FOAM: general purpose

Monte Carlo Cellular Algorithm

S. Jadach

Institute of Nuclear Physics, Kraków, Poland

Presented with help of Thorsten Ohl

Outline:

- **Introduction and motivation**
- **Cellular algorithm of FOAM**
- **Examples of numerical results**
- **Conclusions**

These and related slides on http://home.cern.ch/jadach

Introduction: What is general purpose?

For the problem of function minimalization one takes MINUIT or some other program and applies it to arbitrary user-function. One may find also general purpose programs for integration of "arbitrary" integrand. General-purpose means that these tools work, in principle, for a very wide range of user-functions.

For multi-dimensional Monte Carlo simulation problem, that is for the problem of generation randomly points according to a given n-dimensional distribution, there is precious little examples of the General Purpose Monte Carlo Simulators (GPMCS), that is programs which work (in principle) for arbitrary integrand.

Two essential reasons for scarcity of GPMCS's:

(a) lack of ideas about the efficient algorithm,

(b) need of much CPU power and memory – only recently available/affordable.

General features of General Purpose Monte Carlo Simulators (GPMCS)

Inevitably the GPMCS has to work in 2 stages: exploration and generation.

During exploration GPMCS is "digesting" the entire shape of the n-dimensional distribution $\rho(x_1, x_2, ...x_n)$ to be generated and memorize it as efficiently as possible using all CPU power and memory available.

Obviously, for the memorized $\rho'(x_1, x_2, ...x_n)$ a method of the MC generation of the points $\vec{x}$ exactly according to $\rho'(\vec{x})$, has to be available.

The quality of the distribution of the weight $w = \rho/\rho'$ for events in the generation (small variance, good ration of maximum to average, etc.) is determined by the algorithm of the exploration. In other words, the "target weight distribution" in the generation is determining the algorithm of exploration.

The GPMCS programs will be always limited to "small dimensions".
With presently available computers "small" means in practice $n < 10$
(up to $n < 15$ for certain function). This is already not so bad!!!

Cellular exploration of the distribution

The most obvious method to minimize the variance (or maximum weight) of the target weight distribution in generation (proposed already some 40 years ago) is to split integration domain into many cells, such that $\rho(\vec{x})$ is approximated by constant $\rho'(\vec{x})$ within each cell. This is a cellular class of general purpose MC algorithms.

(I think that “stratifi ed sampling”, used in the literature, has a narrower meaning.)

FOAM: shape of cells, how to cover space with cells?

Three shapes of cells are used pure simplices, pure hypercubes and Cartesian products of them. They can be rather easily and efficiently parametrized.

The system of cells can be created all at once (like in Vegas) or in a more evolutionary way, by the “split process”. In the Foam algorithm we rely on the binary split of cells.)Choice of a cell to be split driven by target weight distribution.)

The binary split assures automatically full coverage of the space,
simply because the primary “root cell” is the entire integration domain.

Variance reduction versus maximum weight reduction

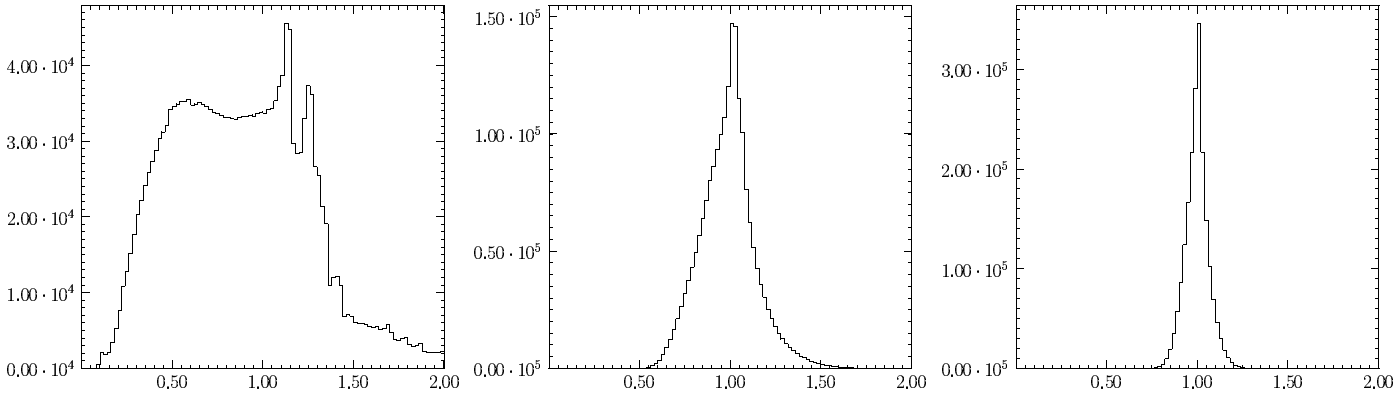
In construction of the FOAM algorithm I have put most effort on the minimization of the ratio of the maximum weight to the average weight $w_{\max}/\langle w\rangle$.
This parameter is essential, if we want to transform w-ted events into $w=1$ events, at the latter stage of the MC generation.

