

**MONTE CARLO METHODS
FOR HIGH ENERGY PHYSICS**

by S. Jadach

Institute of Nucl. Physics, Kraków, Poland

Lecture 3

Addaptive Monte Carlo algorithms

Slides, program sources: <http://home.cern.ch/jadach>

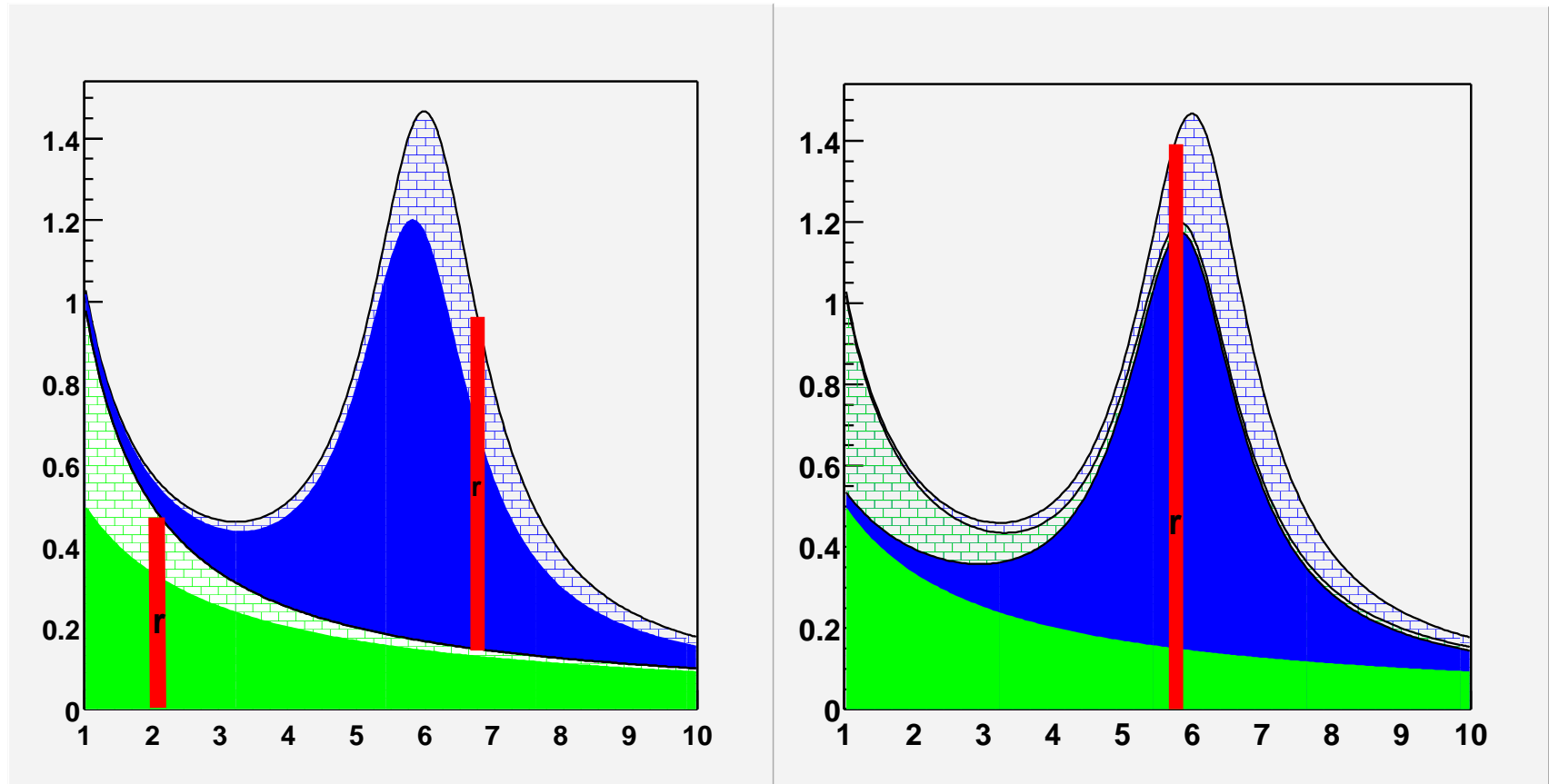
Numerical examples exploit ROOT package: <http://root.cern.ch/>

Outline:

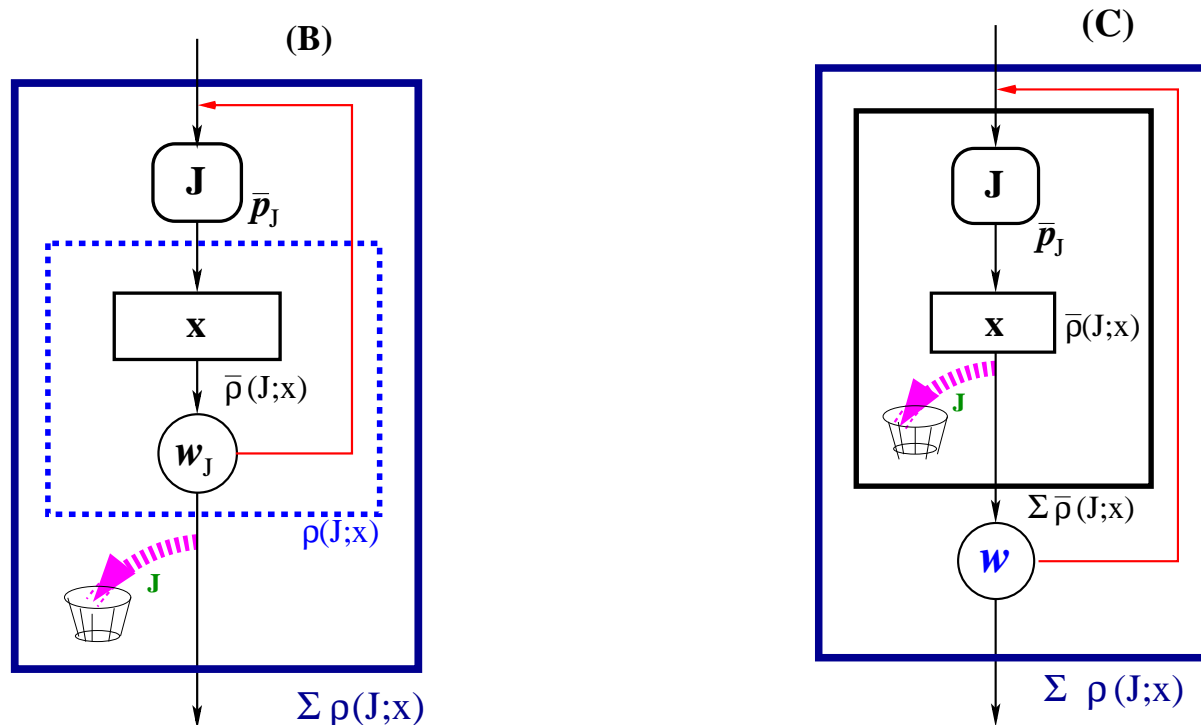
1. Introduction
2. Optimizing branching probabilities for wt-ed events
3. Vegas
4. Foam

Additional comments on combining branching and rejection

We may encounter “Equivalent algorithms” which provide the same distributions and integrals, use the same elementary methods and differ only in the efficiency. They may have the same average weight but different weight distribution. In the next slide there is an example of such 2 algorithms.



Define: $\langle\langle U \rangle\rangle \equiv \sum_J \bar{p}_J \int \frac{\bar{\rho}(J;x)}{R_J} U(J;x) dx^n \equiv \frac{1}{R} \sum_J \int \bar{\rho}(J;x) U(J;x) dx^n$



$$w_B(J;x) = \frac{\rho(J;x)}{\bar{\rho}(J;x)}$$

$$\langle\langle w_B \rangle\rangle = \int \sum_j \rho(J;x) = R$$

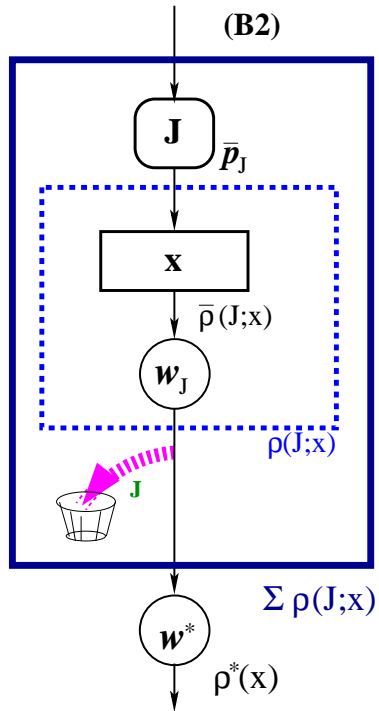
$$\langle\langle w_B^2 \rangle\rangle = \frac{1}{R} \int \sum_J \frac{\rho^2(J;x)}{\bar{\rho}(J;x)} dx^n$$

$$w_C(x) = \frac{\sum_J \rho(J;x)}{\sum_J \bar{\rho}(J;x)}$$

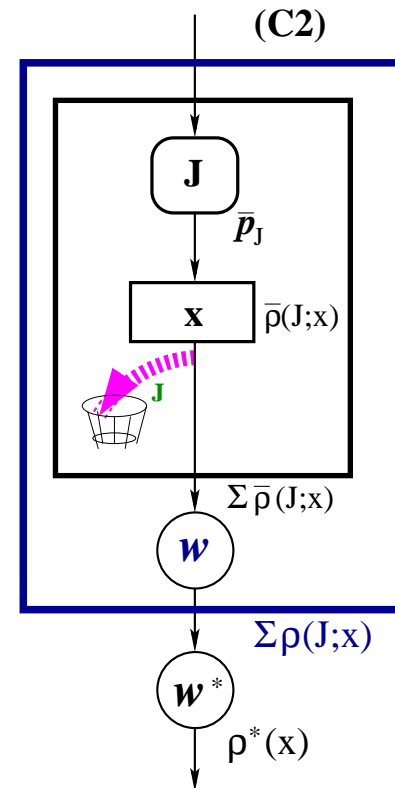
$$\langle\langle w_C \rangle\rangle = \int \sum_j \rho(J;x) = R$$

$$\langle\langle w_C^2 \rangle\rangle = \frac{1}{R} \int \frac{(\sum_J \rho(J;x))^2}{\sum_J \bar{\rho}(J;x)} dx^n$$

The above “equivalence” looks more interesting when we add additional branch-independent weight w^* and open rejection loops:



$$w = \frac{\rho(J;x)}{\bar{\rho}(J;x)} \frac{\rho^*(x)}{\sum_J \rho(J;x)}$$



$$\begin{aligned} w &= \frac{\sum_J \rho(J;x)}{\sum_J \bar{\rho}(J;x)} \frac{\rho^*(x)}{\sum_J \rho(J;x)} \\ &= \frac{\rho^*(x)}{\sum_J \bar{\rho}(J;x)} \end{aligned}$$

Branch optimization in case of wt-ed events (Kleiss&Pittau)

Consider (C) with “opened rejection loop”. Target is $\rho(x)$.

