FOAM: General purpose Monte Carlo Cellular Algorithm

S. Jadach

Institute of Nuclear Physics, Kraków (Cracow), Poland

Outline:

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

Work supported in part by the European Community's Human Potential Programme under contract HPRN-CT-2000-00149 "Physics at Colliders", and NATO grant PST.CLG.977751.

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

MC integration $\neq$ MC generation $\neq$ MC simulation

MY AIM $\longrightarrow$ Monte Carlo Simulation (w=1 events); MC integral is a byproduct

General purpose means:

- Applies to arbitrary (wide range) user-distributions
- Works as a BLACK BOX component or stand-alone

"Adaptive" is almost a synonym of "general purpose".

e-Print: physics/0203033 describes on 60 pages:

- the entire algorithm, all variants/options
- program architecture,
- how to use the program
- and results of the numerical tests.

Here I cover mainly the algorithm.

Programs available from the author, to appear soon in Comp. Phys. Commun.

Foam v2.02, C++ :

- Fully object oriented, dynamic memory allocation, but no persistency,
- Unlimited number of cells N_c and dimension nDim+kDim,
- Simplices, hyperrectangles and Cartesian products of them as cells,
- For h-rectangles very economic memory consumption,
- Provisions for “multibranching”, automatic adjustment (similar to Kleiss$\&$Pittau)
- Possibility to start Foam from *single* simplex instead of unit hyper-cube.

Foam v2.02, C++: variant using ROOT package (R. Brun et.el.)

Implements persistency of the class TFOAM using automatic streamers of ROOT!

MCell v2.02, F77: Up to 1Mega cells is specialized for higher dimensions, $n \leq 20$, hyperrectangular cells only. (Both f77 versions are frozen!)

Foam v2.02, F77 : Up to 10K cells. Simplices ($n \leq 5$) and hyperrectangles ($n \leq 10$) available. Low memory consumption optionally for h-rects.

Works related to General Purpose MC simulation (integration) problem

VEGAS and the like

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

Cellular:

- Earlier unpublished trials in 80's by S. Kawabata, R. Kleiss and S. Jadach.
- 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).
- Keijiro Tobimatsu and Setsuya Kawabata, "Multi-dimensional Integration Routine DICE for Vector Processor" Research report of Kogakuin University, 1996, No. 85.
- E. de Doncker and A. Gupta, "Multivariate Integration on Hypercubic and Mesh Networks", Parallel Computing 24 (1998) 1223.
- S. Jadach, Comput.Phys.Commun. **130**, 244 (2000);
  and "Foam: A general purpose cellular Monte Carlo event generator" e-Print: physics/0203033.

General features of General Purpose Monte Carlo Simulators (GPMCS)

Inevitably the GPMCS works 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.

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 and driving the algorithm of exploration.)

Cellular exploration of the distribution

One can minimize the variance (or maximum weight) of the target weight distribution in MC generation by splitting the integration domain into many small cells, such that $\rho(\vec{x})$ is approximated by constant $\rho'(\vec{x})$ within each cell. This is a cellular class of GPMCSs.

