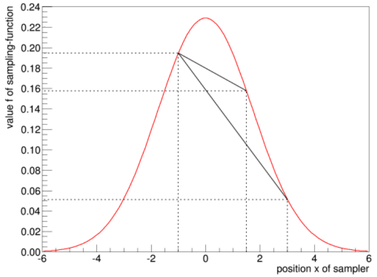
$x_{start} = 3$

Figure 1.5: A starting point is chosen.



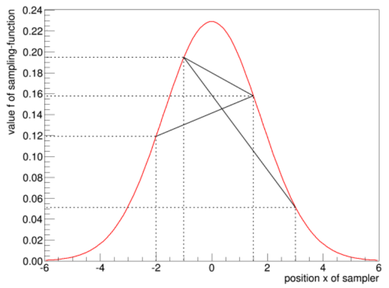
$$x_{old} = x_{start} = 3$$
$$p = -4$$
$$x_{new} = x_{old} + p = -1$$
$$\rho = \min\left(1, \frac{f(x_{new})}{f(x_{old})}\right) = \min(1, 3.75) = 1$$
$$u = 0.9$$
$$\rho > u \Rightarrow \textbf{accept}$$

Figure 1.6: For $f(x_{new}) > f(x_{old})$ the step is always accepted.



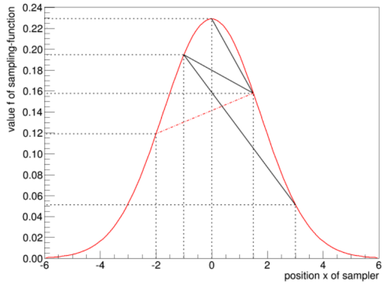
$$x_{old} = -1$$
$$p = 2.5$$
$$x_{new} = x_{old} + p = 1.5$$
$$\rho = \min\left(1, \frac{f(x_{new})}{f(x_{old})}\right) = \min(1, 0.81) = 0.81$$
$$u = 0.4$$
$$\rho > u \Rightarrow \textbf{accept}$$

Figure 1.7: For $f(x_{new}) < f(x_{old})$ it depends on u whether a step is accepted.



$$x_{old} = 1.5$$
$$p = -3.5$$
$$x_{new} = x_{old} + p = -2$$
$$\rho = \min\left(1, \frac{f(x_{new})}{f(x_{old})}\right) = \min(1, 0.75) = 0.75$$
$$u = 0.8$$
$$\rho < u \Rightarrow \textbf{reject}$$

Figure 1.8: For $\rho < u$ the step is rejected.



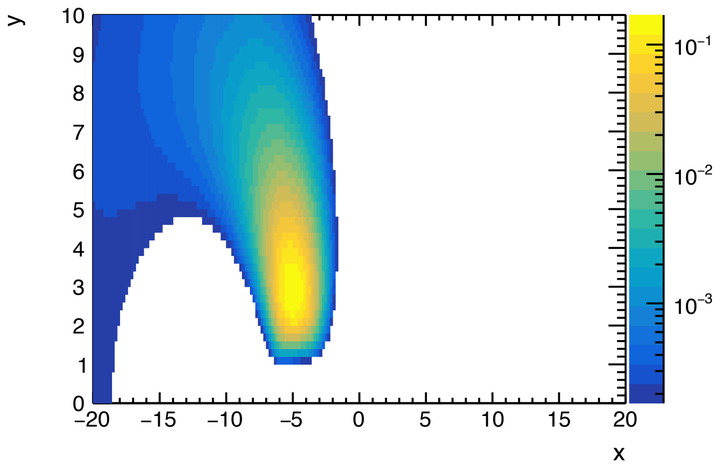
$$x_{old} = 1.5$$
$$p = -1.5$$
$$x_{new} = x_{old} + p = 0$$
$$\rho = \min\left(1, \frac{f(x_{new})}{f(x_{old})}\right) = \min(1, 1.45) = 1$$
$$u = 0.6$$
$$\rho > u \Rightarrow \textbf{accept}$$