Generated primarily: $\bar{\rho}(x) = \sum \bar{\rho}(J; x)$ (mapping).

Normalization “anchor”:

$$\bar{R} = \sum_J \bar{R}_J = \sum_J \int \bar{\rho}(J; x) dx^n. \text{ Integral is}$$

$$R = \bar{R} \langle \langle w \rangle \rangle, \text{ where the average is over branches and}$$

over integration domain. The MC error is

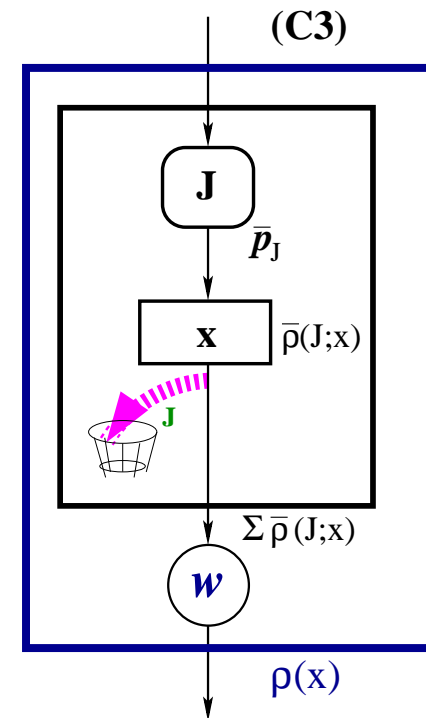
$$\delta R = \frac{\sigma}{\sqrt{N}} = \frac{\bar{G}}{\sqrt{N}} \sqrt{\langle \langle w^2 \rangle \rangle - \langle \langle w \rangle \rangle^2}$$

Define “MC inefficiency” as $iE = N \frac{\delta R^2}{R^2} + 1 = \frac{\langle w^2 \rangle}{\langle w \rangle^2}$

$$\text{We find: } iE = \frac{1}{\int \rho(x) dx^n} \bar{R} \int \frac{(\rho(x))^2}{\sum_J \bar{\rho}(J; x)} dx^n$$

Our aim is to get **the smallest iE** by manipulating relative overall normalizations of the branch distributions

$$\bar{\rho}(J; x) \rightarrow \lambda_J \bar{\rho}(J; x).$$



$$w = \frac{\rho(x)}{\sum_J \bar{\rho}(J; x)}$$

Branch optimization in case of wt-ed events, cont

Our aim is to get **the smallest iE** by manipulating relative normalizations of the branch distributions $\bar{\rho}(J; x) \rightarrow \lambda_J \bar{\rho}(J; x)$. (λ_J are independent of x !)

We look for a minimum of $E = A \int \frac{(\rho(x))^2}{\sum_J \bar{\rho}(J; x)} dx^n \bar{R}$ as a function of λ_J .

Obviously $\bar{\rho}(J; x) \rightarrow \lambda \bar{\rho}(J; x)$ changes nothing.

Let us put constraint $\bar{R} = \sum_J \lambda_J \bar{R}_J = \text{const}$, to fix normalization.

By means of Lagrange method, differentiation $\partial/\partial\lambda_J$ provides us local minimum condition:

$$W_K = \int \frac{(\rho(x))^2}{(\sum_J \bar{\rho}(J; x))^2} \frac{\bar{\rho}(K; x)}{\bar{R}_K} dx^n = B, \quad \text{all } W_K \text{ the same, independent of } K!$$

Kleiss&Pittau proposed iterative solution in which one starts from, say, $\lambda_J = 1$, then

$W_K = \langle \langle w^2 \frac{\bar{\rho}(K; x)}{\bar{R}_K} \frac{\bar{R}}{\bar{\rho}(x)} \rangle \rangle$ is calculated from the trial MC run and for the next trial run we

$$\text{substitute } \lambda_J \rightarrow \lambda_J \sqrt{W_J \bar{R}_J}.$$

This ansatz is based on the observation that for $\rho(J; x)$ distributions which are vanishingly overlapping, we can solve the “minimum condition equations”, almost exactly, just after first iteration! (modulo statistical errors). See next slide....

Branch optimization in case of wt-ed events, cont.

How to understand the origin of K&P ansatz?

Imagine that the target distribution $\rho(x)$ has just two distinct peaks with almost zero overlap, located in two subdomains (1) and (2).

We have at our disposal two “primitive” distributions $\bar{\rho}(1; x)$ and $\bar{\rho}(2; x)$ which fit very well $\rho(x)$ up to a normalization. Furthermore, $\bar{\rho}(1; x)$ is nonzero in domain (1) and practically zero in (2), while $\bar{\rho}(2; x)$ is nonzero in domain (2) and practically zero in (1).

Even more, let us assume that someone told us correct optimal normalization of $\bar{\rho}(I; x)$, so we know that the two integrals:

$$W_K = \int \frac{(\rho(x))^2}{(\sum_J \bar{\rho}(J; x))^2} \frac{\bar{\rho}(K; x)}{\bar{R}_K} dx^n = B, \quad K = 1, 2 \text{ are equal.}$$

Nevertheless we ignore this information and we run trial MC using “wrong” $\lambda_J \bar{\rho}(J; x)$ and obtain “wrong” W'_1 and W'_2 .

Since two integrals very well “separated” in the space, we get:

$$W'_1 \simeq \frac{1}{\lambda_1^2 \bar{R}_1} \int \frac{\rho(x)}{\bar{\rho}(1; x)} dx^2 = \frac{C}{\lambda_1^2 \bar{R}_1} \text{ and } W'_2 \simeq \frac{1}{\lambda_2^2 \bar{R}_2} \int \frac{\rho(x)}{\bar{\rho}(2; x)} dx^2 = \frac{C}{\lambda_2^2 \bar{R}_2}.$$

Obviously, to get back to original correct normalization,

we should to correct our “wrong” $\bar{\rho}(I; x)$ using factor $\lambda_I^{-1} = (W'_I \bar{R}_I)^{1/2}$ $I = 1, 2$.

(Modulo constraint $\bar{R} = \bar{R}_1 + \bar{R}_2 = \text{const}$, of course.)

Custom versus General-Purpose MC's

Custom-made MC's is for a narrow class of distributions.

For a well defined family of $\rho(\vec{x})$. For single physical problem.

Made by combining cleverly a handful of elementary methods:

- (a) Mapping of variables,
- (b) Rejection according to a certain weight,
- (c) Multi-branching (multi-channeling).

σ -Reduction and W_{\max} -Reduction done “by hand”.

All (?) MC Simulators, most of MC Generators are in this class.

General-Purpose MC's with built-in ability of adjusting to ρ .

For a wide class of ρ 's. Practically only one type exists:

- (1) Divide integr. domain Ω into many small cells $\omega_i, i = 1 \dots l$
- (2) Generation: choose i and generate $\vec{x} \in \omega_i$ uniformly.
- (3) σ & W_{\max} -Reduction automatic, with clever choice of ω_i .

VEGAS family: cells are cubical.

Factorisability $\rho(\vec{x}) \simeq \prod_{j=1}^n f_j(x_j)$ is required.

Foam: cells simplicial/hyp-cubical. Factorisability not required.

Hybrid: Custom-made + General-Purp. for some variables

VEGAS and the like

- G.P. Lepage, J. Comput. Phys. **27**, 195 (1978).

“Recent” activity in the subject:

- G. I. Manankova, A. F. Tatarchenko, and F. V. Tkachev, MILXy way: How much better than VEGAS can one integrate in many dimensions?, 1995, a Contribution to AINHEP-95, Pisa, Italy, Apr 3-8, 1995 (extended version).
- T. Ohl, Vegas revisited: Adaptive Monte Carlo integration beyond factorization, Comput.Phys.Commun. **120** (1999)13, eprint: hep-ph/9806432.
- S. Kawabata, Comp. Phys. Commun. **88**, 309 (1995).
- R. Kleiss and R. Pittau, Comput. Phys. Commun. **83**, 141 (1994).

Cellular:

- S. Jadach, Comput.Phys.Commun. **130**, 244 (2000) ;
e-Print: physics/9910004.
- Earlier unpublished trials by S. Kawabata, S. Jadach, T. Ohl, R. Kleiss...??

Simulation \neq Generation \neq MC-Integration

Distribution $\rho(x_1, x_2, \dots, x_n)$, integral $R = \int_{\Omega} d^n x \rho(\vec{x})$.

MC Integration (MC Integrator):

Provides **integral I** , but **NO MC events!**

$\rho(\vec{x}) < 0$ allowed! Efficiency = $\frac{\langle W \rangle}{\sqrt{\langle W^2 \rangle}}$; $\Delta R = \frac{\sigma(W)}{\sqrt{N}}$. Special random numbers allowed.

MC Generation (MC Generator):

Provides **W -ted MC events $\{\vec{x}, W\}$** and R ,

$\rho(\vec{x}) < 0$ allowed! Efficiency = $\frac{\langle W \rangle}{\sqrt{\langle W^2 \rangle}} = \frac{\langle W \rangle}{\sqrt{\langle W \rangle^2 + \sigma(W)^2}}$. Random numbers true or special.

MC Simulation (MC Simulator):

Provides **$W=1$ MC events $\{\vec{x}\}$** and R , ($\rho(\vec{x}) \geq 0$!)

Efficiency = $\frac{\langle W \rangle}{W_{\max}}$, $W =$ (internal) rejection weight, Required true (pseudo)random numbers.