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

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

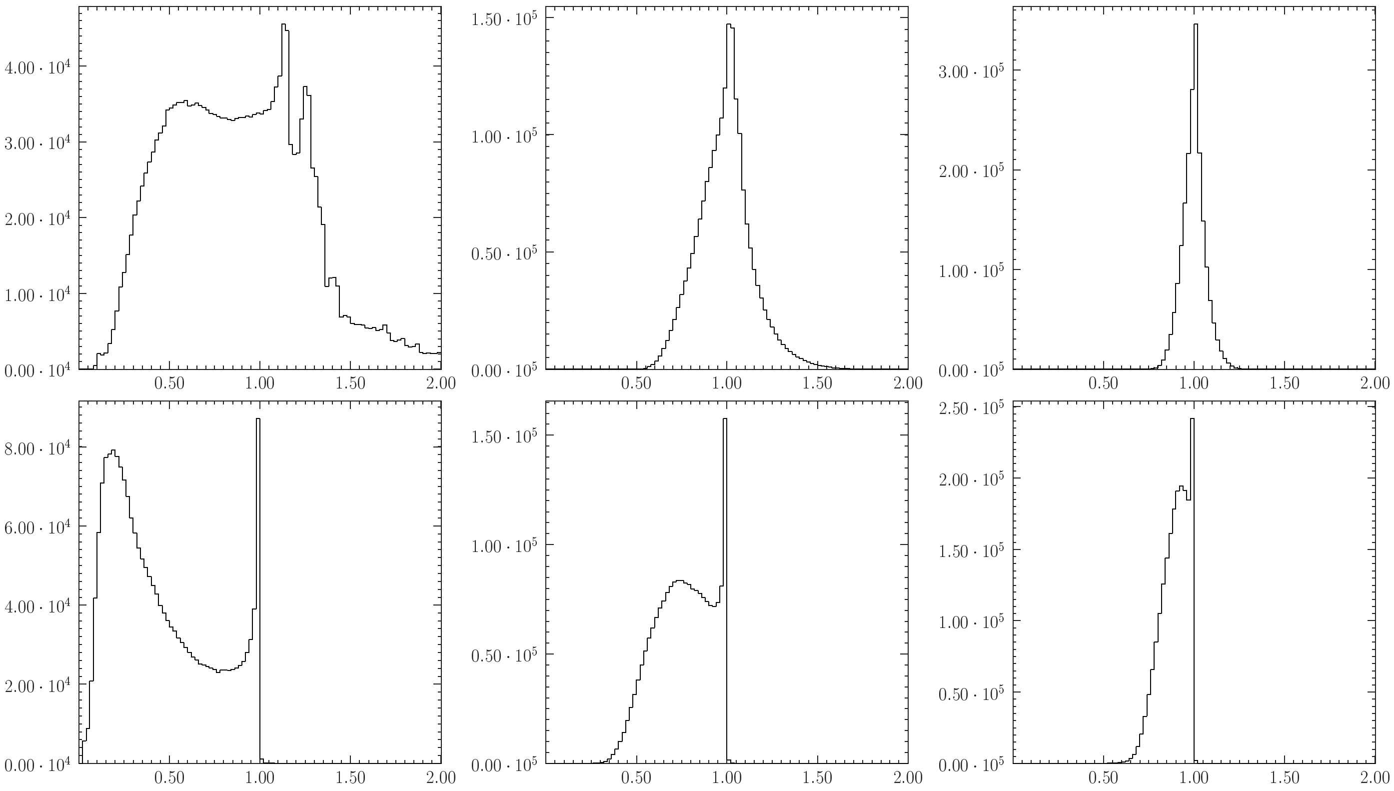
The system of cells can be created all at once (like in Vegas) or in a more evolutionary way, by the “split process”. In the Foam algorithm we rely on the binary split of cells. (Choice of a cell to be split driven by target weight distribution.)

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

Variance reduction versus maximum weight reduction

Foam can minimize the ratio of the maximum weight to the average weight $w_{\max}/\langle w \rangle$ (our main maim) or variance $\sigma = \sqrt{\langle w^2 \rangle - \langle w \rangle^2}$.

Evolution of the weight distribution for increasing No. of cells: 200, 2k, 20k



Two auxiliary distributions $\rho'(x)$ and $\rho_{loss}(x)$

Both evolve step-by-step in the process of binary split of cells in the exploration process. $\int \rho_{loss}(x)$ measures "MC inefficiency".
Events generated according to $\rho'(x)$ in exploration and generation.
$R' = \int \rho' d^n x$ known exactly! $R = R'\langle w \rangle'$, where $w = \rho/\rho'$.

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

$\rho'(x) \equiv \max_{y \in Cell_i} \rho(y)$, for $x \in Cell_i$, the "ceiling function".
$R_{loss} = \int d^n x\ [\rho'(x) - \rho(x)] = \int d^n x\ \rho_{loss}(x)$.
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 $\langle \dots \rangle_i$ is over $i-th$ Cell.
$\rho_{loss}(x) \equiv \sqrt{\langle \rho^2 \rangle_i} - \langle \rho \rangle_i$, for $x \in Cell_i$.
Final MC variance is just $\simeq R_{loss}$.

Rules governing binary split of a cell

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

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

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

[1] We always split "the worst" cell with the biggest R_{loss} (or with prob. $\sim R_{loss}$)