Minimizing maximum weight is not the same as minimizing variance $\sigma=\sqrt{\langle w^2\rangle-\langle w\rangle^2}$. Usually minimizing $w_{\max}$ is more difficult.

In FOAM minimizing variance is also implemented and optionally available.
It can be useful if case them w-ted events are acceptable.

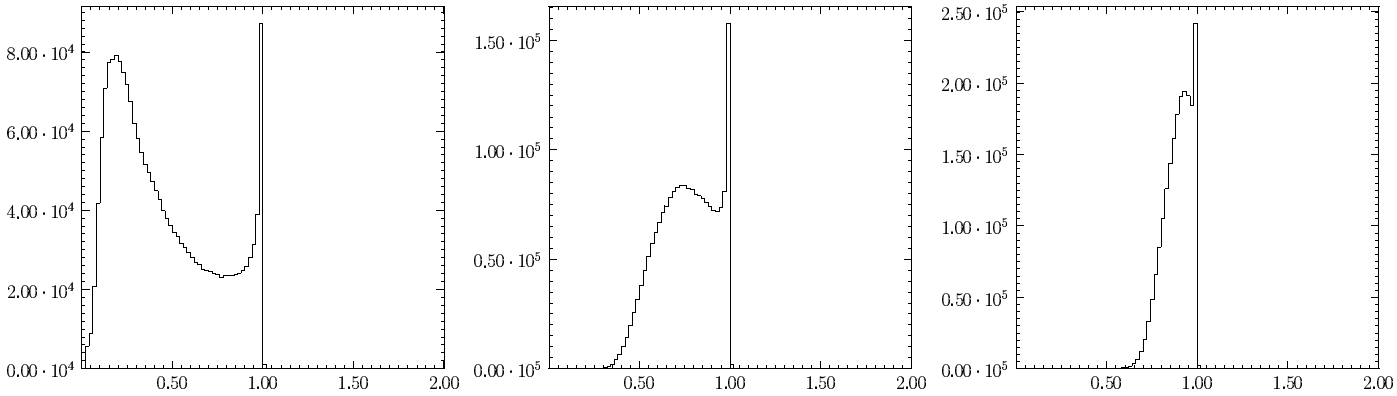
Next slide shows two examples of the weight distribution evolution in the Foam, when adding more and more cells.

Generation weight distribution: minimization of variance



Number of cells: 200, 2k, 20k

Generation weight distribution: minimization of maximum weight



Number of cells: 200, 2k, 20k

Cell split algorithm: covers both (a) $w_{\max}$ and (b) variance minimization

We define two auxiliary distributions $\rho'(x)$ and $\rho_{loss}(x)$ related to integrand $\rho(x)$.

Both are constructed together with the foam of cells, in the exploration process.

(1) EXPLORATION of $\rho(x)$ and BUILD-UP of Foam of cells :

$\rho_{loss}(x)$ and $\rho'(x)$ are evolving in the process of the division of cells.

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

(2) MC event 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 $\langle \ldots \rangle'$ is over events generated according to ρ'.

(a) Minimization of $w_{\max}$

$\rho'(x) \equiv \max_{y \in Cell_i} \rho(y)$, for $x \in Cell_i$, the "ceiling function".

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

Note that rejection rate in final MC run = R_{loss}/R.

(b) Minimization of of variance

$\rho'(x) \equiv \sqrt{\langle \rho^2 \rangle_i}$, for $x \in Cell_i$. The average $\langle \ldots \rangle_i$ is over $i-th$ Cell assuming flat distribution.

$\rho_{loss}(x) \equiv \sqrt{\langle \rho^2 \rangle_i} - \langle \rho \rangle_i$, for $x \in Cell_i$. Final MC variance is just $\simeq R_{loss}$.