MC Simulator is the most difficult to construct.

MC generator \Rightarrow MC simulator unfeasible, if crazy W_{\max} (the usual case).

**Original VEGAS = MC-Integrator, PYTHIA = MC-Simulator, MC-Generators:
customized VEGAS, FOWL (1968)**

How to improve the MC efficiency?

The answer depends on what the efficiency is!

If MC events $(\vec{x}_j, j = 1, N)$ generated **primarily** with $\bar{\rho}(\vec{x})$ then MC weight is $W_j = \frac{\rho(\vec{x}_j)}{\bar{\rho}(\vec{x}_j)}$ and integral $R = \langle W \rangle \bar{R}$.

MC Integration (MC Integrator):

Variance-Reduction: $\frac{\sigma(W)}{\langle W \rangle} \rightarrow 0$.

MC Generation (MC Generator):

Variance-Reduction: once again.

MC Simulation (MC Simulator):

W_{\max} -Reduction: Try hard $\frac{\langle W \rangle}{W_{\max}} \rightarrow 1$!!!

In every case we try $\bar{\rho} \rightarrow |\rho|$.

However, a given method for $\bar{\rho} \rightarrow |\rho|$ good for Variance-Reduction is not necessary good for W_{\max} -Reduction, and vice-versa.

Definition of W_{\max}

Naive def. $W_{\max} = \text{Max}_j w(\vec{x}_j)$ **not good!**

Problems in case of W with “weak tail” due to:

- (1) seldom numerical instabilities or**
- (2) “weak singularity” like $x^{-0.1}$ or $\ln 1/x$ in ρ .**

The remedy is to define W_{\max}^ε as follows:

For a given precision level $\varepsilon \ll 1$, the W_{\max}^ε is determined from the weight distribution in such a way, that the relative contribution to the $\langle W \rangle$ (that is to the total integral) due to truncation $W \rightarrow W_{\max}$ for all $W > W_{\max}$ events is equal ε .

It is a little bit of effort to determine in practice the W_{\max}^ε from the weight distribution. Needed histogram of weight distribution with at least ~ 1000 bins, in order to determine it with say 2-digit precision.

Peculiarities of VEGAS

- No published official version of the source code. You may get one from Peter Lepage. However, it is not exactly the same which produced tables in the article.
- Better check that you have got the right code. (Not spoiled by someone).
- In order to understand the algorithm one has to read carefully article and “decompile the code”. (Program is in FORTRAN IV, modern in 1965 but almost assembler by today’s standard).
- Hidden/unadvertized feature: oscillations of the binning.
- Most of users treat it as a black box, of course.

VEGAS algorithm

The integrand function in n dimensions is assumed to be fairly well approximated by a product of functions, $\rho(x) = \prod_{i=1}^n \rho_i(x_i)$ each one depending just on one integration variable.

The integration range of each variable is divided into k bins of unequal width, with the binning (bin sizes) different for each variable.

The entire integration domain, that is an n -dimensional rectangle, is divided into k^n sub-rectangles.

The whole structure is explored by means of the MC generation of random points within each sub-rectangle, with a uniform distribution. Exploration repeated iteratively.

The binning is adjusted iteratively, such that the the ratio of the dispersion to the average weight is minimized – variance reduction.

The “driving function” which controls binning in each direction is the density $\frac{\rho(x)^2}{p(x)}$. The grid evolves iteratively such that projection of this function on each direction is as flat as possible. In each iteration bins are first subdivided, integral and projections onto each axis of $\frac{\rho(x)^2}{p(x)}$ are calculated and the new binning is established.

VEGAS customization

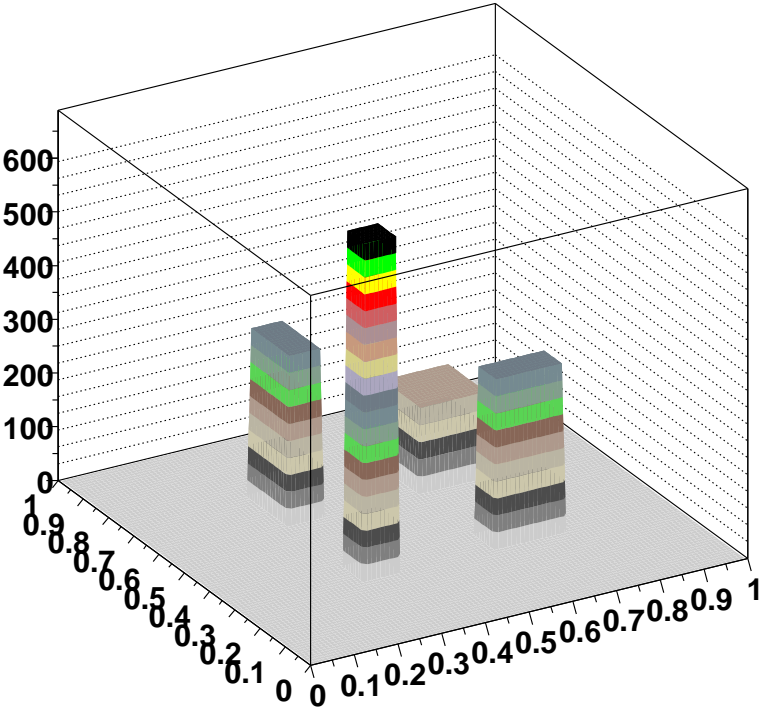
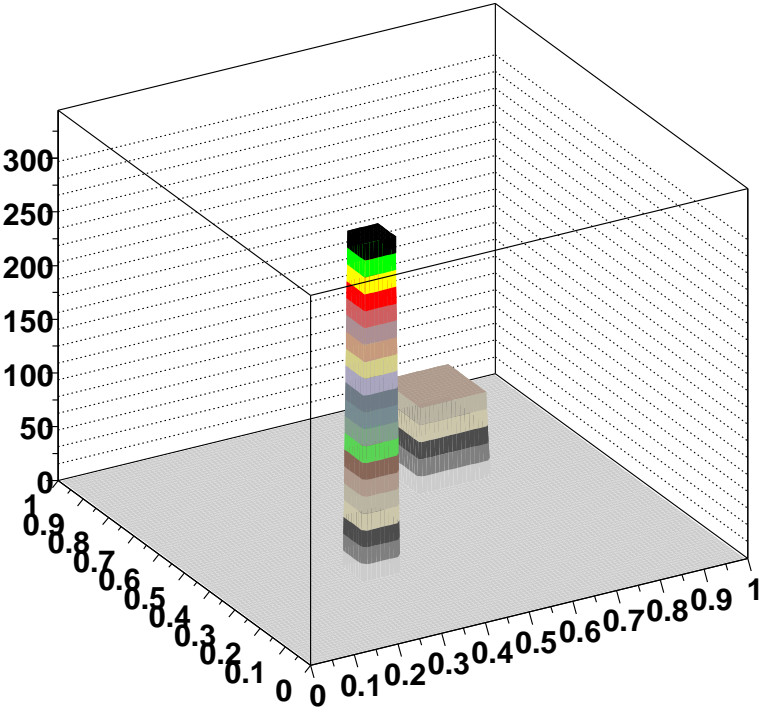
VEGAS is designed as MC integrator. It can be turned into MC generator relatively easily. However, VEGAS algorithm does not provide for W_{\max} -reduction.

One has to be also aware of binning oscillations. The question is: the binning from which iteration to use for MC generation? For simple functions from the second.

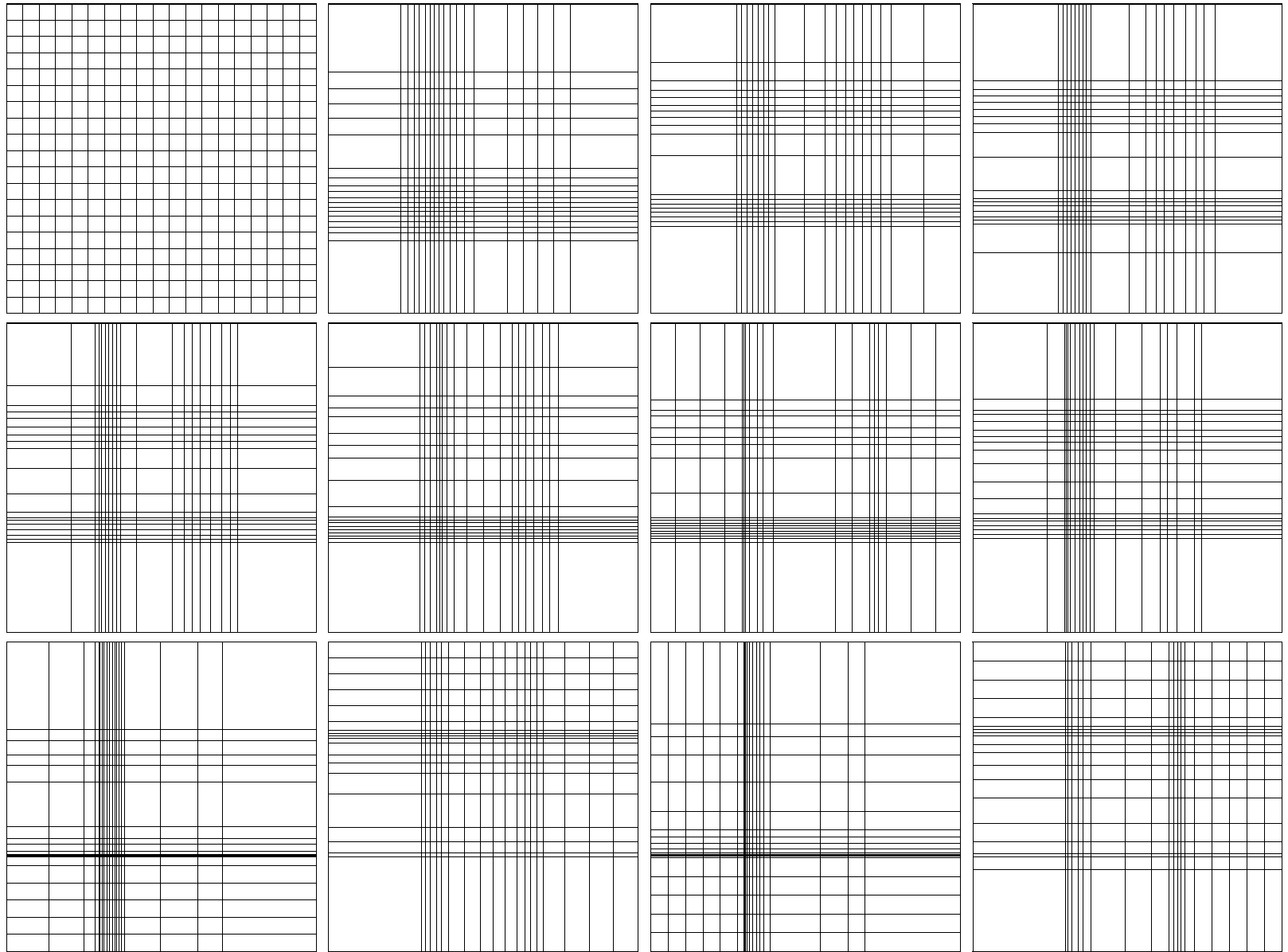
VEGAS deficiency

For a given $\rho(\vec{x})$ there is certain limiting value of the MC Simulation efficiency $\frac{\langle W \rangle}{W_{\max}}$ which cannot be overcome by further enlarging the grid and/or number of the MC trials. This is due to factorizability requirement.