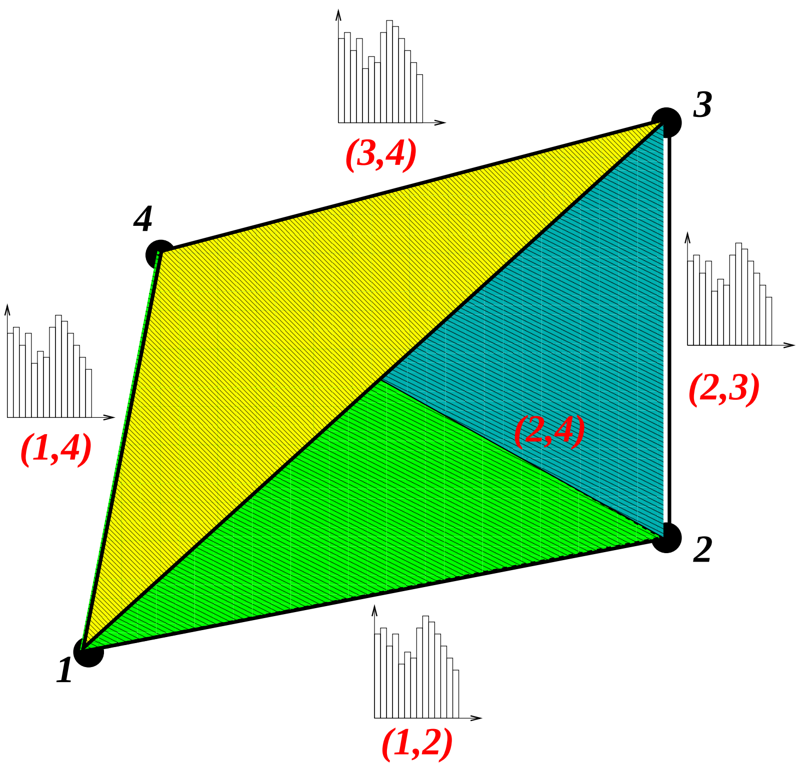
[2] Position/direction of a plane dividing a parent cell into two daughter cells is chosen to get the smallest total R_{loss}.

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-rect. 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...

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

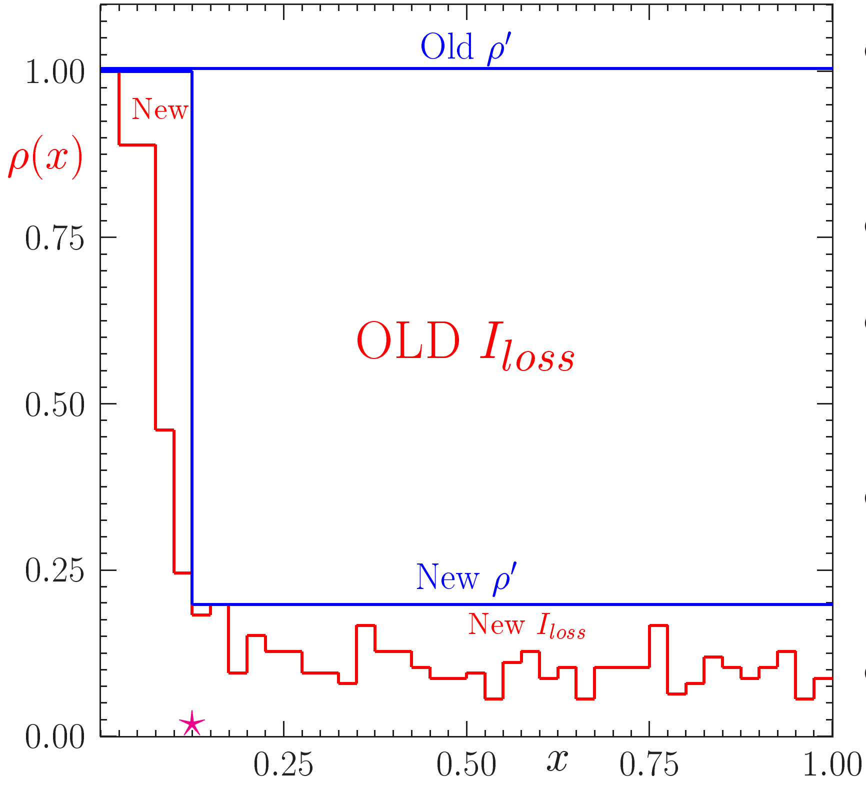


We select the best (i,j) out of $\frac{n(n-1)}{2}$ possibilities:

- For each (i,j) the $dw/d\lambda$ histogrammed, its R_{loss} is estimated.
- The edge (i,j) with the biggest R_{loss} is selected!
- For the cell division λ is read from the histogram.

