Monte Carlo Methods in High Energy Physics III

Peter Uwer

HUMBOLDT-UNIVERSITÄT ZU BERLIN

PHYSICS AT THE TERA SCALE
Helmholtz Alliance

HEPTOOLS
Tools and Precision Calculations for Physics Discoveries at Colliders

Computational Theoretical Particle Physics
KA AC B
SFB TR9

Contents

Monte Carlo Integration

Consider $I = \int f(x) dx$

The integral can be interpreted as an expectation value of *f(x)* with uniform distributed random numbers:

$$I \approx I_N = \frac{1}{N} \sum_i f(x_i)$$

The central limit theorem tells us $\lim_{N\infty} I_N = I$

The central limit theorem also gives an handle on the uncertainty:

$$\frac{\Delta I}{I} = O(1/\sqrt{N})$$

Monte Carlo Integration

In principle we could think to do better:

$$\frac{\Delta I}{I} = \frac{\sqrt{Var(f(x_i))}}{\langle f(x) \rangle} \frac{1}{\sqrt{N}}$$

→ cannot make use of the exact formula in a simple way since:

$$\langle f(x) \rangle = \int f(x) dx, \quad Var(f(x)) = \int (f(x) - \langle f(x) \rangle)^2 dx$$

→ we would need the integrals over f(x) and f(x)^2 to improve the error estimate

Idea: Replace both quantities by the MC estimates.

Monte Carlo Integration in d dimensions

The extension to d dimensions is straight forward:

$$I = \int f(x_1, x_2, \ldots, x_d) d^d x$$

The Monte Carlo estimate is given by:

$$I \approx \frac{1}{N} \sum_i f(\vec{x}_i)$$

(we form d dimensional vectors, conceivable that the
Marsaglia effect indeed causes problems…)

Everything is integration…

Using Monte Carlo techniques we estimate the resulat from random numbers:

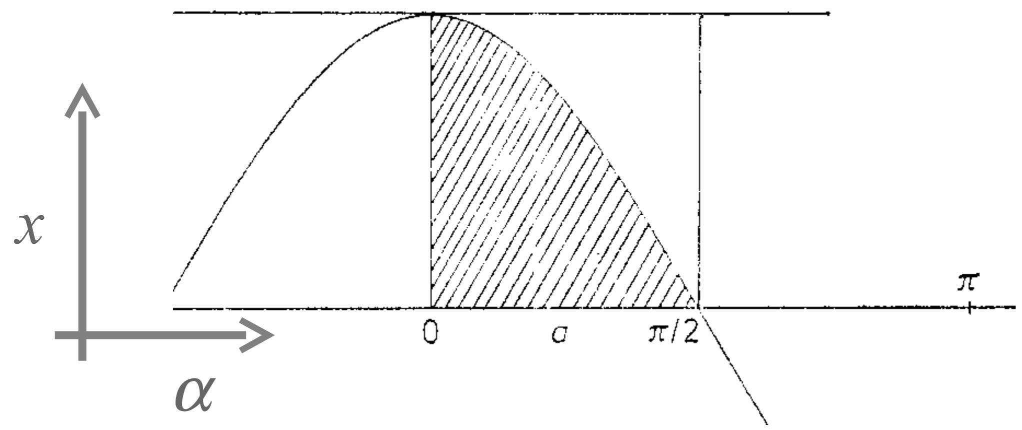
$$F = F(x_1, x_2, \ldots, x_n)$$

F can be interpreted as an estimate of the integral:

$$\int \int \ldots \int F(x_1, x_2, \ldots, x_n) dx_1 \ldots dx_n$$

Buffon’s needle

If we take the distance between the parallel lines and length of the needle to be 1, we find as condition:



The ratio of π is obtained from the ratio of the two areas which is in Buffon’s method estimated from the hit and miss MC

Hit and miss not very sensitive to the shape of the curve,
→ we need many points

A direct integration would perform much better

Monte Carlo Integration in d dimensions

How changes the error estimate ??