Integrand distribution and Vegas factorizable ansatz $f(x) \times g(y)$:



VEGAS grid oscillations, iterations 1-12:



Cellular approach: Assumptions and Aims

- **Take care of several ($n < 20$) “wildest variables”.** These with the strongest singularities. Other to be “averaged over” as in BASES.
- **No factorizability assumption. Arbitrary integrand.** Singularities on arbitrarily shaped hyperspaces like ridges along diagonals, and “thin” hypersurfaces like sphere etc. Also big voids. (For extremely narrow peaks, it always make sense to map variables.)
- **Integrand positive and integrable but weak integrable singularities should be allowed.** Singularities of the type \sqrt{x} or $\ln(1/x)$.
- **INITIALIZATION: a “foam of cells” is built, adapting automatically to the integrand.** The resulting simulation efficiency $\langle W \rangle / W_{\max}$, limited by CPU and memory, not by the algorithm.
- **Iterative improvement of the foam is desirable as an option. (Not implemented.)**
- **EVENT GENERATION: one cell is chosen and a point within this cell.**

Just to give an idea...

Modern PC desktop can create within 15minCPU $n_C = 5\text{k}-50\text{k}$ cells and subsequently generate $N = 10\text{M}$ events for relatively strongly peaked n -dimensional integrand, $n < 12$.

With $\frac{\sigma}{\langle W \rangle} = 0.3$ that provides 3-digit value of the integral,

$$\frac{\delta R}{R} = \frac{\sigma}{\sqrt{N}} \simeq 10^{-3},$$

or alternatively 3M events with $W = 1$, assuming $\frac{W_{\max}}{\langle W \rangle} = 0.3$.

Not bad!! What are the key limitations?

Memory: $\sim 16 \times n \times n_C$ Bytes. $< 50\text{MB}$.

CPU time: $\sim n_C/1\text{k}$ Minutes CPU. $< 1\text{h}$.

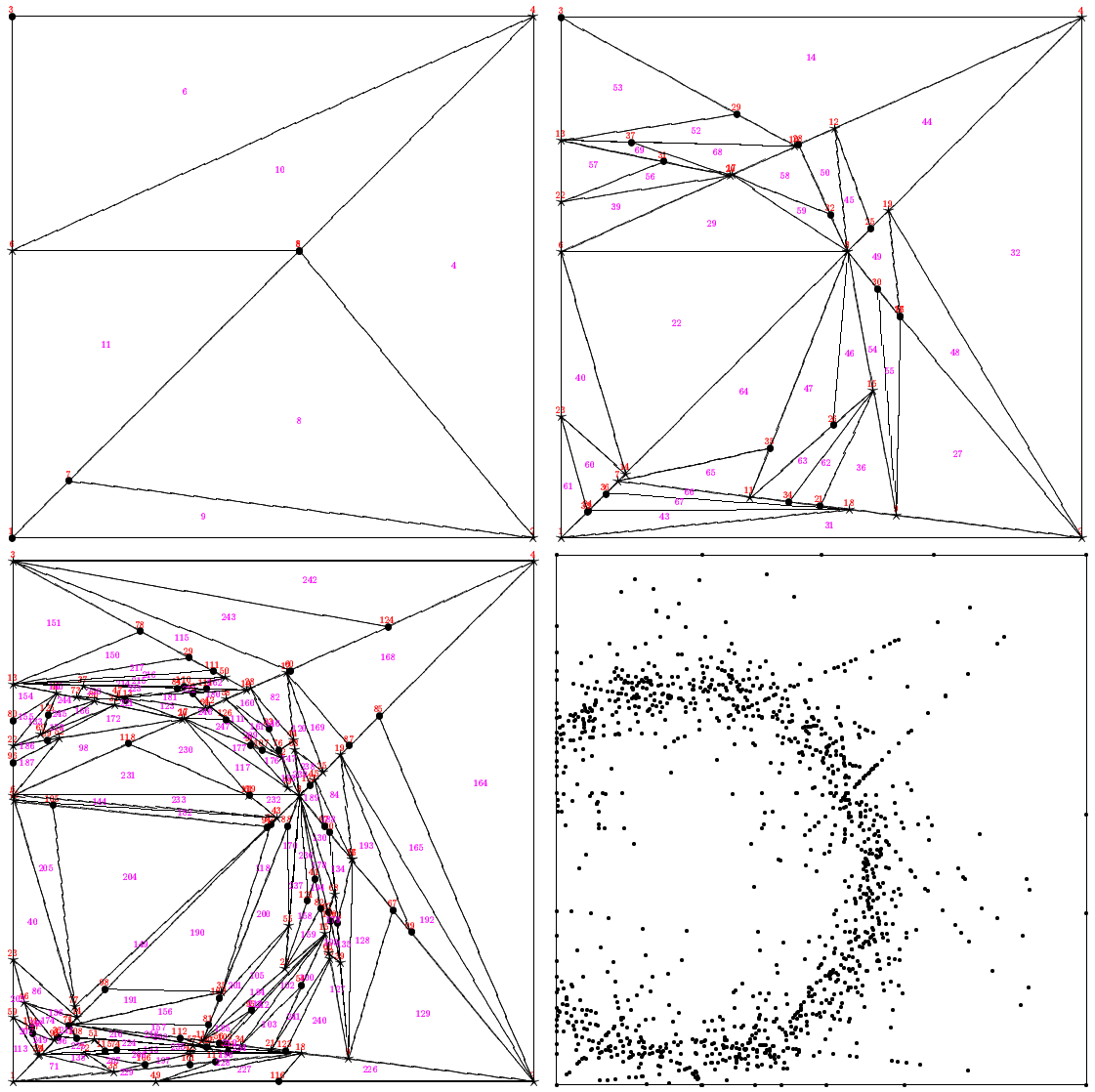
Efficient algorithm of build-up of cells:

maximizing $\frac{W_{\max}}{\langle W \rangle}$ and/or minimizing $\frac{\sigma}{\langle W \rangle}$.

Algorithms based on BINARY SPLIT of cells seems to fulfill the constraints.

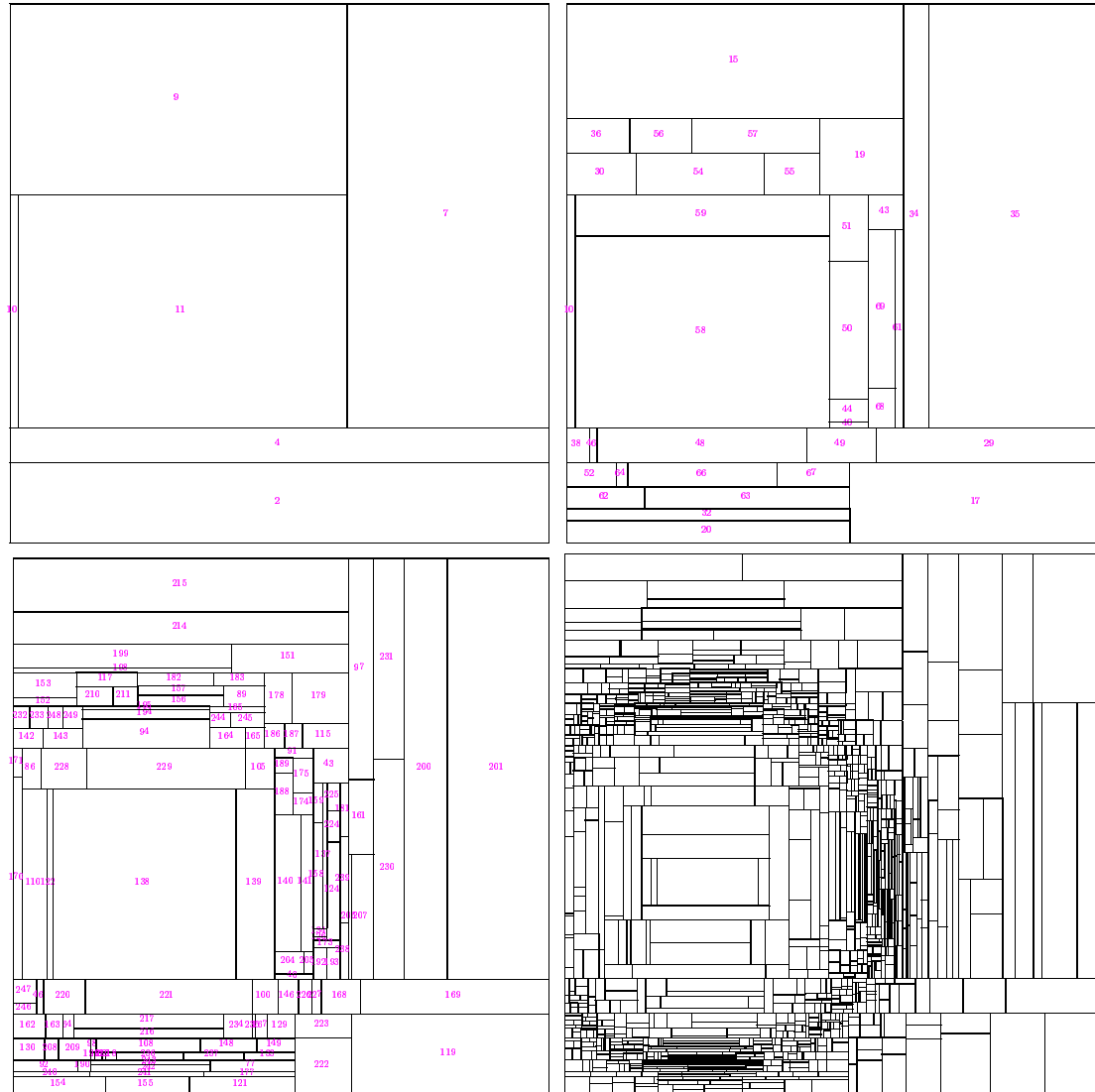
Parent cells kept in record and hierarchical data organization (tree-like linked list).