Two Rules governing binary split of a cell

Each split of a Cell: $\omega \longrightarrow \omega' + \omega''$ should decrease total R_{loss}.

$$R^{\omega'}_{loss} + R^{\omega''}_{loss} << R^{\omega}_{loss}$$

How to get the best total decrease ΔR_{loss}?

[1] For each next cell split we choose a cell with the biggest R_{loss}.

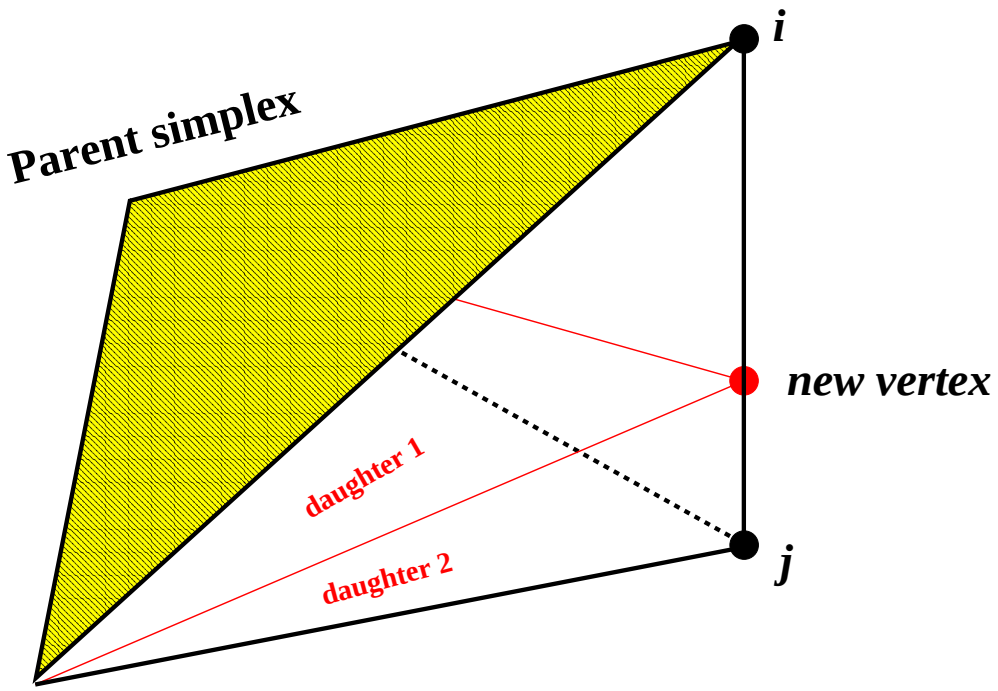
[2] Position/direction of a plane dividing a parent cell into two daughter cells is chosen to get the smallest total R_{loss}.

How do we split a cell into two daughter cells?

General method relies on the small MC exercise on which events are generated with flat distribution, weighted with ρ and projected onto n (simplical case) or $n(n+1)/2$ (h-cubical case) of the cells.

Resulting histograms are analysed and the best "division geometry" found, for which the estimate of ΔR_{loss} is calculated. See next slides...

Geometry of n-dim. Simplical Cell division, 3-Dim. case



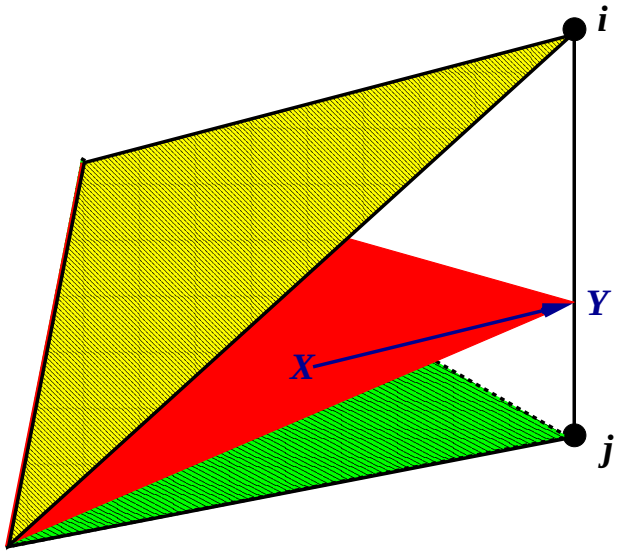
Pair of vertices x_i and x_j is chosen and a new vertex Y is put somewhere on the line in between: $Y = \lambda x_i + (1-\lambda)x_j, \;\; 0 < \lambda < 1$