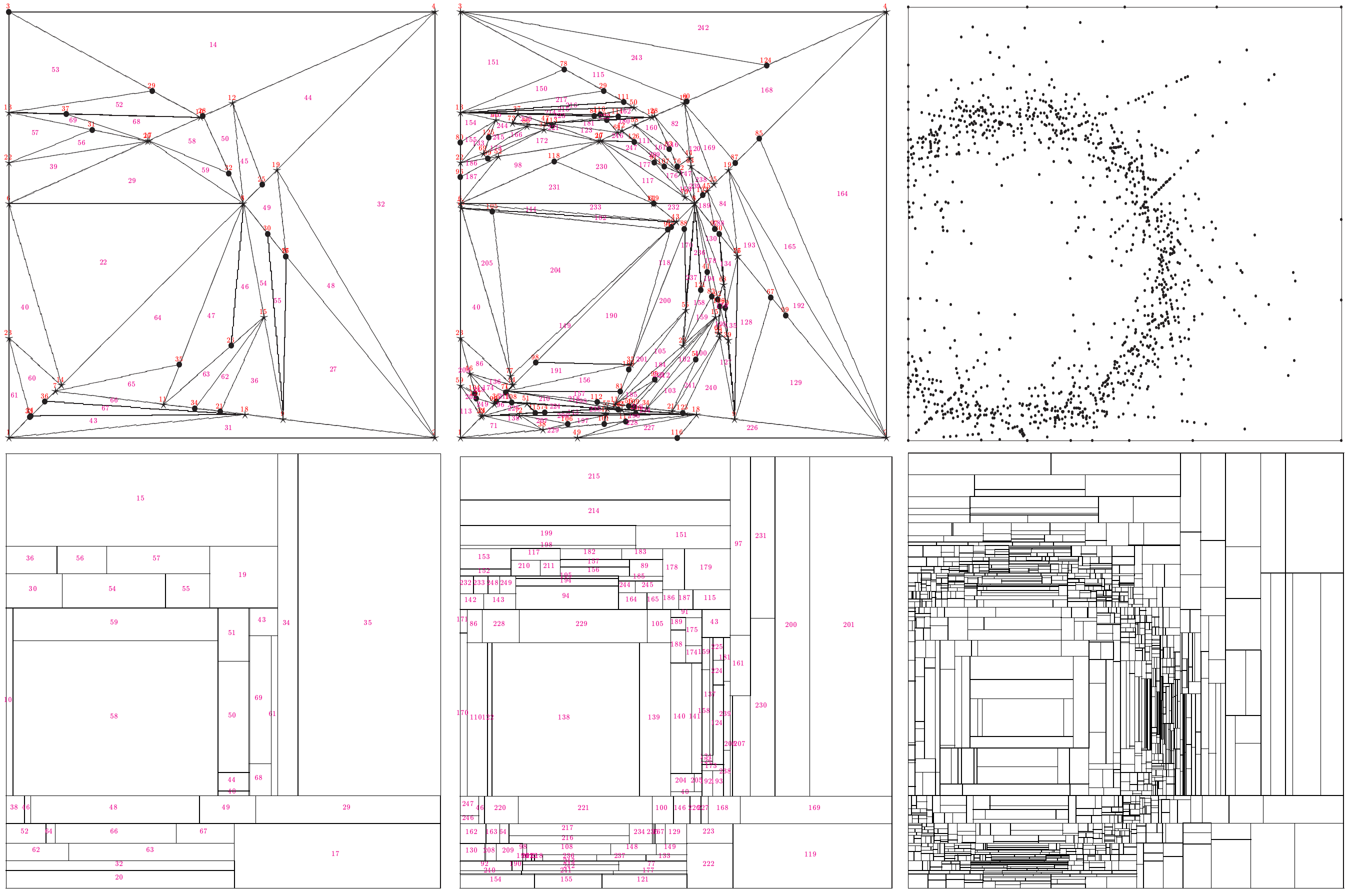
NB. λ is always a rational number, n/N_{bin}. The key for economic memory storage.

The best choice of the division plane position from the “edge histogram”



- 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 and hyper-rect. foam at 2-dim.



Number of cells= 70, 250, 2500.

Hyperrectangles or simplices?

- **Simplices are limited to low dimensions $n < 6$ because of $n!$, and because of many determinants heavily consuming CPU time (for integration domain being simplex it is slightly better).**
- **For simplices memory consumption is $\sim$ 16n Bytes/Cell, this is a serious limitation.**
- **For hyperrectangles memory consumption is $\sim$50Bytes/Cell independently of n. This is really great! How it is done? See e-Print: physics/0203033.**
- **Experience with many testing distributions has shown that quite often hyperrectangles provide comparable or even better final MC efficiency than simplices.**

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.