Evolution of simplicial foam



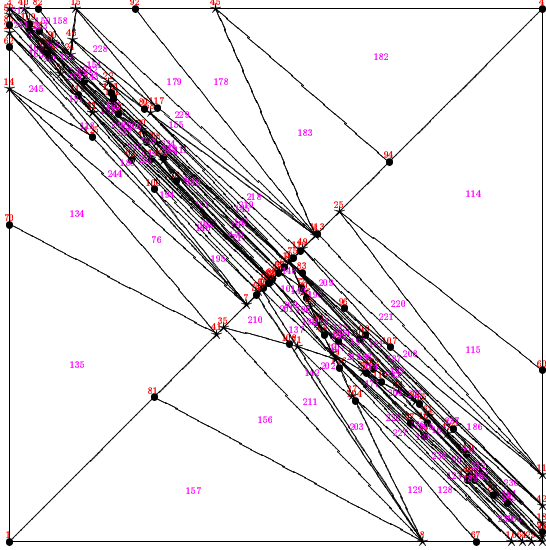
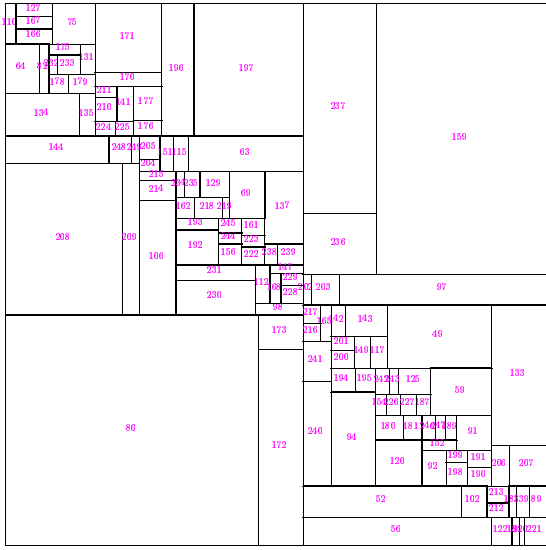
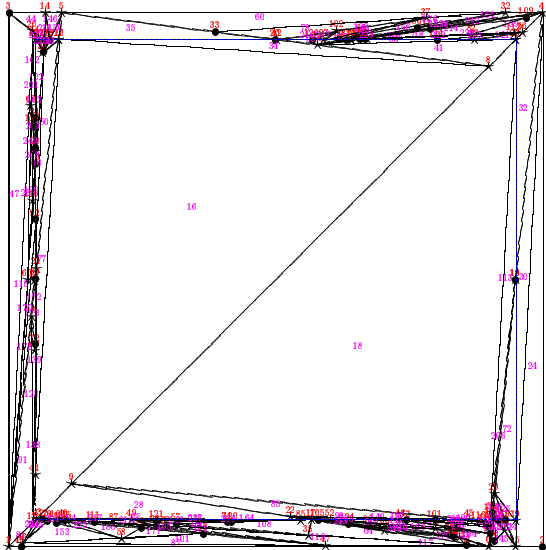
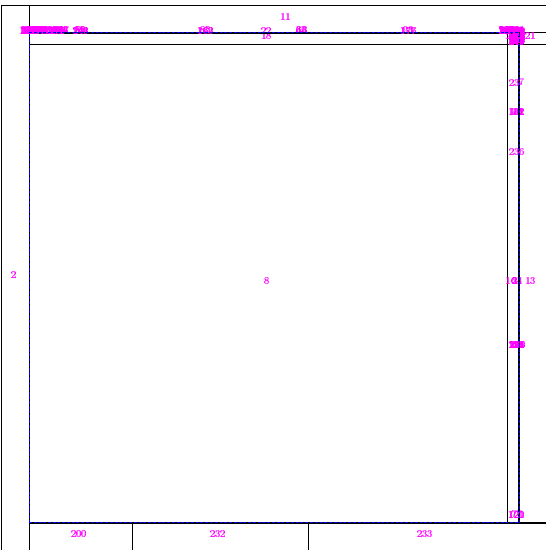
Number of cells= 10, 70,
250, 2500.

Evolution of hyper-cubic. foam



Number of cells= 10, 70,
250, 2500.

Void and Diagonal



Number of cells= 250.

Terminology & Notation

Integration domain $\Omega = [0, 1]^n$ is split into Cells $\omega_1 \dots \omega_k$.

Two AUXILIARY distributions $\rho'(x)$ and $\rho_{loss}(x)$ related to integrand $\rho(x)$, which are CONSTANT over each cell are constructed. How? See next slides.

ALWAYS TWO STEP PROCEDURE:

(1) Foam of cells BUILD-UP:

$\rho_{loss}(x)$ is evolving with the foam, it DRIVES the division of cells,

$R_{loss} = \int \rho_{loss} d^n x$ is minimized in the process.

(2) MC generation:

Events are generated according to $\rho'(x)$.

$R' = \int \rho' d^n x$ is known exactly.

$R = R' \langle W \rangle'$ where $W = \rho/\rho'$,

The average $\langle \dots \rangle'$ means over events generated according to ρ'

OUR AIMS:

Minimize rejection (loss) rate $L = 1 - \frac{W_{\max}}{\langle W \rangle}$, OR

Minimize variance $\sigma = \sqrt{\langle W^2 \rangle - \langle W \rangle^2}$.

binary split of a cell

How do we minimize $L = 1 - \frac{\langle W \rangle}{W_{\max}}$?

DEF: $\rho'(x) \equiv \max_{y \in Cell_i} \rho(y)$, for $x \in Cell_i$.

is a maximum of $\rho(x)$ within each cell:

Maximum over cell found experimentally using ~ 1000 MC events.

We MINIMIZE during the foam construction:

$$R_{loss} = \int d^n x [\rho'(x) - \rho(x)] = \int d^n x \rho_{loss}(x)$$

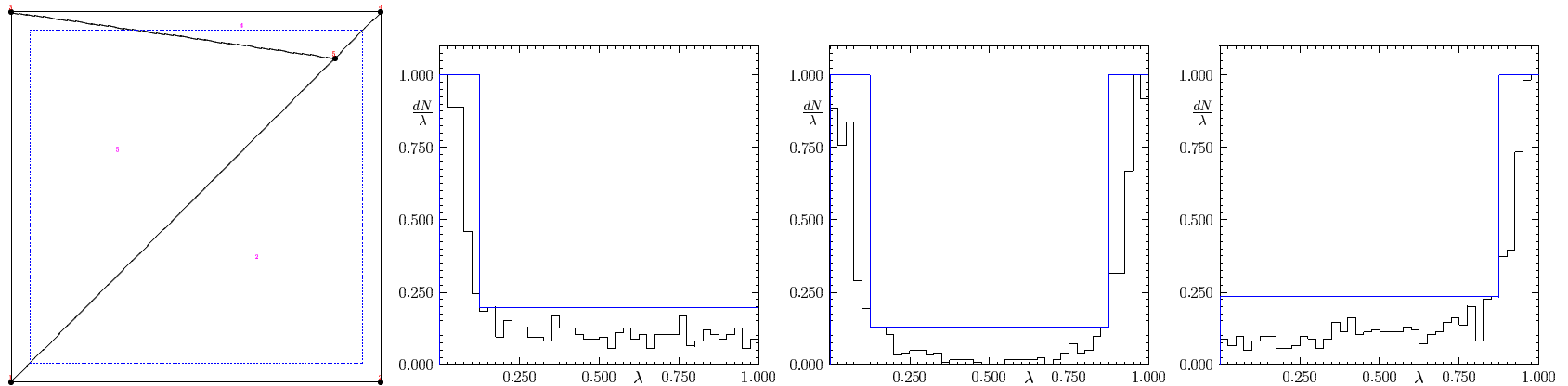
Each SPLIT of a Cell: $\omega \rightarrow \omega' + \omega''$

should obey: $R_{loss}^{\omega'} + R_{loss}^{\omega''} \ll R_{loss}^{\omega}$.

RULES to follow:

- For next split we choose a CELL with the biggest R_{loss} .
- Position/direction of a plane dividing a parent CELL into two daughter CELLS is chosen to minimize total R_{loss} .

Binary split 2-dim. example



Integrand covers narrow strip along edges. Intend to split upper triangle.

Generate 1000 w-ted events and project them on 3 sides of the parent triangle.