Two daughter simplices are defi ned with the list of vertices:

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

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

Geometry of n-dim. Simplical Cell division, 3-Dim. case



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 $X \to Y$ onto a given edge $(i,j), i \neq j$:

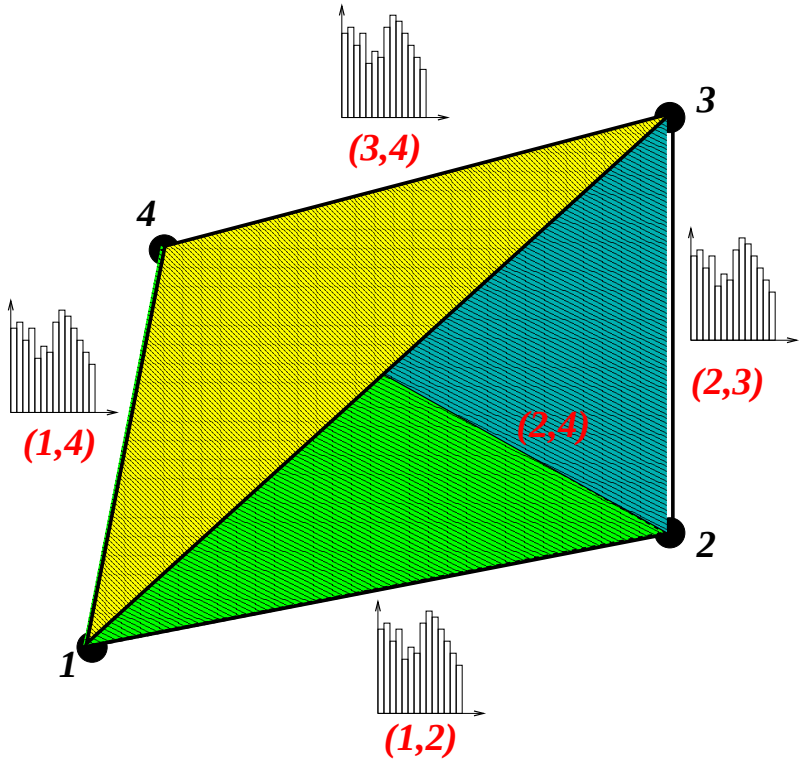
$$Y = \lambda_{ij} x_i + (1-\lambda_{ij}) x_j, \quad \lambda_{ij}(X) = \frac{|\mathrm{Det}_i|}{|\mathrm{Det}_i|+|\mathrm{Det}_j|},$$

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

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

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

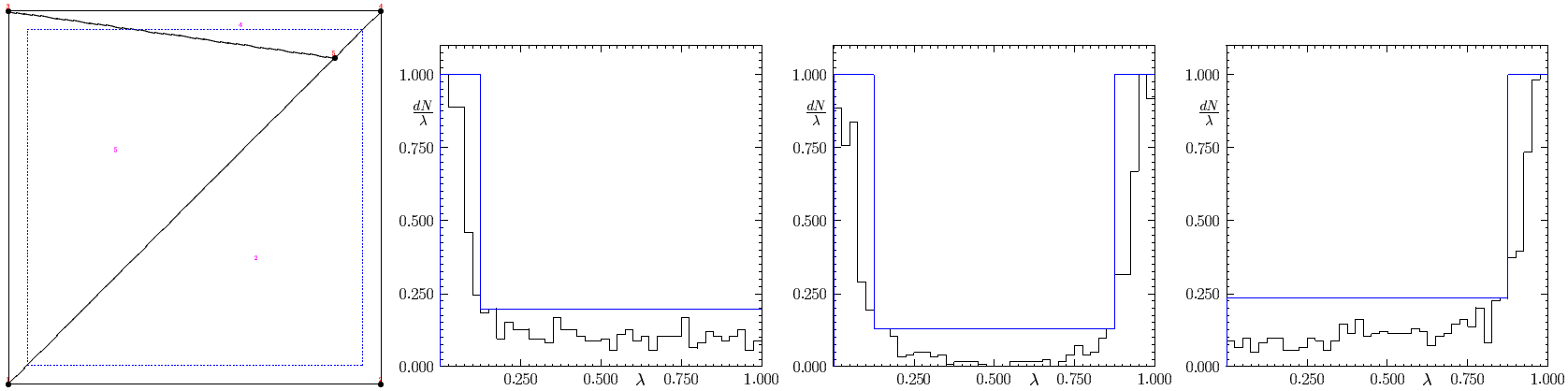
Choice of the best division edge, Simplical 3-Dim. case