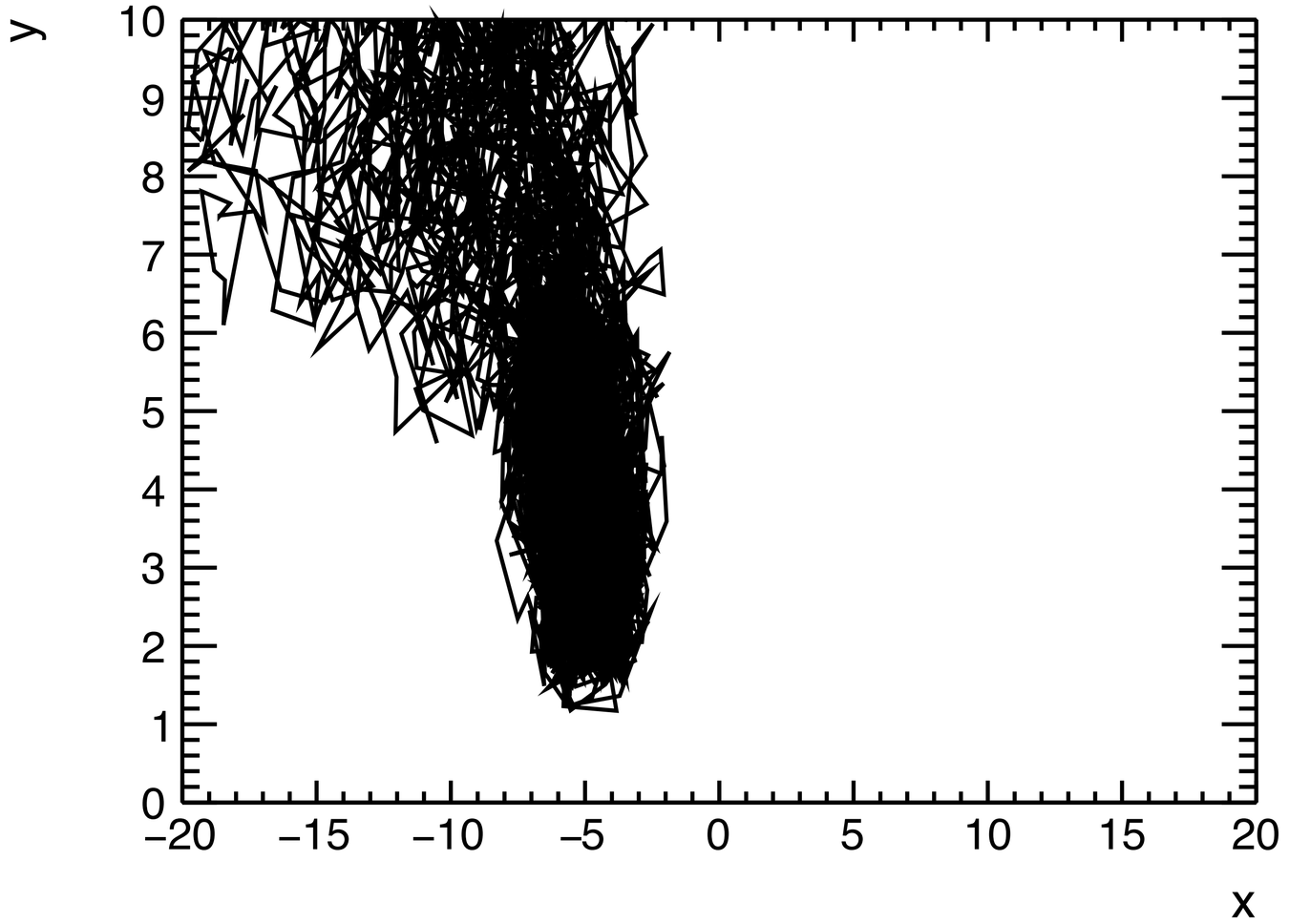
Figure 1.9: For $\rho > u$ the step is accepted again.

- Each step only depends on the previous point

Metropolis-Algorithm Example 2

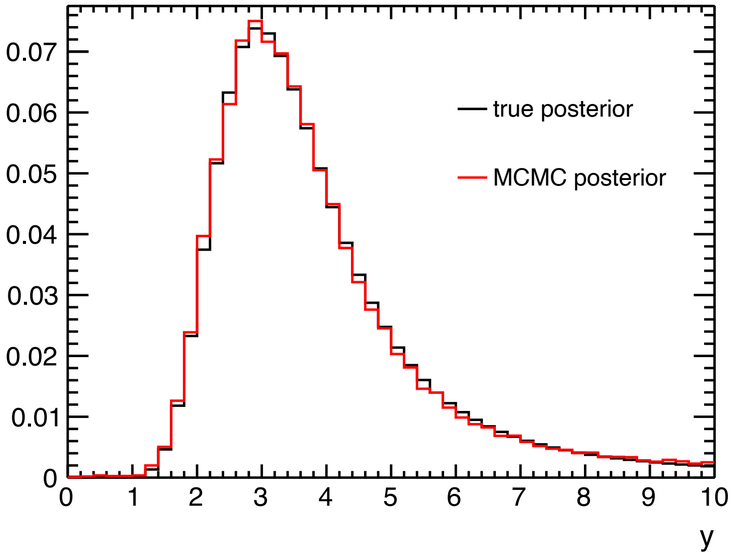
- Random-walk-like behaviour
- More time is spent in regions of high probability
- True distribution:

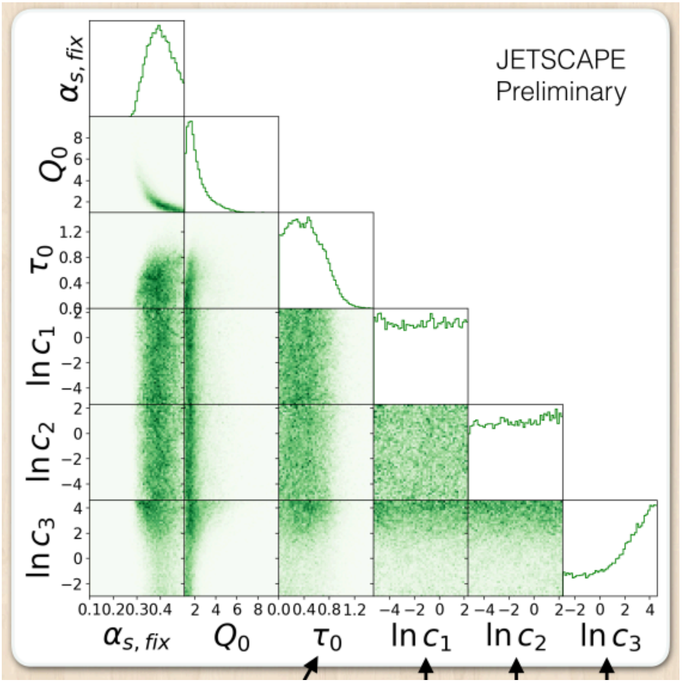




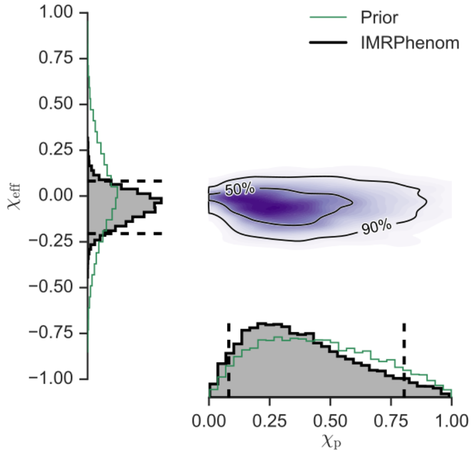
MCMC marginals

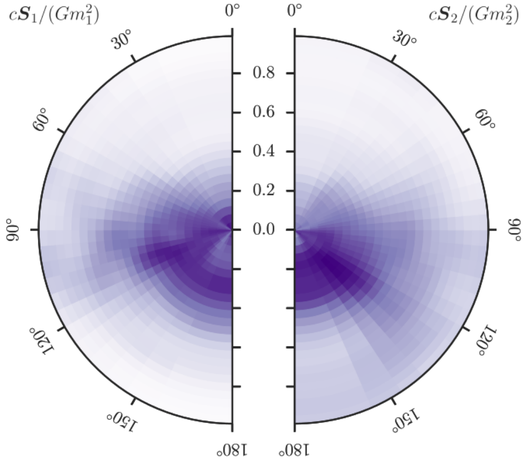
- With sampling, marginals become trivially easy: Just take the distribution of the coordinate
- Means and variances are estimated by the means and variances of the sampled points