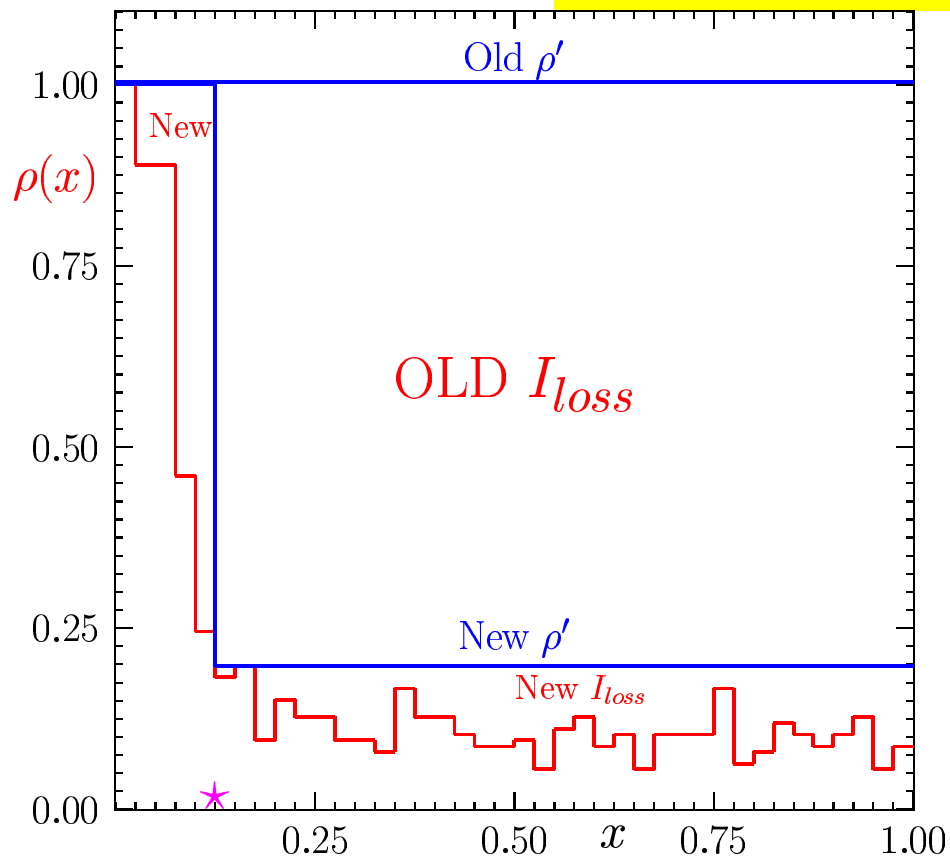
3 Projections are analyzed.

We choose the one with the smallest LOSS functional I_{loss} (middle plot).

Two resulting daughter triangles are shown.

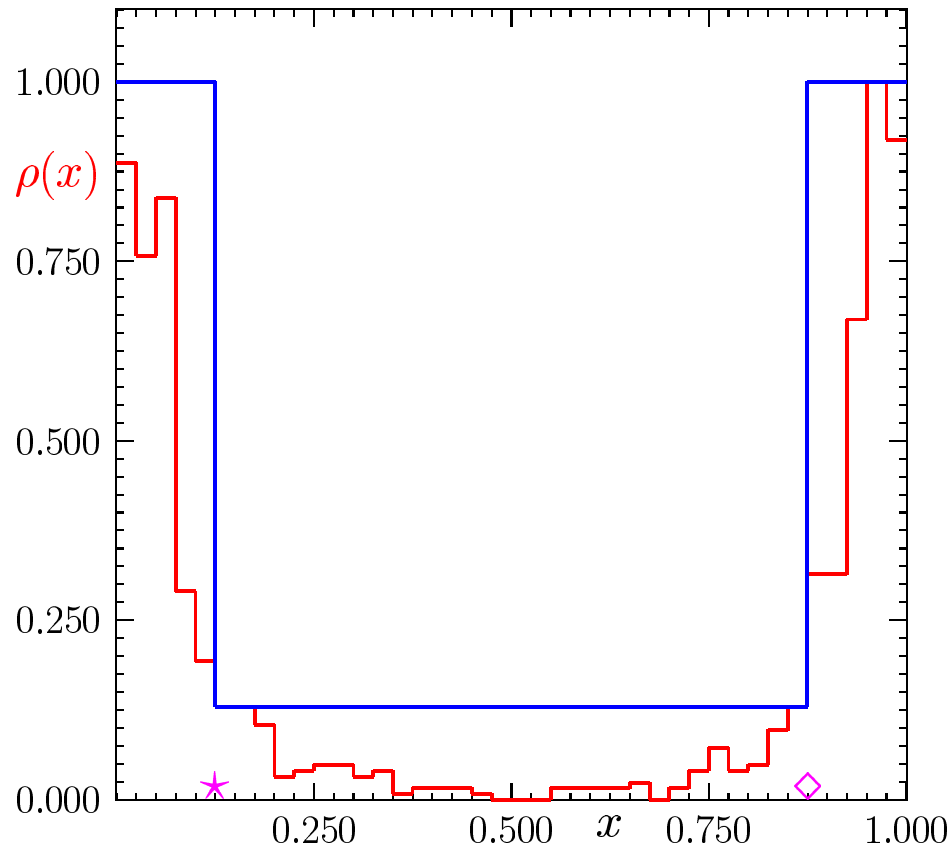
The DIVISION line EXPELLED from the “empty” area in the middle!!!

Binary split 2-dim. example



$\rho(x)$ is integrand. **Old ρ'** for parent cell (majorizing $\rho(x)$). **New ρ'** for two daughter cells. **OLD I_{loss}** all area above red line, for the parent cell. **New I_{loss}** between red line and **New ρ'** , for 2 daughters. Obviously $I'_{New} < I'_{Old}$, the division point \star is chosen to MINIMIZE THE LOSS functional/integral I_{loss} !

Binary split 2-dim. example



The two-maximum case is a little bit more complicated. We follow simple prescription: Determine the **blue line** MAXIMIZING I_{loss} , see picture, and apply the division procedure twice (obtaining 3 cells). Division points are \star and \diamond

Minimizing variance also available

How to Minimize $\frac{\sigma^2}{\langle W \rangle^2} = \frac{\langle W^2 \rangle}{\langle W \rangle^2} - 1$?

For the moment the following temporary solution is used:

DEFINE: $\rho'(x) \equiv V_i \times \sqrt{\langle \rho^2 \rangle_i}$, for $x \in Cell_i$.

The $\langle .. \rangle_i$ is for uniformly distributed points $\in Cell_i$.

Cells with bigger contribution to total variance, “promoted” in MC generation.

During the foam build-up MINIMIZE:

$I_{loss} = \int d^n x \rho_{loss}(x)$ where

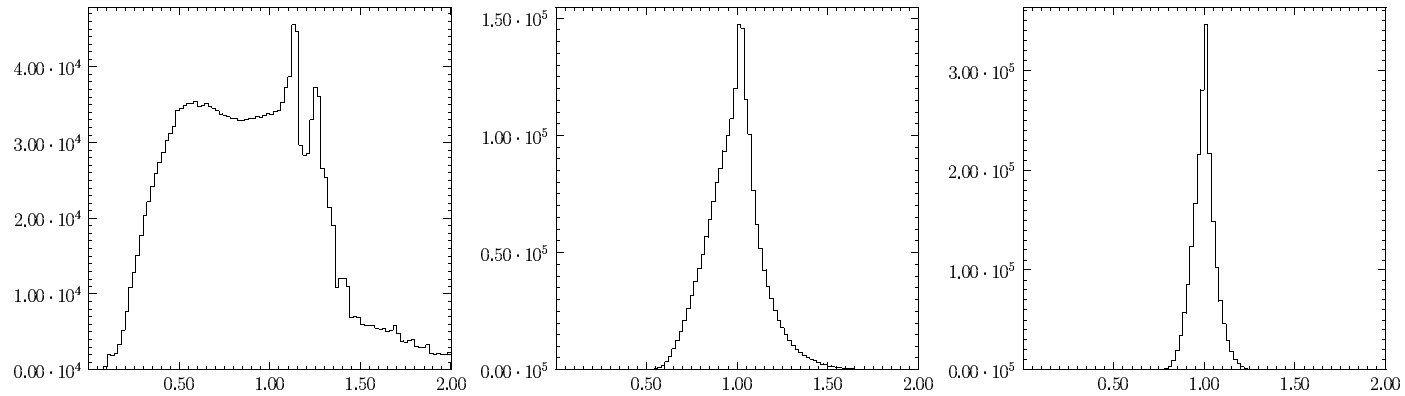
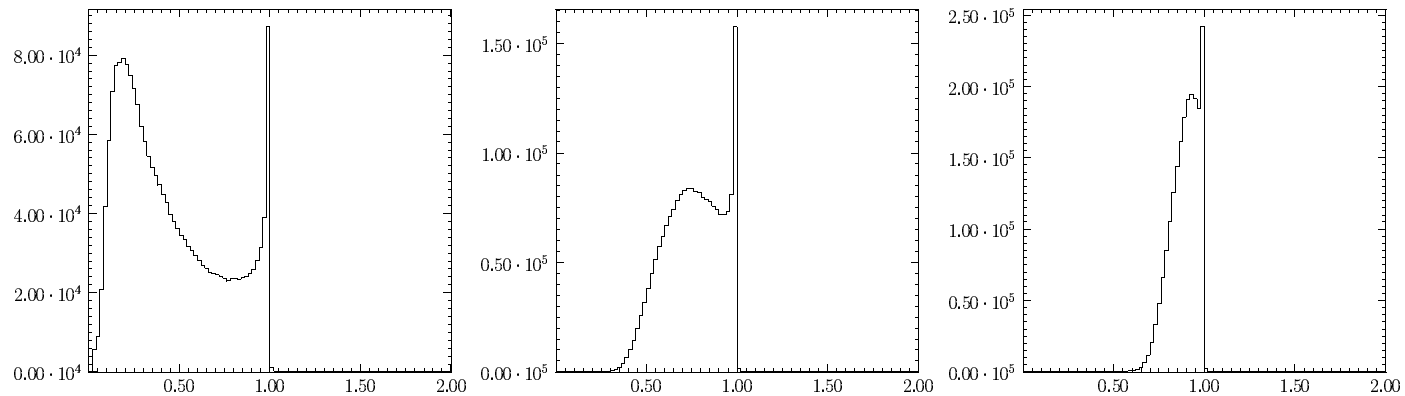
$\rho'(x) \equiv V_i \times \sqrt{\langle \rho^2 \rangle_i - \langle \rho \rangle_i^2}$, for $x \in Cell_i$.