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

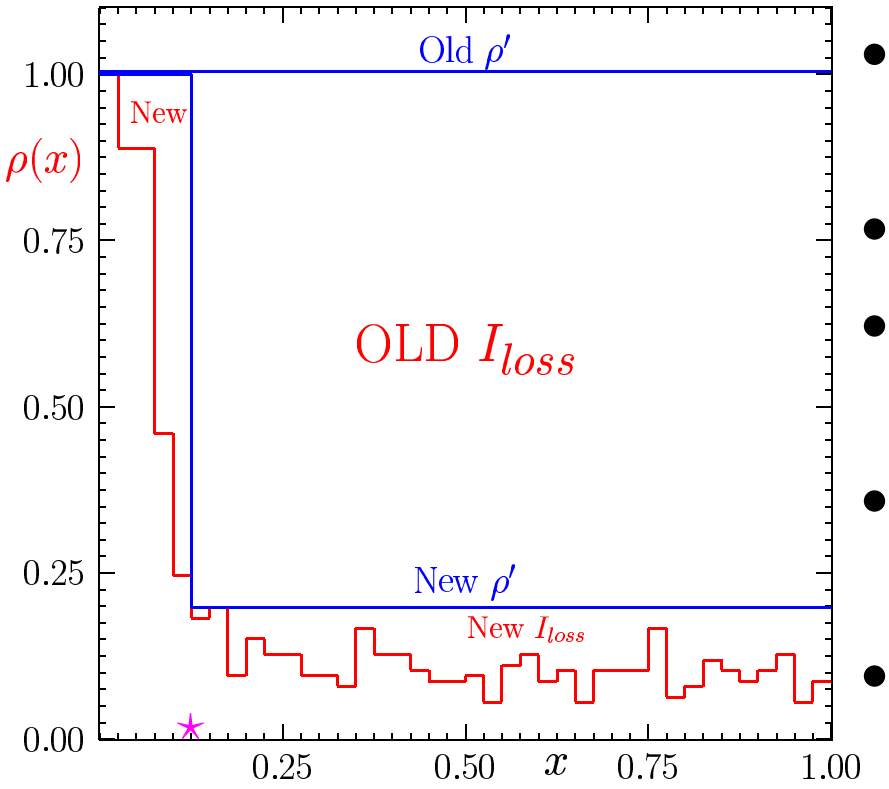
For each (i,j) the $dN/d\lambda$ is histogrammed, and its LOSS functional R_{loss} is estimated. The edge (i,j) with the biggest LOSS is selected! For the cell division λ is read from the histogram. λ is always a rational number, n/N_{bin}!