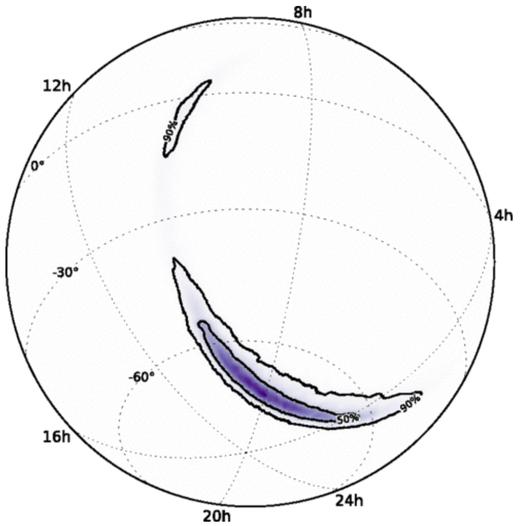




JETSCAPE Collaboration



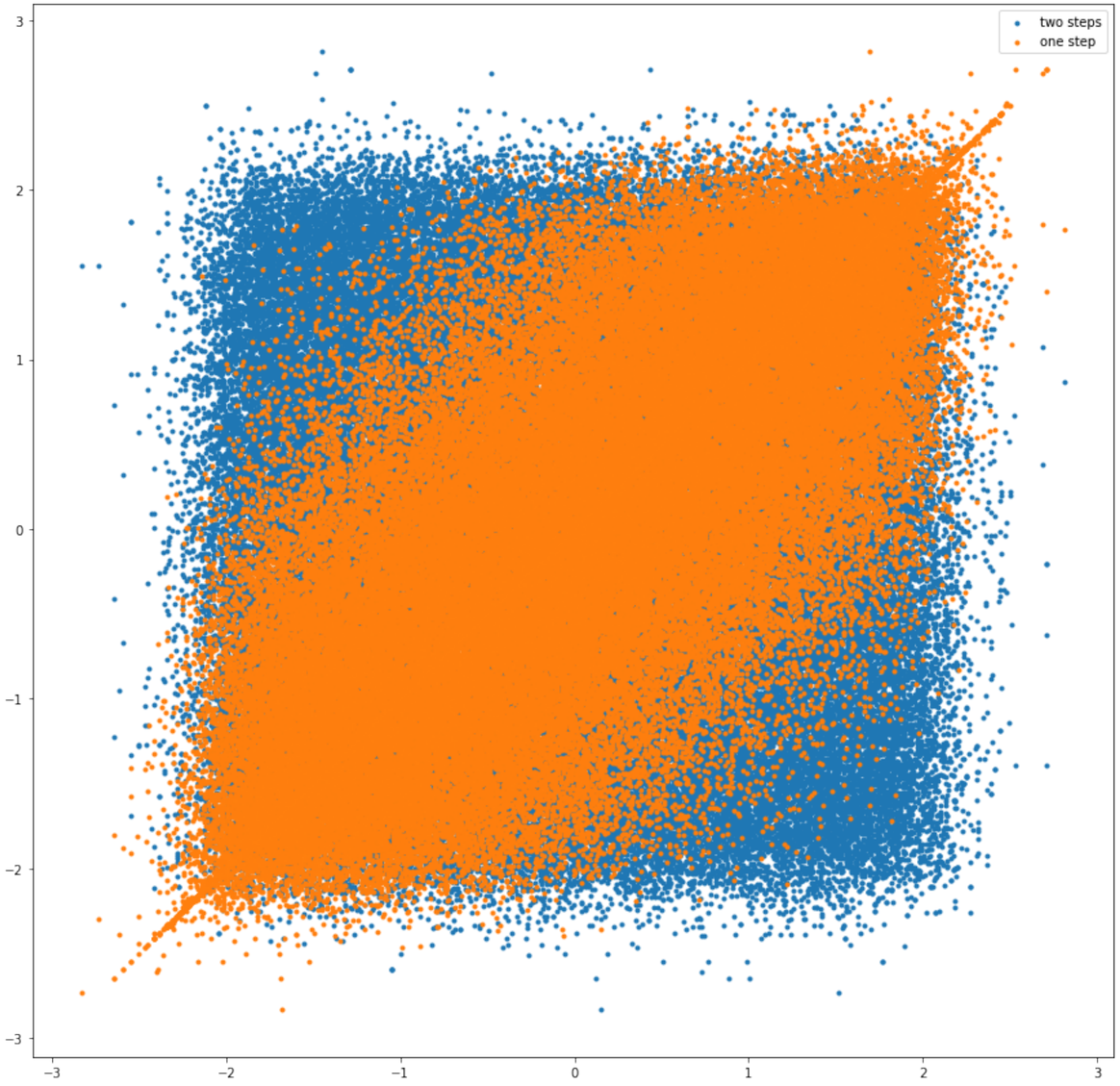




LIGO Collaboration

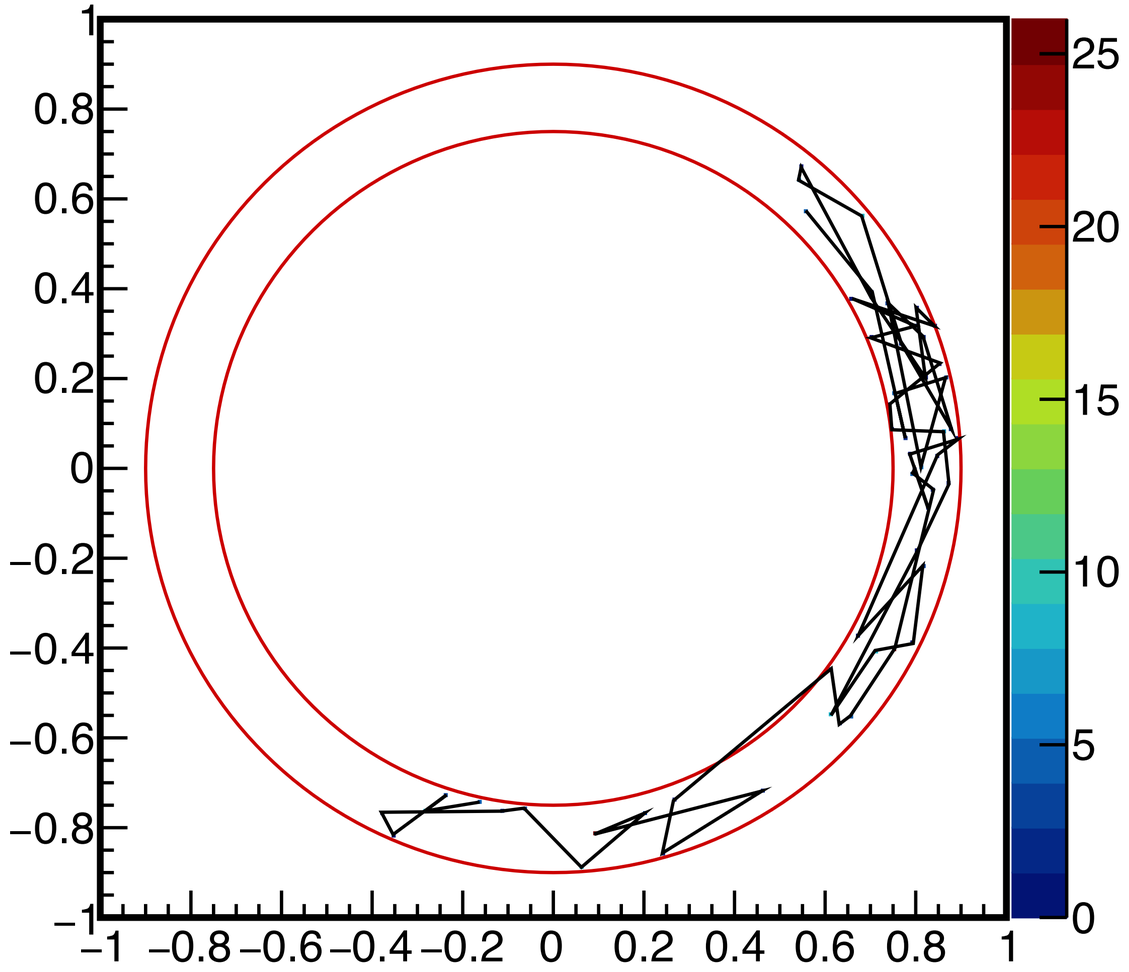
Properties of MCMCs

- Next position depends on current one - samples are not uncorrelated
  - But after a few steps they essentially are
- As long as all parts of the distribution are reachable by the steps, the result will always converge to the true distribution
  - But this can take a long time
- If the algorithm starts at a point of very low probability, then these initial points take a long time to become unimportant for e.g. averages
  - Ignore some of the initial steps - the “burn-in” - for faster convergence