The algorithm avoids dividing cells in the areas where the integrand is flat,

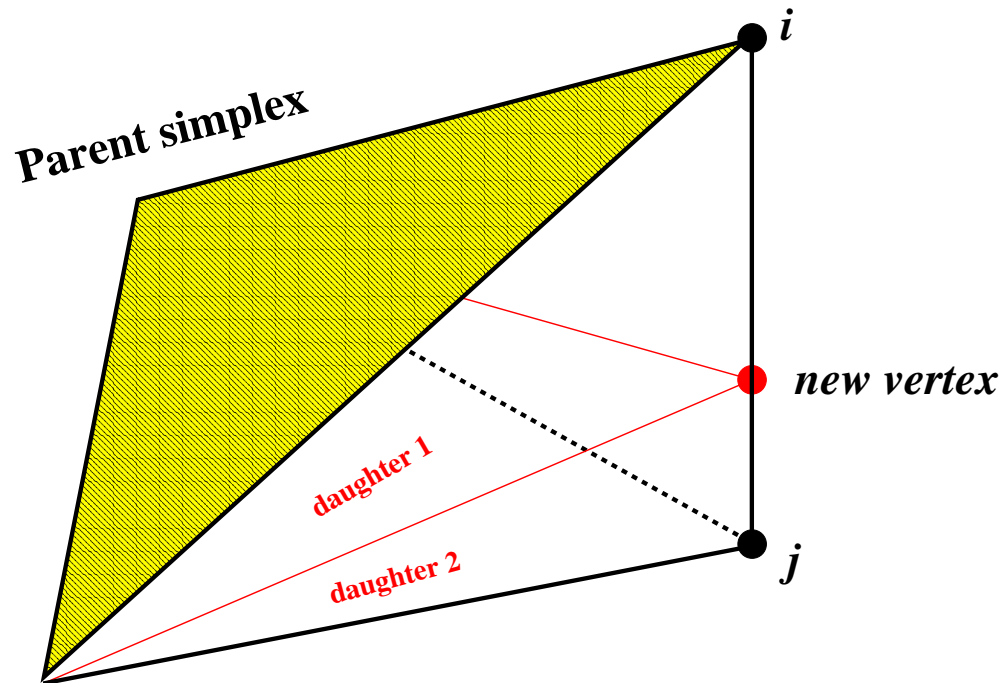
the dispersion of the function over the cell is small.

The algorithm of cell division remains the same.

There was less effort to optimize this mode of the program.

Weight distribution: minimization of variance**Number of cells: 200, 2k, 20k****Weight distribution: minimization of rejection rate****Number of cells: 200, 2k, 20k**

Geometry of n -dim. Simplicial Cell division



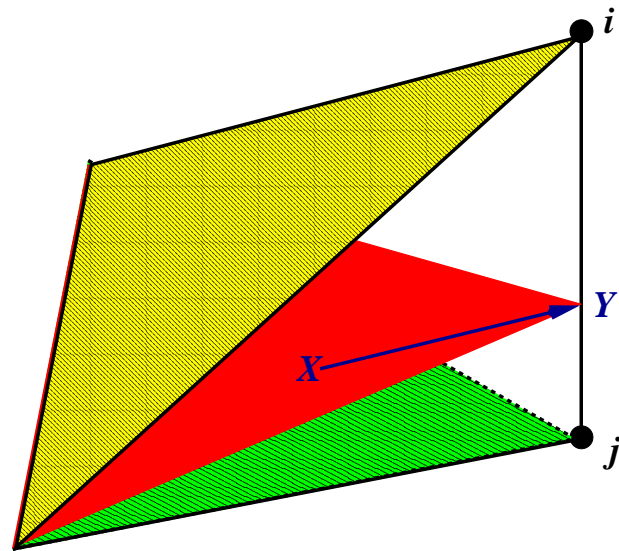
Pair of vertices x_i and x_j is chosen and a new vertex Y is put somewhere on the line in between: $Y = \lambda x_i + (1 - \lambda)x_j$, $0 < \lambda < 1$

Two daughter simplices are defined with the list of vertices:

$$(x_1, x_2, \dots, x_{i-1}, Y, x_{i+1}, \dots, x_{j-1}, x_j, x_{j+1}, \dots, x_n, x_{n+1}),$$

$$(x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_{j-1}, Y, x_{j+1}, \dots, x_n, x_{n+1}).$$

Geometry of n -dim. Simplicial Cell division



How do we choose (i, j) pair and the value of λ ?

Short sample of the MC events (100-1000) is generated \in cell.

Each MC point projected $X \rightarrow Y$ onto a given edge $(i, j), i \neq j$:

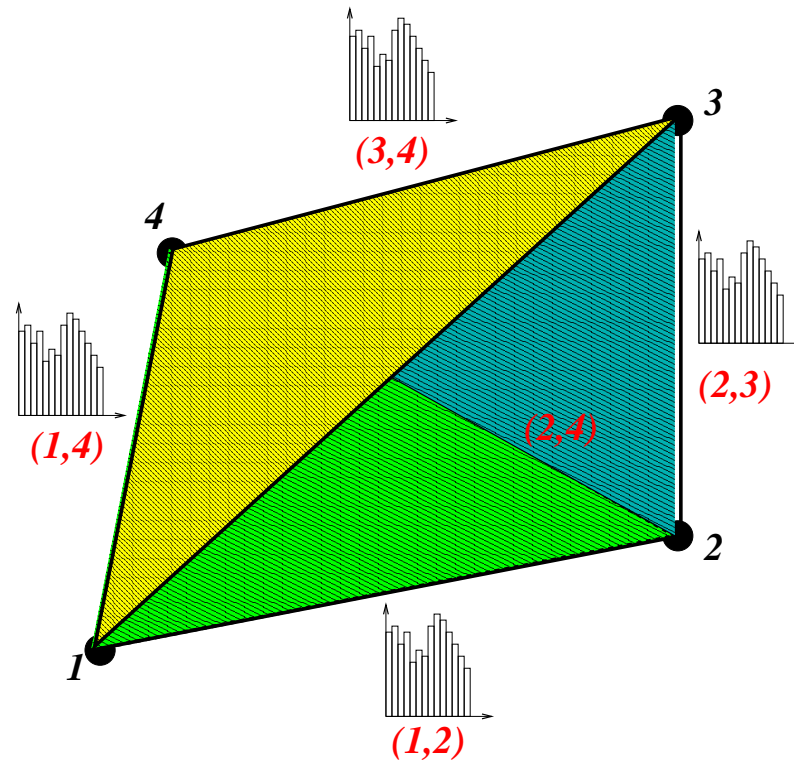
$$Y = \lambda_{ij}x_i + (1 - \lambda_{ij})x_j, \quad \lambda_{ij}(X) = \frac{|\text{Det}_i|}{|\text{Det}_i| + |\text{Det}_j|},$$

$$\text{Det}_i = \text{Det}(r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n, r_{n+1}),$$

$$\text{Det}_j = \text{Det}(r_1, \dots, r_{j-1}, r_{j+1}, \dots, r_n, r_{n+1}), \quad r_k = x_k - X,$$

where $\text{Det}(x_1, x_2, \dots, x_n)$ determinant.

Choice of the future division edge



How do we select (i, j) ? out of $\frac{n(n-1)}{2}$ possibilities.

For each (i, j) the $dN/d\lambda$ is histogrammed, and its LOSS functional R_{loss} is estimated. The edge (i, j) with the biggest LOSS is selected! For the cell division λ is read from the histogram. λ is always a rational number, n/N_{bin} !

Why hypercubes as cells? Why not?

Consideration in favor of simplices: MEMORY: In n -dimensions, simplicial foam (binary division) requires only one vertex vector \vec{v} for each new cell! That means: $8n$ Bytes/Cell.

Every new hypercubic at $n=15$ has 32k vertices :-)

However, it can be parameterized using only 2 vectors: \vec{v} for “lower left corner” vertex position, and \vec{d} defining n lengths along each direction :-)

$16n$ Bytes/Cell. (The interior of the hypercube is $x^i = v^i + \lambda_i d^i, 0 < \lambda_i < 1$.)

Consideration in favor of hypercubes:

Simplices limited to $n < 6$ and $N_{Cell} < 5k$ because:

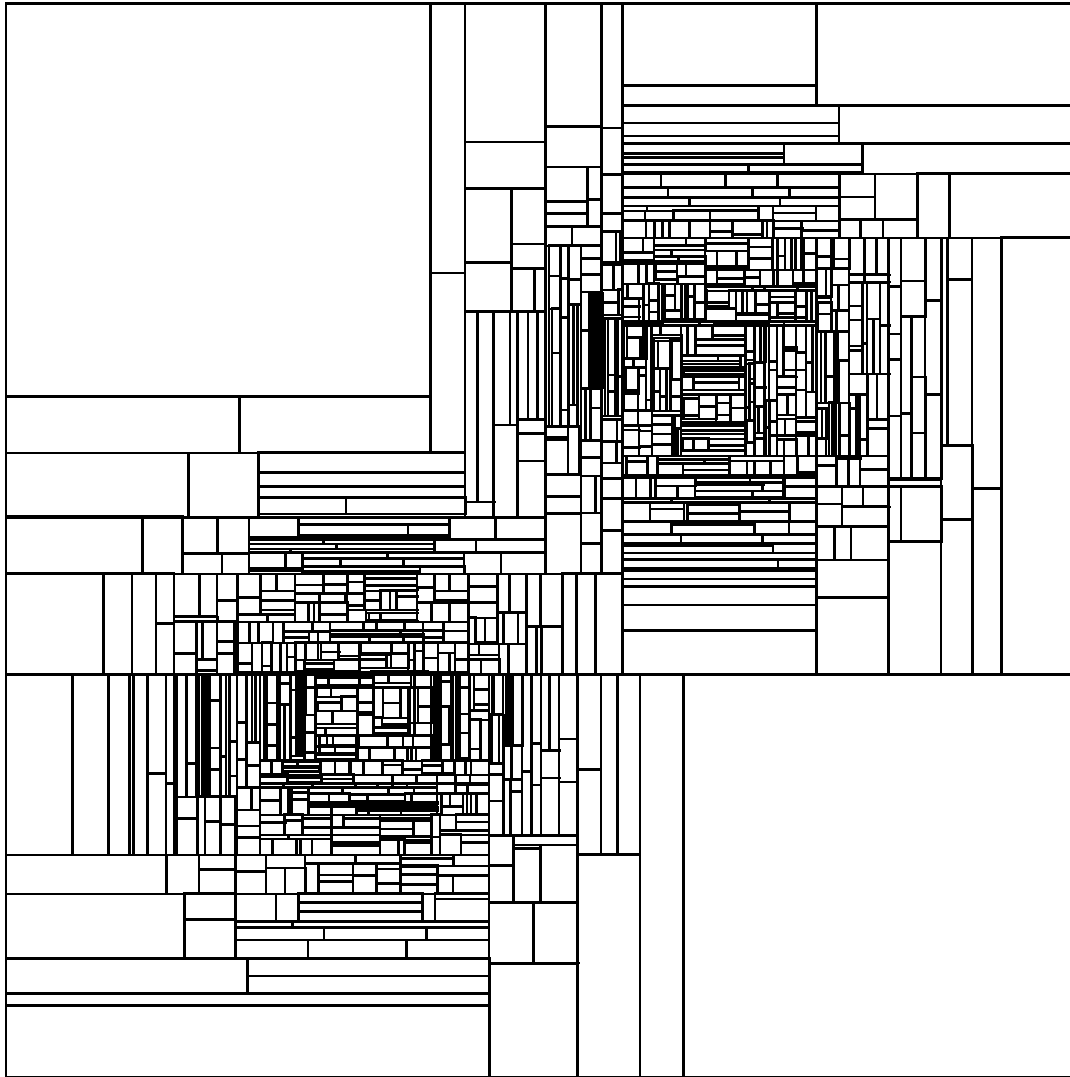
- (1) Already for $n=10$ initial h-cube $\rightarrow n! \sim 10^6$ simplices,
- (2) geometry become CPU consuming (determinants).
- (2) memory limitation,