Tests of Foam at low dimensions

Functions at 2-dimens.	Foam 1.01	Simpl.	H-Rect.	VEGAS
$\rho_a(x)$ (diagonal ridge)	0.93	0.93	0.86	0.03
$\rho_b(x)$ (circular ridge)	0.82	0.82	0.82	0.16
$\rho_c(x)$ (edge of square)	0.57	1.00	1.00	0.53

Functions at 3-dimens.	Foam 1.01	Simpl.	H-Rect.	VEGAS
$\rho_a(x)$ (thin diagonal)	0.67	0.74	0.66	0.002
$\rho_b(x)$ (thin sphere)	0.36	0.47	0.53	0.11
$\rho_c(x)$ (surface of cube)	0.37	0.95	1.00	0.30

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.

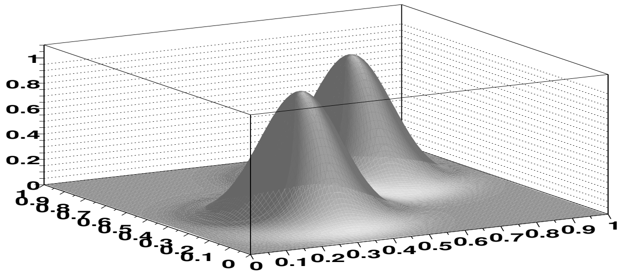
Keep in mind that efficiency of Foam $\langle w \rangle / w_{\max}$ can be in the above example easily increased to almost 100% by increasing no of cells, while for VEGAS we are here at the limiting value!

Tests of Foam at higher dimensions, Camel test-function of P. Lepage, normalized to one

nDim	kDim	nCalls	nCells	nSampl	$w^{\varepsilon}_{\max}/\langle w\rangle$	$\sigma/\langle w\rangle$	$\Delta_{statist.}R$	R
0	1	206192	1000	1000	0.99147	0.014752	1.043e-05	0.99999962
0	1	206192	1000	10000	0.99147	0.014752	1.043e-05	0.99999962
0	3	435112	1000	1000	0.50886	0.54033	0.000382	1.00017104
0	3	834094	1000	10000	0.50359	0.54674	0.000386	1.00056316
0	3	1015157	1000	33333	0.51091	0.54035	0.000382	0.99983999
0	3	2675820	10000	1000	0.72677	0.27504	0.000194	0.99995080
0	3	3333479	10000	10000	0.72200	0.27720	0.000196	1.00008994
0	3	3575366	10000	33333	0.72243	0.27786	0.000196	0.99997875
0	4	3825046	10000	1000	0.50363	0.51168	0.000361	1.00013082
0	4	6559430	10000	10000	0.50297	0.51001	0.000360	0.99960319
2	2	4493961	10000	1000	0.43076	0.63185	0.000446	1.00072564
2	2	9374351	10000	10000	0.44922	0.60669	0.000429	1.00013171
4	0	6642202	10000	1000	0.21029	1.19420	0.000844	1.00072248
4	0	12337748	10000	10000	0.20817	1.20067	0.000849	1.00020405
0	6	2311881	1000	3333	0.04199	2.12091	0.001499	0.99856206
0	6	12844256	1000	33333	0.03279	2.61028	0.001845	0.99799089
0	6	12737314	10000	3333	0.15385	1.15211	0.000814	1.00039754
0	6	42827237	10000	33333	0.14168	1.22627	0.000867	0.99954178
0	6	42808972	100000	1000	0.30910	0.71250	0.000503	0.99972833
0	6	92531875	100000	10000	0.30905	0.71423	0.000505	0.99985093
0	9	78325890	100000	1000	0.03718	1.64608	0.001163	0.99367339
0	9	353943409	100000	10000	0.05196	1.80538	0.001276	1.00196909
0	9	272162624	400000	1000	0.08490	1.30193	0.000920	1.00065580
0	9	924011087	400000	10000	0.08853	1.38579	0.000979	1.00052122
0	12	261911066	100000	3333	0	5.83954	0.004129	0.97304842
0	12	671460574	100000	10000	0.00640	3.85823	0.002728	0.98878698
0	12	913072065	400000	3333	0.01285	2.73991	0.001937	0.98688299
0	12	2117963809	400000	10000	0.01235	2.92642	0.002069	0.99301117

Efficiency depends mainly on number of cells `nCells`

Number of MC trials per Cell cannot be too small.



Conclusions

- FOAM is a versatile adaptive general purpose Monte Carlo simulator.
- 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 the 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 2.02 in C++ and f77 to appear in in Comp.Phys.Commun., available from the author.
- First “real-life” applications: ISR and beamstrahlung in KKMC. Looking for more applications.