It doesn' change at all, we still get the $\frac{1}{\sqrt{N}}$

Keep in mind:

If we want to decreas the uncertainty by a factor of 10 we have to increase the N by a factor 100 !

If the initial calculation took 1 day it will take 100 days afterwards

But:

- We don't need to throw away what we calculated already
- Calculation can be parallelized

Comparison with numerical quadrature

Numerical quadrature in one dimension looks very similar:

$$I \approx \sum_i w_i f(x_i)$$

Variants of this type:

- Trapezoidal rule
- Simpson's rule
- Gauss rule

In one dimensions the error goes as:

$$\frac{1}{n^2}, \frac{1}{n^4}, \frac{1}{n^{2m-1}}$$

→ much better than Monte Carlo integration

Comparison with numerical quadrature

In higher dimensions numerical quadrature not so well developed

We can still use an iterated 1-dim. version

Uncertainty as a function of number of points n	In one dimension	In d dimensions
Monte Carlo	$n^{-1/2}$	$n^{-1/2}$
Trapezoidal rule	n^{-2}	$n^{-2/d}$
Simpson's rule	n^{-4}	$n^{-4/d}$
Gauss rule	n^{-2m+1}	$n^{-(2m-1)/d}$

[James]

→ convergence of MC integration slow

Numerical quadrature seems to be competitive for moderate dimensions

But:
- At a certain point brute force doesn´t work anymore
- Monte Carlo method can be improved

Can we improve Monte Carlo integration?

Error estimate:
$$\frac{\Delta I}{I} = \frac{\sqrt{Var(f(x_i))}}{\langle f(x) \rangle} \frac{1}{\sqrt{N}}$$

We can change the $\frac{1}{\sqrt{N}}$

But we can decrease the error by decreasing $Var(f)$

Suppose we want to calculate

$$\int_0^1 6z(1-z)dz$$

We learned in the previous lecture how to generate random numbers according to $6z(1-z)$

Simple example how to improve MC

We can interpret the integral in a different way:

We are calculating the expectation value of one with respect to the probability distribution function $6z(1-z)$

Our estimate then reads:

$$I \approx \frac{1}{N}\sum_i 1$$

There will be no fluctuation at all, one throw is sufficient to calculate the integral

Clear enough the result is not surprising:

When interpreting the 6z(1-z) as probability we knew already that the integral is 1

Simple example how to improve MC

Example shows:

Even if it is not possible to improve the $1/\sqrt{N}$ one still improve the convergence by decreasing the Variance using sophisticated techniques

Variance reduction: importance sampling

Importance sampling

Chose large number of points where the integrand is largest

$$I = \int f(x)dx = \int \frac{f(x)}{g(x)} g(x) dx$$

with $g(x)$ being probability distribution function

The expectation value is then obtained from

$$I_N = \frac{1}{N} \sum_i \frac{f(x_i)}{g(x_i)}$$

where the xi are distributed according to $g(x)$

Variance reduction: importance sampling

Note that we need to know the integral over $g(x)$

$g(x)$ should be chosen such that $\frac{f(x)}{g(x)}$ is nearly constant

This improves the variance since we have now

$$Var(I) \sim Var(f/g)$$

→ Overall convergence can be improved

Variance reduction: importance sampling

Comments:

- In general difficult to find appropriate g(x), in particular in higher dimension

  → need to know integral
  → need to be able to generate appropriate random numbers

- If g(x) goes to zero somewhere it can become instable

- Useful if something about the integral is know

  → transform variables to absorb part of the integrand

Variance reduction: Stratified sampling

Stratified sampling

Split integral into integrals over sub-region:

$$I = \int_0^1 f(x)dx = \int_0^a f(x)dx + \int_a^1 f(x)dx$$

In general: j sub-spaces with N_j points per sub-space j

For each sub-space a partial sum is performed

The partial sums are added weighted with $\frac{V_j}{N_j}$

Variance reduction: Stratified sampling

Integral estimate:

$$I = \sum_{k=1}^{j} \frac{V_k}{N_k} \sum_{i=1}^{N_k} f(x_{ki}$$

Using this technique the variance is given by

$$\sum_{k=1}^{j} \frac{V_j^2}{N_j} Var(f)|_{V_j}$$

Total variance can be reduced by a proper choice of (V_j, N_j)