A difficult example

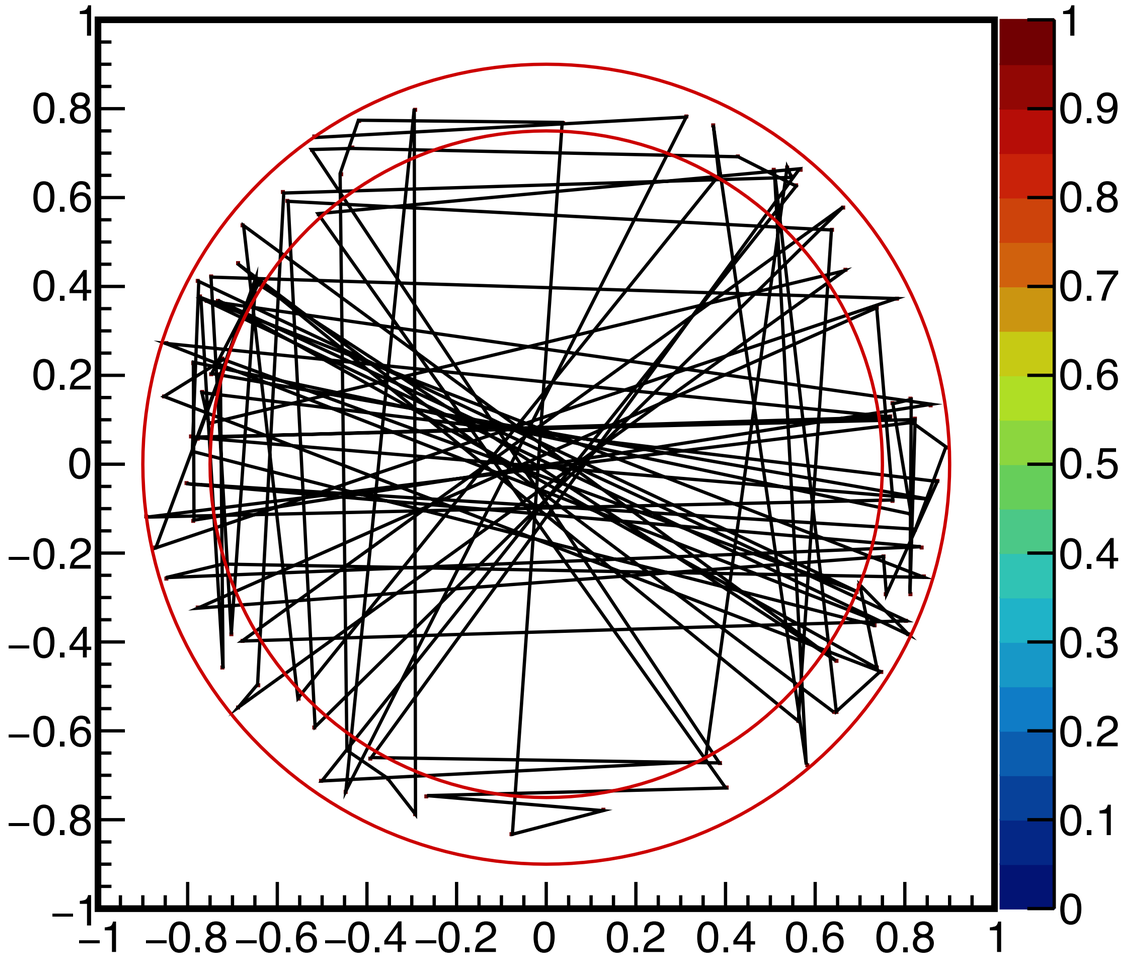
- Thin distributions can make it difficult for the walker to traverse the distribution
  - Small steps: many iterations to go around
  - Large steps: most steps lead out of the ring and are rejected
- Many types of MCMC algorithms are developed to deal with difficult distributions, high dimensions, improve convergence etc.



The distribution is constant on the ring and 0 elsewhere

The Gibbs-sampler

- Useful if we can sample easily from the conditional probability (keeping all coordinates except for one fixed)
  - e.g. $p(x_j \,|\, x_1, x_2, \ldots, x_{j-1}, x_{j+1}, \ldots x_n)$
- Start sampling from $p(x_1 \,|\, x_2, \ldots x_n)$ - new value x_1'
- Now sample from $p(x_2 \,|\, x_1', x_3, \ldots x_n)$
- Repeat for all variables, new position is $\vec{x}'$
  - No rejected steps
  - Can move through distributions quickly
- Special case of the Metropolis-Hastings algorithm