Binary split 2-dim. example



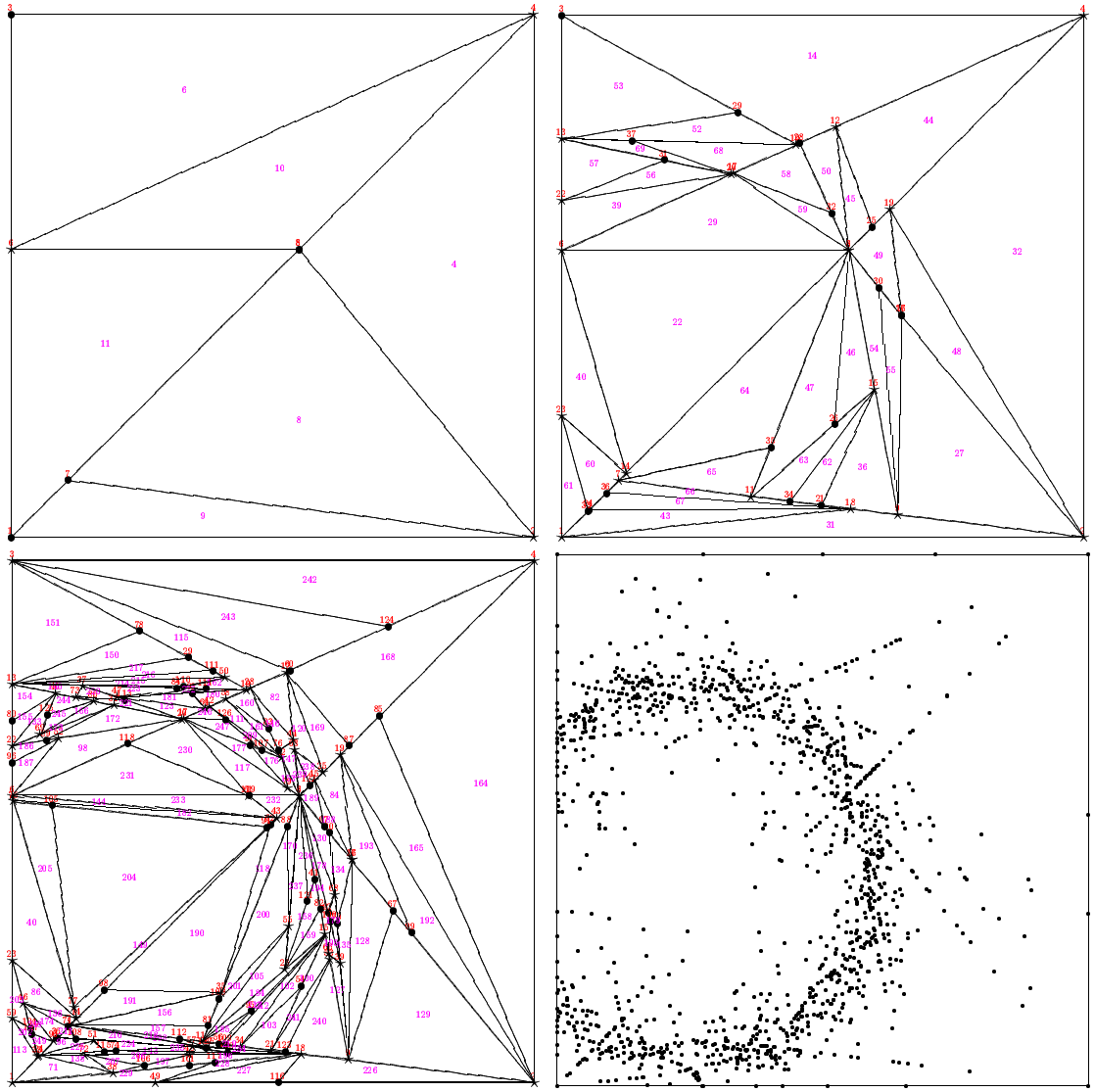
- **Integrand covers narrow strip along edges.**
- **We intend to split upper triangle.**
- **1000 w-ted events are generated and projected onto 3 sides of the parent triangle.**
- **3 Projections are analyzed.**
- **Choosen is the cell with the smallest R_{loss} (middle plot).**
- **Two resulting daughter triangles are shown at leftmost plot.**