“sample more points where the error gets large contributions”

Variance reduction: Control variates

Control variates:

Idea similar to importance sampling:

Make use of knowledge about the integrand,
to avoid instabilities we use the sum:

$$\int f(x)dx = \int (f(x) - g(x))dx + \int g(x)dx$$

If g approximates f very well the variance of f-g will be smaller than the variance of f !

Need to know the integral over g or have at least an efficient Monte Carlo method

Variance reduction: Antithetic variables

In lecture 1 we saw:

$$Var(f+g) = Var(f) + Var(g) + 2cor(f,g)$$

Idea:

use subsequent points which are negatively correlated

Example:

Suppose we want to integrate a monotonically increasing function of x from 0 to 1

Use as two subsequent points x_i and $1-x_i$

$$I_N = \frac{1}{2N}\sum_i f(x_i) + f(1-x_i)$$

Quasi-random variables

Taking antithetic variables to the extreme:

→ Random numbers are no longer “random”

In many MC integrations it is more important to sample the integration region as uniform as possible, than having truly random numbers

Will not be discussed in this lecture

For a integration making also use of quasi-random numbers see:

Cuba by Thomas Hahn

Variance reduction

Seen so far:

There are techniques to reduce the variance of the Monte Carlo integration

In all cases information about the integrand is needed

Integrand might be complicated function in many variables

→ in general not trivial to obtain this information

→ automatic procedure applying some variance reducing techniques highly desired

→ adaptive procedure which learn about the integrand during integration

Vegas by G.P.Lepage

JOURNAL OF COMPUTATIONAL PHYSICS **27**, 192–203 (1978)

A New Algorithm for Adaptive Multidimensional Integration

G. PETER LEPAGE

Stanford Linear Accelerator Center, Stanford University, Stanford, California 94305

Received November 10, 1976; revised June 15, 1977

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 ($n \gtrsim 4$).

→ traditional working horse for MC integration,

different implementations exist (Fortran,C), parallelized versions available, easy to use, still used by many people, less flexible than Cuba library

Vegas by G.P.Lepage: some details

Basic assumption:

Integrand can be approximated by factorized form

→ Each integration variable is divided into a certain numbers of subintervalls

Note that using a more general decomposition we would run out of memory:

Using 10 intervalls in each direction we get 10^d hypercubes

During the integration information on the contribution to the integral and to the error is collected

Vegas by G.P.Lepage: some details

Before starting the next iteration this information is used to apply importance sampling/stratified sampling

The user specifies:

- function to integrate
- number of calls per iteration
- number of iterations
- accuracy when the integration should be stopped