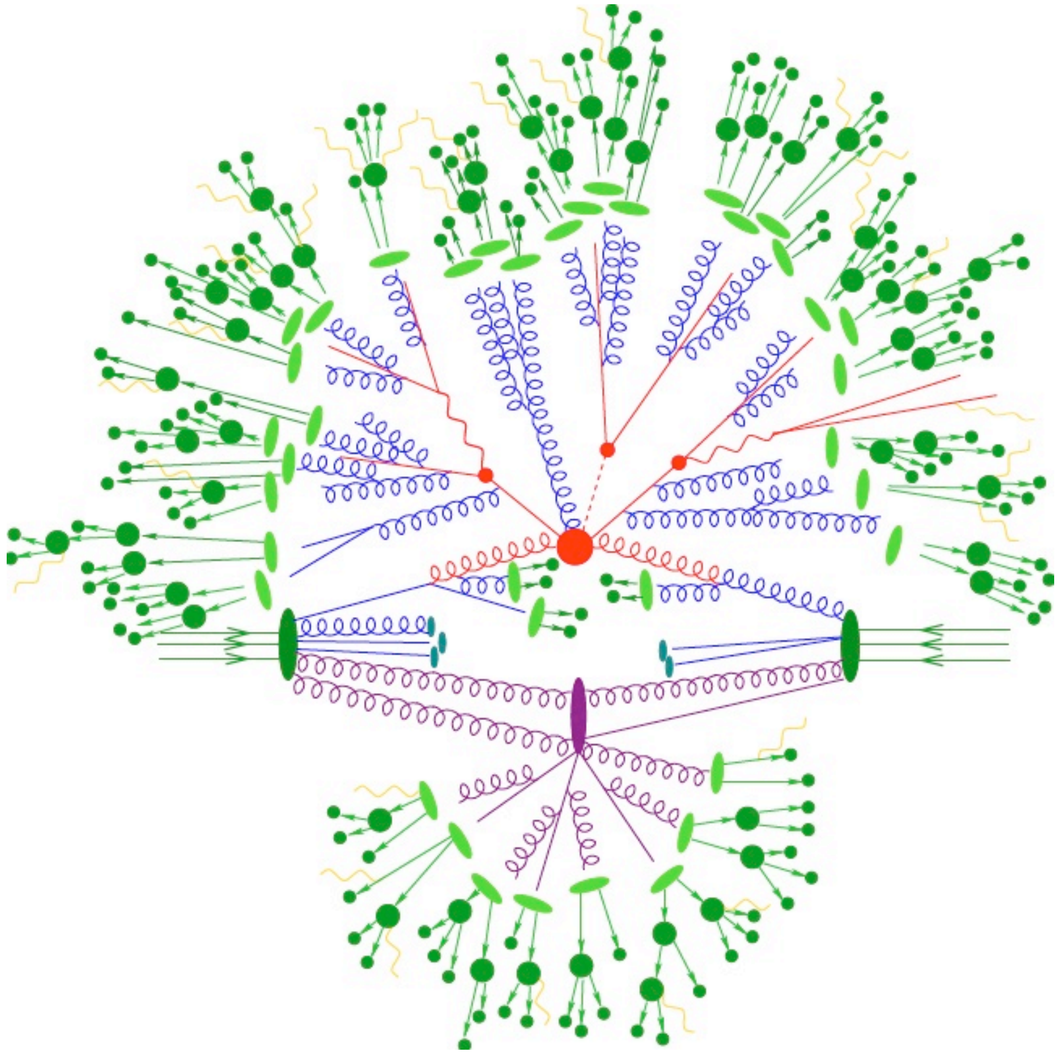
Event Generators

- Imagine a likelihood for getting the measured detector signals based on a parameter of interest and a bunch of others:
  - Consider all possible processes; all possible angles in multiple scattering, all fluctuations in detector signal
  - Then integrate out all other variables
  - Not viable
- Instead simplify:
  - Tracking algorithms, detector signal reconstruction, track end event selections/ triggers
  - Results in efficiencies - would still need to integrate over all processes and detector signals to find them
- Sampling from physics events and detector responses makes this easier (“Monte Carlo” generators/simulations)

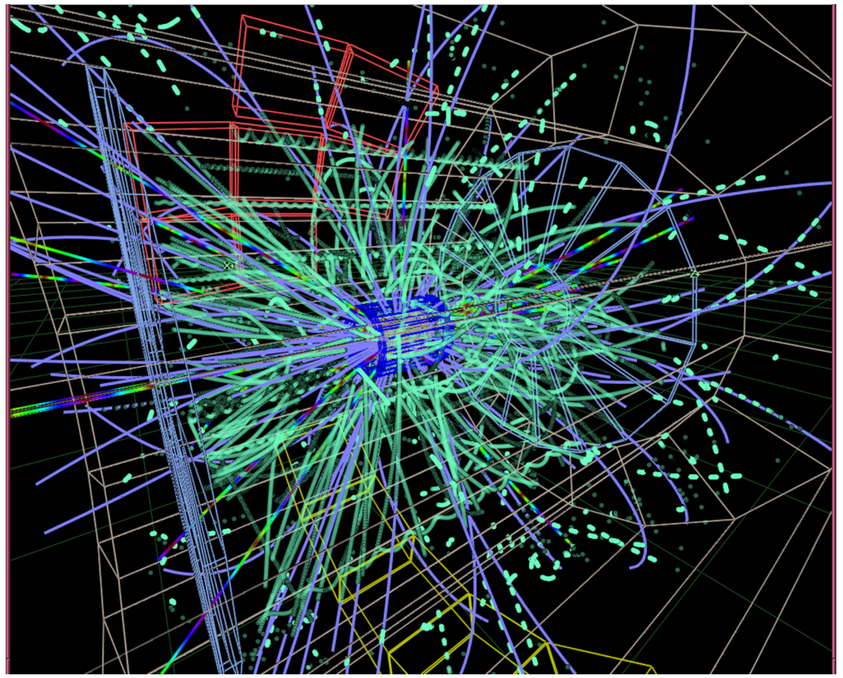
Monte Carlo simulation I: Event generators (Pythia, Sherpa, …)