2-dimensional example of binary split: projection on one of edges



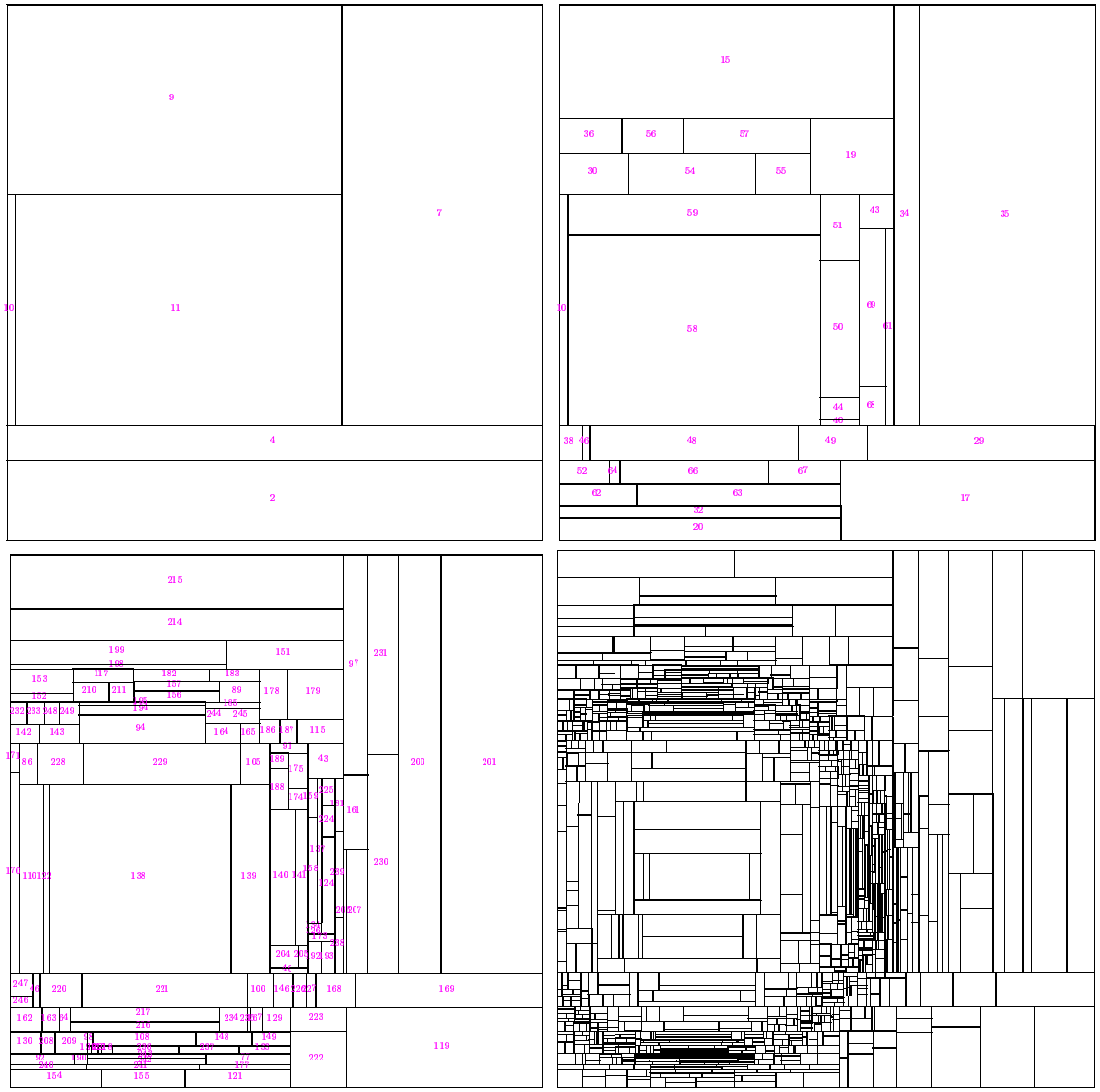
- Projected integrand $\rho(x)$.
- Old ρ' for parent cell (majorizing $\rho(x)$).
- New ρ' for two daughter cells.
- OLD R_{loss} all area above red line, for the parent cell.
- New R_{loss} between red line and New ρ', for 2 daughters.
- Obviously $I'_{New} < I'_{Old}$, the division point $\star$ is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

Evolution of simplical foam at 2-dim.