Typical usage:

```
vegas(f,itmx,ncall,acc)
```

default integration volume is $[0,1]^d$

Vegas

uwer on pepnote01: /home/uwer/projekte/WWj

```
uwer@pepnote01:WWj>./objs/x86_64/intel/test.exe
#
#   RANDOM GENERATOR: RANLUX
#
#   INPUT PARAMETERS FOR VEGAS   NDIM =  10  NCALL=        9216.
#                                IT =      0  ITMX =   10
#                                ACC = -.300E-21
#                                MDS=  1   ND=  50
#
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   1.   INTEGRAL = 0.53486263E-01
#                       STD DEV  = 0.3055E-03
#   ACCUMULATED RESULTS.   INTEGRAL = 0.53486263E-01
#                          STD DEV  = 0.3055E-03
#                          CHI**2 PER ITN   = 0.000
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   2.   INTEGRAL = 0.53828753E-01
#                       STD DEV  = 0.1377E-03
#   ACCUMULATED RESULTS.   INTEGRAL = 0.53771525E-01
#                          STD DEV  = 0.1253E-03
#                          CHI**2 PER ITN   = 1.040
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   3.   INTEGRAL = 0.54026075E-01
#                       STD DEV  = 0.4821E-04
#   ACCUMULATED RESULTS.   INTEGRAL = 0.53993645E-01
#                          STD DEV  = 0.4483E-04
#                          CHI**2 PER ITN   = 2.310
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   4.   INTEGRAL = 0.53964938E-01
#                       STD DEV  = 0.2091E-04
#   ACCUMULATED RESULTS.   INTEGRAL = 0.53970040E-01
#                          STD DEV  = 0.1891E-04
#                          CHI**2 PER ITN   = 1.650
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   5.   INTEGRAL = 0.53996902E-01
#                       STD DEV  = 0.1490E-04
#   ACCUMULATED RESULTS.   INTEGRAL = 0.53986652E-01
#                          STD DEV  = 0.1169E-04
#                          CHI**2 PER ITN   = 1.547
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   6.   INTEGRAL = 0.53962666E-01
#                       STD DEV  = 0.1430E-04
```

$$\prod_{i=1}^{10}\int_0^1 dx_i \exp(-x_i^2)$$

Vegas

uwer on pepnote01: /home/uwer/projekte/WWj

```
uwer@pepnote01:WWj>./objs/x86_64/intel/test.exe
#
#   RANDOM GENERATOR: RANLUX
#
#   INPUT PARAMETERS FOR VEGAS   NDIM =  10  NCALL=       9216.
#                                IT =     0  ITMX =    10
#                                ACC = -.300E-21
#                                MDS=  1   ND=  50
#
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   1.   INTEGRAL = 0.10788242E+09
#                       STD DEV  = 0.5100E+08
#   ACCUMULATED RESULTS.   INTEGRAL = 0.10788242E+09
#                          STD DEV  = 0.5100E+08
#                          CHI**2 PER ITN   = 0.000
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   2.   INTEGRAL = 0.17464528E+11
#                       STD DEV  = 0.1678E+11
#   ACCUMULATED RESULTS.   INTEGRAL = 0.34916527E+10
#                          STD DEV  = 0.5238E+10
#                          CHI**2 PER ITN   = 1.723
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   3.   INTEGRAL = 0.14042757E+09
#                       STD DEV  = 0.7164E+08
#   ACCUMULATED RESULTS.   INTEGRAL = 0.21219552E+10
#                          STD DEV  = 0.3289E+10
#                          CHI**2 PER ITN   = 1.418
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   4.   INTEGRAL = 0.59333667E+09
#                       STD DEV  = 0.2057E+09
#   ACCUMULATED RESULTS.   INTEGRAL = 0.14043260E+10
#                          STD DEV  = 0.2076E+10
#                          CHI**2 PER ITN   = 1.304
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   5.   INTEGRAL = 0.88301688E+10
#                       STD DEV  = 0.2869E+10
#   ACCUMULATED RESULTS.   INTEGRAL = 0.39917629E+10
#                          STD DEV  = 0.1679E+10
#                          CHI**2 PER ITN   = 2.084
#
#   INTEGRATION BY VEGAS
#   ITERATION NO   6.   INTEGRAL = 0.16553505E+11
#                       STD DEV  = 0.2967E+10
#   ACCUMULATED RESULTS.   INTEGRAL = 0.10695305E+11
```