However, memory consumption for h-cubes can be limited to

below 50Bytes/Cell indep. of n , albeit with some tolerable CPU overhead, no determinants :-)

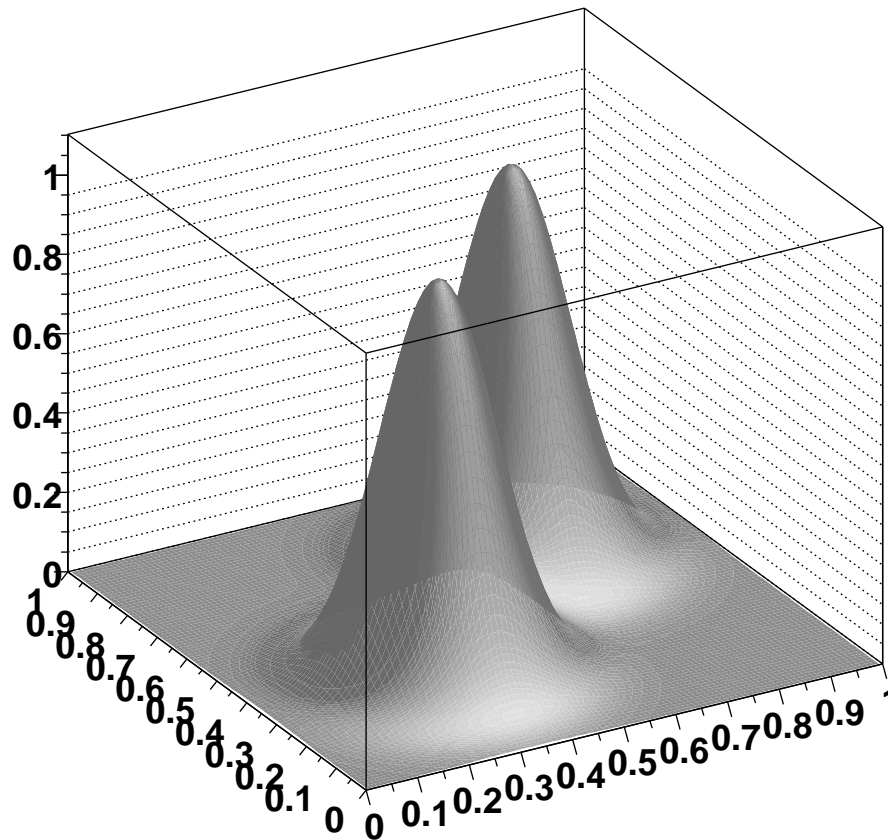
BOTTOM LINE: With hypercubes 1Mega Cells at $n \sim 15$ can be reached!

Example: Hyper-cubics (= rectangles) in 2-dimen.



The “Camel” Double-Gauss function of Peter Lepage.

Testing function in n-dim. Here for n=2:



The “Camel” Double-Gauss function of Peter Lepage.

Miracle of ~ 50 Bytes/Cell

Two essential ingredients:

- \vec{v} and \vec{x} are not stored but (re)calculated using the division parameter λ and the index i of the edge along which the division occurred, all over $\sim \ln_2 N_{Cell}$ steps, up to first Cell.
- Furthermore parameter $\lambda = j/n_{bin}$ can be stored as a short inter j (2Bytes sufficient).

Each cell has several other attributes like R' , R_{loss} (16B), pointers to parent and daughter cells (12B), status (8B) etc., altogether ~ 50 B/Cell. Can be easily reduced to 32B/Cell if necessary.

BOTTOM LINE:

The way to 1Mega Cells is open. We hit CPU barrier instead. See next slide.

CPU barrier: one quick fix is found

Numeric experiments show that the obtained efficiency $\frac{\langle W \rangle}{W_{\max}}$ and variance σ are improved **ESSENTIALLY** by the **INCREASING** of N_c .

CPU time is proportional to the number of MC events N_{samp} used to explore each newly created Cell. Can we limit it somehow?

Also, necessary value of N_{samp} increases for higher n . Hence, dramatic increase of total CPU time for the foam build-up, which is $T \sim n \times N_c \times N_{\text{samp}}$.

The trick limiting the above effect:

During MC exploration of a new cell continuously monitor the no. of accumulated effective $W = 1$ events: $N_{\text{eff}} = \frac{(\sum w_i)^2}{\sum w_i^2}$ and stop when $N_{\text{eff}}/n_{\text{bin}} > 25$, where n_{bin} is the number of bins in the histogram used to estimate the best division direction/edge and parameter.

The essence of the trick: The increase of N_{samp} not wasted for cells in which integrand is varying very little.

We are now able to push input N_{samp} to high values, with little increase of CPU time.

Programs, available from the author

MCell v2.02, f77: with up to 1Mega cells is specialized for higher dimensions, $n \leq 20$, hypercubical cells only.

Foam v2.02, f77 : with $N_c \leq 10000$ cells, is aimed for up to six-dimensions. It is upgraded and improved: both simplices ($n \leq 5$) and hypercubes ($n \leq 10$) available. Low memory consumption optionally.

Foam v2.02, c++ : unlimited number of cells N_c and dimension $n\text{Dim}+k\text{Dim}$. Both simplices and hypercubes available. Includes hypercubical algorithm MCell with low memory consumption as default option.

Foam v2.02, c++: using ROOT package (R. Brun et.al.)

Other improvements:

- Cells can be simultaneously simplices in n -dimensions and hypercubics in k -dimensions.
- It is possible to start FOAM from SINGLE simplex instead of unit hyper-cubic.

Tests of Foam at low dimensions

Three 2-dimensional testing functions:

$$f_a = 1 - \Theta(0.5 - |x_1 - 0.5| - \gamma) \Theta(0.5 - |x_1 - 0.5 - \gamma|), \quad \gamma = 0.05,$$

$$f_b = \frac{1}{4\pi R^2} \frac{\gamma}{\pi[(R - \sqrt{(x_1 - 0.25)^2 + (x_2 - 0.40)^2})^2 + \gamma^2]},$$

$\gamma = 0.02, R = 0.35$

$$f_c = \frac{\gamma}{\pi[(x_1 + x_2)^2 + \gamma^2]}, \quad \gamma = 0.02.$$

Functions, 2-dimens.	Simplic.	H-cubes	VEGAS
$f_a(x_1, x_2)$ (diagonal ridge)	0.94	0.74	0.03
$f_b(x_1, x_2)$ (circular ridge)	0.80	0.80	0.16
$f_c(x_1, x_2)$ (edge of square)	1.00	1.00	0.53

Results from Foam/MCell are for 5000 cells (2500 active cells) and cell exploration based on 200 MC events/cell.

Efficiencies are $\langle W \rangle / W_{\max}^\varepsilon$ with $\varepsilon=0.0005$.

Compare 2 and 3 dimensions

Functions, 2-dimens.	Simplic.	H-cubes	VEGAS
$f_a(x_1, x_2)$ (diagonal ridge)	0.94	0.74	0.03
$f_b(x_1, x_2)$ (circular ridge)	0.80	0.80	0.16
$f_c(x_1, x_2)$ (edge of square)	1.00	1.00	0.53

Functions, 3-dimens.	Simplic.	H-cubes	VEGAS
f_a (thin diagonal)	0.56	0.62	0.002
f_b (thin sphere)	0.27	0.50	0.11
f_c (surface of cube)	0.66	1.00	0.30

Keep in mind that efficiency of Foam $\langle w \rangle / w_{\max}$ can be in the above example easily increased to almost 100% by increasing no of cells, while for VEGAS we are here at the limitig value!

(Comment: Is this stratified sampling?)

Tests of Foam at higher dimensions

Camel test-function of P. Lepage, normalized to one:

n	N_{Cell}	$\frac{N_{MC}}{Cell}$	Effic.	$\frac{\sigma}{\langle W \rangle}$	I	δI
6	10k	0.3k	0.0387	1.18	0.99789	0.00083
6	10k	1k	0.1275	1.22	1.00007	0.00086
6	10k	10k	0.1231	1.28	1.00070	0.00090
6	10k	33k	0.1325	1.21	1.00040	0.00086
6	100k	0.3k	0.2574	0.75	0.99939	0.00053
6	100k	1k	0.2626	0.77	1.00030	0.00054
6	100k	3k	0.2750	0.76	1.00016	0.00053
6	100k	10k	0.2681	0.78	1.00022	0.00055
9	100k	1k	0.0336	1.74	0.99285	0.00123
9	100k	3k	0.0435	1.84	0.99765	0.00130
9	100k	10k	0.0407	1.89	0.99871	0.00133
12	100k	1k	0.0037	3.13	0.84019	0.00221
12	100k	3k	0.0038	3.33	0.48950	0.00235
12	100k	10k	0.0036	4.48	0.99512	0.00317
12	100k	100k	0.0032	5.20	1.00003	0.00368
12	1000k	10k	0.0055	3.89	0.99779	0.00275

Efficiency depends mainly on number of cells M_{Cell} .

Number of MC trials per Cell cannot be too small.

Conclusions

- **Adaptive adjustment of branching probabilities is available.**
- **VEGAS is alive and well.**
- **Cellular algorithms come to state of art.**