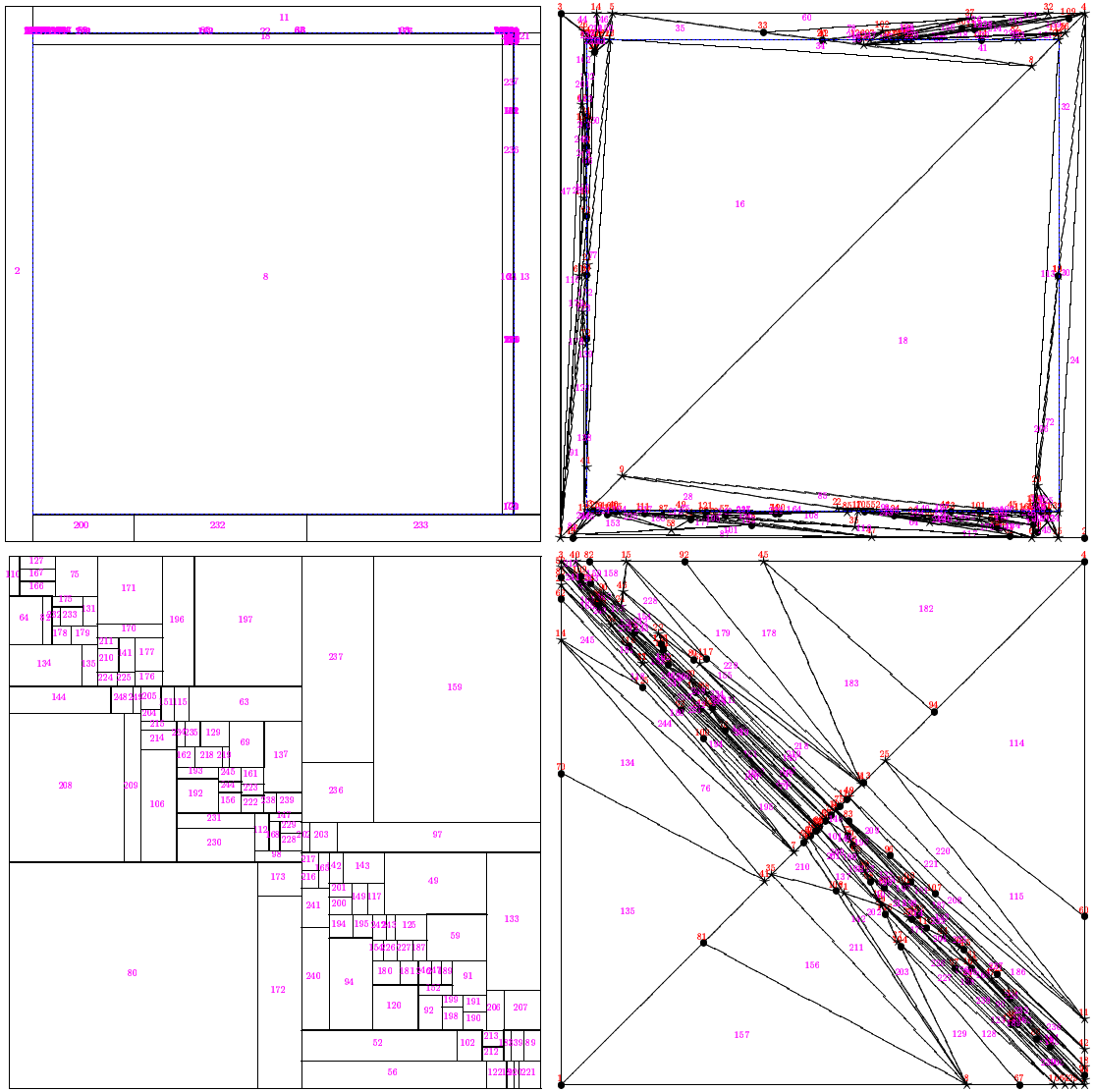
Number of cells= 10, 70, 250, 2500.

Evolution of hyper-cubic. foam at 2-dim.



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

Void and Diagonal at 2-dim.



Number of cells= 250.

Hypercubes or simplices?

- **Simplices are limited to low dimensions $n < 6$ because of $n!$, and because of many determinants heavily consuming CPU time (unless the integration domain is simplex itself).**
- **For simplices memory consumption is $\sim$ 16n Bytes/Cell, this is a serious limitation.**
- **For hypercubes memory consumption is below 50Bytes/Cell independently of n. This is really great! How it is done? See http://jadach.home.cern.ch/jadach/Krakow2001_generators_jadach.ps.gz**
- **Experience with various about 10 testing functions shown that in most cases hypercubes provide better final MC efficiency.**

CPU barrier: one quick fix is found

Final MC efficiency is improved essentially by the increasing No. of cells N_c.

CPU time of exploration $T \sim n \times N_c \times N_{samp}$ where N_{samp} is the number of MC events used in exploration of each newly created Cell.

Can we limit N_{samp} somehow in order to increase N_c?

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

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

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 nDim+kDim. Both simplices and hypercubes available. Includes hypercubical algorithm MCell with low memory consumtion as default option.

Foam v2.02, c++: using ROOT package (R. Brun et.el.) Towards persistency!

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:

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

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

$$\gamma = 0.02,\ R = 0.35$$

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

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 $< W > / W^{\varepsilon}_{\max}$ with ε=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 limiting value!

Tests of Foam at higher dimensions

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

n	N_{Cell}	$\frac{N_{MC}}{Cell}$	Effic.	$\frac{\sigma}{<W>}$	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

- FOAM is a versatile adaptive general purpose Monte Carlo similator.
- It is based on the cellular division of the integration domain.
- Geometry of the "foam of cells" is rather simple, following the rule of a binary split (but memory-efficient coding of vertices is found).
- The rules for picking up next cell for the division and division geometry starts to be relatively sophisticated (projection on edges etc.)
- FOAM can deal with peaked distribution up to 10 dimensions, with todays computers.
- Latest version not yet published, available from the author.
- Due to lack of time/space I was not able to refer to similar/related works of T. Ohl, S. Kavabata and F. Tkachov.