Examples: Pythia

- Simulation of pp and e^+e^- collision on quark and gluon level
- Hard and soft interactions, parton showers, fragmentation and particle decay
- Many applications
  - Test underlying physics, e.g., perturbative QCD
  - Calculate QCD background processes, e.g., in Higgs searches
  - Calculation of detector efficiencies



Pythia



Output:

Four-vectors of of produced particles

Event listing (summary)

I	particle/jet	KS	KF	orig	p_x	p_y	p_z	E	m
1	(u)	A 12	2	0	0.000	0.000	10.000	10.000	0.006
2	(ubar)	V 11	-2	0	0.000	0.000	-10.000	10.000	0.006
3	(string)	11	92	1	0.000	0.000	0.000	20.000	20.000
4	(rho+)	11	213	3	0.098	-0.154	2.710	2.856	0.885
5	(rho-)	11	-213	3	-0.227	0.145	6.538	6.590	0.781
6	pi+	1	211	3	0.125	-0.266	0.097	0.339	0.140
7	(Sigma0)	11	3212	3	-0.254	0.034	-1.397	1.855	1.193
8	(K*+)	11	323	3	-0.124	0.709	-2.753	2.968	0.846
9	p~-	1	-2212	3	0.395	-0.614	-3.806	3.988	0.938
10	pi-	1	-211	3	-0.013	0.146	-1.389	1.403	0.140
11	pi+	1	211	4	0.109	-0.456	2.164	2.218	0.140

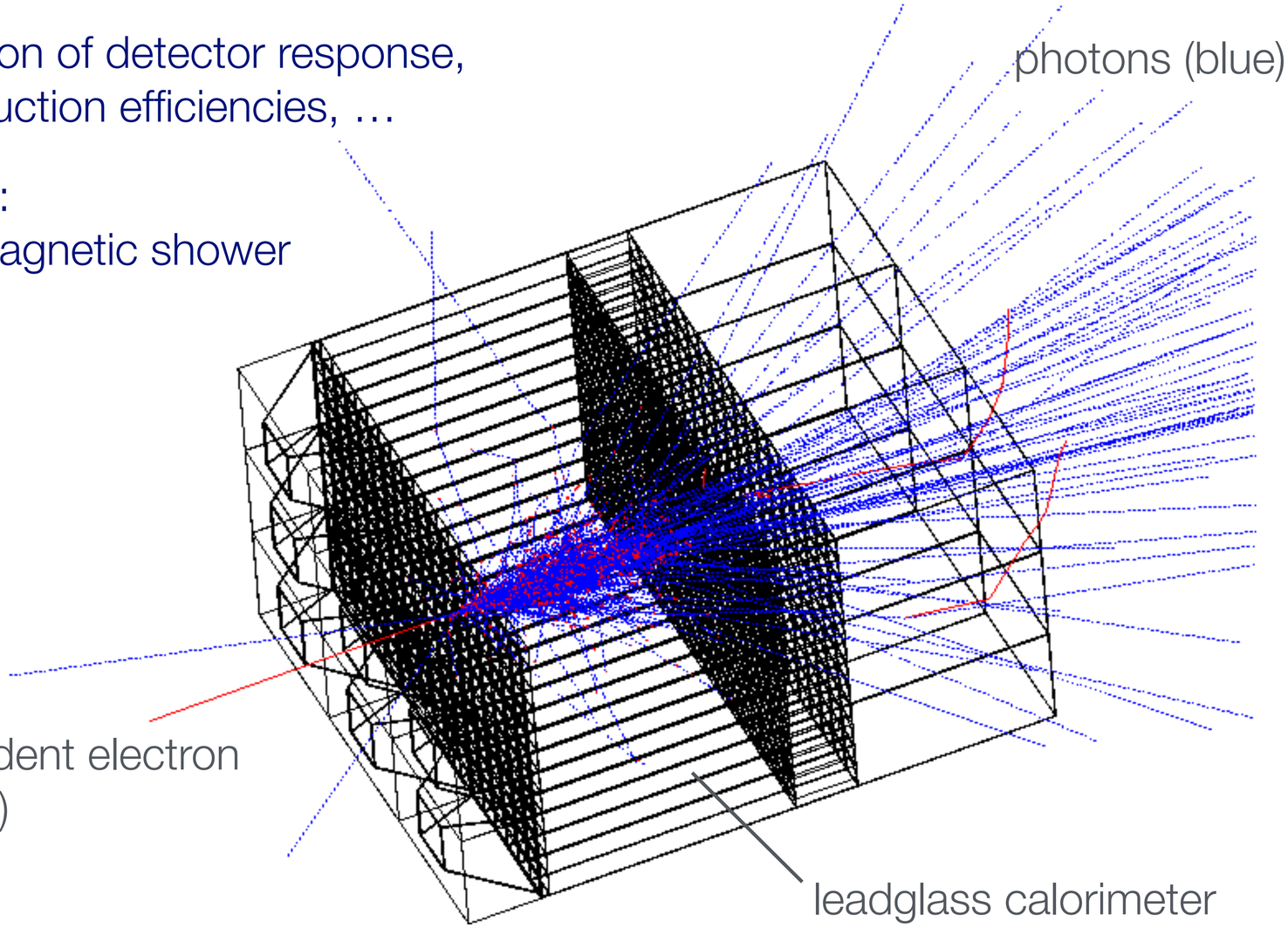
Monte Carlo simulation II: Detector simulation with GEANT