$$\prod_{i=1}^{10} \int_0^1 dx_i \frac{1}{x_i}$$

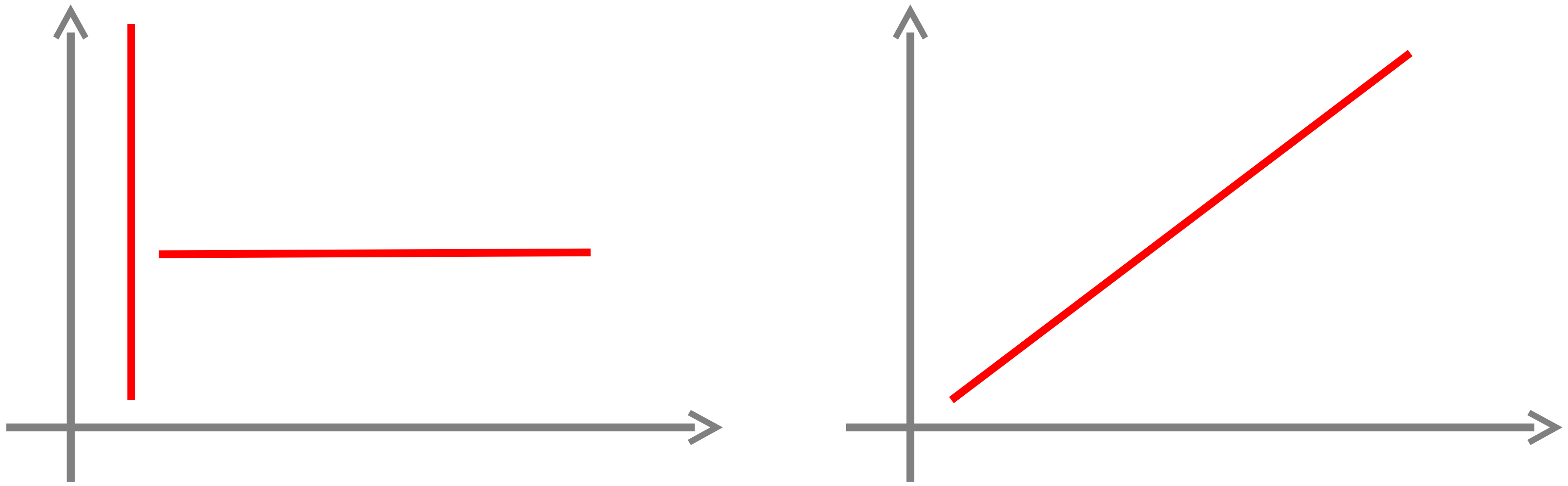
Vegas by G.P.Lepage: some details

Comments: (apply to some extend also to other MC integrators)

- results of individual iterations should be to some extend consistent → increase the number of calls
- The accumulated information about the integrand can be stored and used later to add iterations
- Adaptive Monte Carlo is a trouble finder: if there is problem in the integration region the integrator will find it → large fluctuation, inconsistent results
- Possible to calculate arbitrary distributions together with the integrand

Understanding Vegas

Due to factorization ansatz Vegas cannot tune to all types of problems:



need to use the proper variables to get good performance

Comparison of MC results

What is meant by *convergence of MC results* ?

→ Convergence in a probabilistic sence

Depending on the size of the uncertainy we attribute (1sigma, 2 sigma, …) the results should agree with a certain probability within their errors

(if we assume a Gaussian we can calculate the probaility)

Important:

There is always a probability that they do not agree despite the fact that everything was done correct!

If we have large N…

$$\text{Prob}\left(-a\frac{\sqrt{Var(f)}}{\sqrt{N}} \le \frac{1}{N}\sum_{i=1}^{N} f(x_n) - I \le -b\frac{\sqrt{Var(f)}}{\sqrt{N}}\right) = \frac{1}{2\pi}\int_{-a}^{b} dt \exp\frac{-t^2}{2}$$

The variance Var(f) is estimated from MC:

$$Var(f) = \frac{1}{N-1}\sum_{n=1}^{N}(f(x_n) - \langle f \rangle)^2 \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n)^2 - \langle f \rangle^2$$

Important consequences

If you compare many numbers there are always some which do not agree within 1 StD, 2 StD and even 3 StD.

If it is different something is wrong with the Monte Carlo

If the results are correlated because you are using the same random numbers or something similar that would explain a better agreement than expeced from statistics

Note:

When increasing the statistics, the picture will not change, although the relative uncertainty goes down!

→ to compare just count how many numbers are off by 1StD, 2Std,

The End