http://geant4.cern.ch/

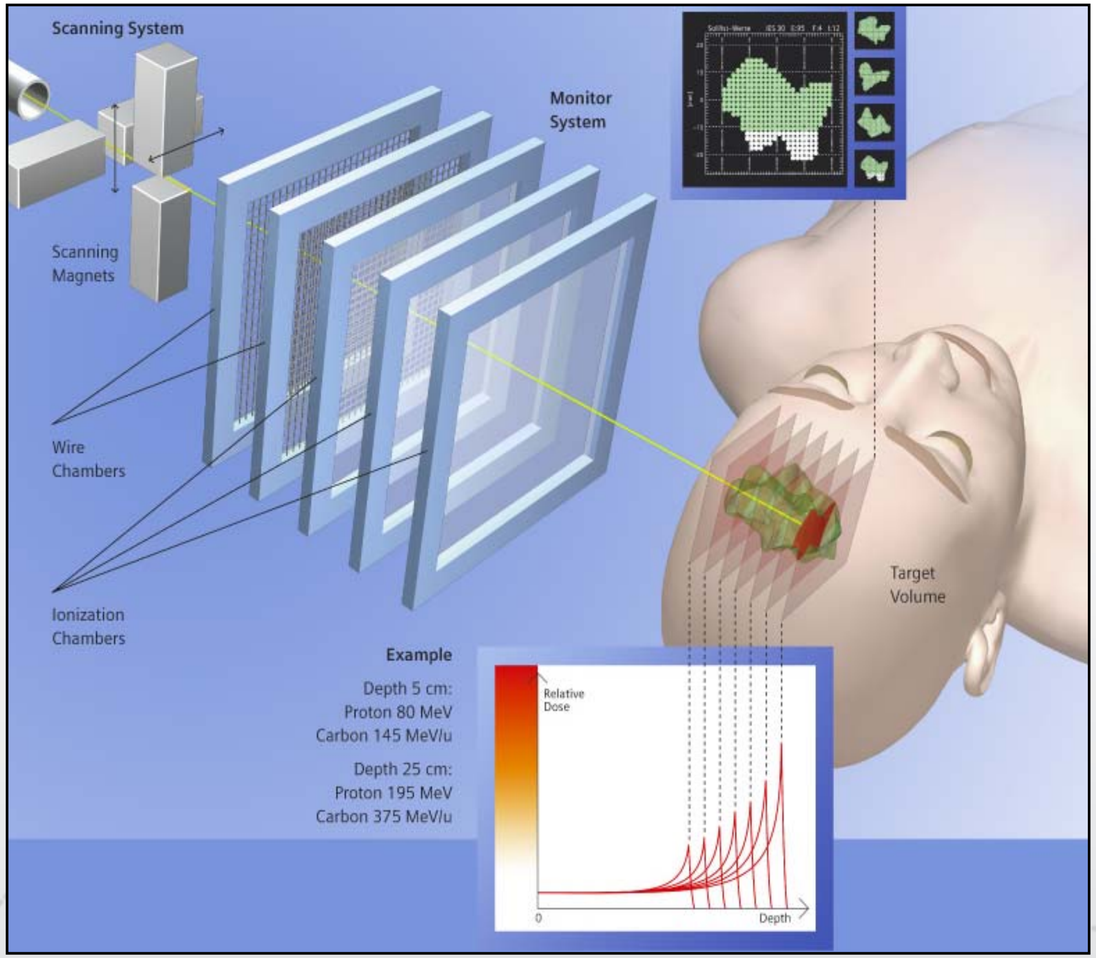
http://www.uni-muenster.de/Physik.KP/santo/thesis/diplom/kees

Calculation of detector response, reconstruction efficiencies, …

Example: electromagnetic shower

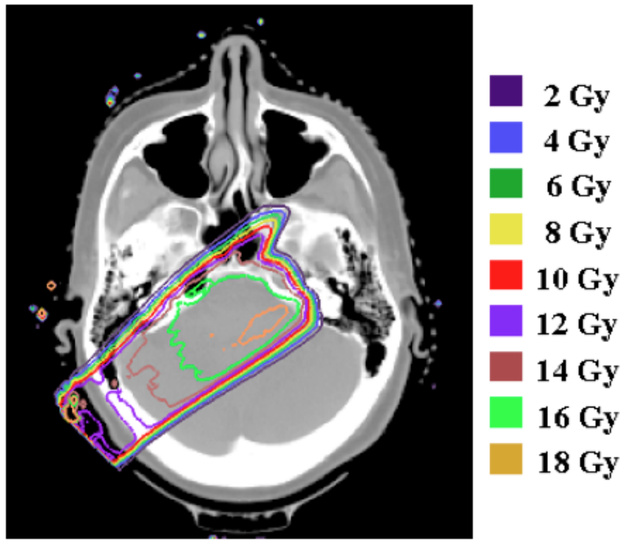


Monte Carlo simulation III: Treatment planning in radiation therapy



Intensity-Controlled Rasterscan Technique, Haberer et al., GSI, NIM A,1993

Source: GSI



Codes

- GEANT 4
- FLUKA
- …