Adaptacyjne techniki całkowania Monte Carlo

- **Wady klasycznych metod MC redukcji wariancji.**
- **Typowy schemat adaptacyjnego algorytmu Monte Carlo.**
- **Przykłady adaptacyjnych algorytmów Monte Carlo:**
  - ▷ **Program** RIWIAD **Sheppeya i Lautrupa.**
  - ▷ **Adaptacyjny algorytm średniej ważonej Friedmana.**
  - ▷ **Program** DIVONNE2 **Friedmana.**
  - ▷ **Program** VEGAS **Lepage'a.**
  - ▷ **Program** FOAM **Jadacha.**
- **Metody całkowania Monte Carlo a kwadratury numeryczne.**

$\Rightarrow$ http://th-www.if.uj.edu.pl/~placzek/dydaktyka/MMC/

Wady klasycznych technik MC redukcji wariancji

- Wszystkie przedstawione klasyczne metody redukcji wariancji, z wyjątkiem równomiernego losowania warstwowego, **wymagają** pewnej **uprzedniej znajomości funkcji**.
- Kiedy metody te zostaną **niewłaściwie zastosowane** mogą łatwo prowadzić do **degradacji efektywności** rachunku Monte Carlo zamiast do jego poprawy!
- Potrzebny jest dodatkowy nakład pracy związany z zastosowaniem danej metody redukcji wariancji do każdej całkowanej funkcji.

▶ Naturalne rozszerzenie metod redukcji wariancji idzie w kierunku **technik adaptacyjnych**, czyli takich, które **„poznają" funkcję w trakcie wykonywania rachunku** (najlepiej nie wymagając żadnej znajomości funkcji *a priori*) i odpowiednio dobierają metodę liczenia całki.

▷ Podobnie inspirowane techniki znajdują szerokie zastosowanie przy obliczaniu całek przy użyciu kwadratur numerycznych.

▷ Prawdziwie adaptacyjne metody dla całkowania Monte Carlo są mniej powszechne, ponieważ są dość trudne do zrealizowania – najczęściej występują w postaci gotowych **programów komputerowych** (poznamy kilka przykładów).

⟶ **Integrator MC** – program przenaczony tylko do całkowania funkcji.

⟶ **Generator MC** – program, który, oprócz całkowania, może wydajnie generować przypadki (punkty losowe) według rozkładów danych przez całkowane funkcje.

Typowy schemat adaptacyjnego algorytmu Monte Carlo

Podział na dwie fazy:

1. **Faza eksploracyjna** – ,,poznawanie" funkcji.
   - Najczęściej rekursywny podział obszaru całkowania na warstwy (stratyfikacja), tak aby całki w poszczególnych warstwach były w przybliżeniu jednakowe, a funkcja w obrębie wartwy była jak najbliższa stałej (niełatwe zadanie!) – do oszacowania całek w warstwach używa się albo prostej metody kwadraturowej, albo metody podstawowej Monte Carlo.
   - Czasami funkcję podcałkową aproksymuje się pewną kombinacją wybranych funkcji elementarnych.
2. **Faza obliczeniowa** – obliczanie wartości całki i jej błędu statystycznego.
   - Obliczana jest całka metodą **losowania warstwowego** lub metodą **średniej ważonej** – zależnie od fazy eksploracyjnej.
   - Czasami istnieje możliwość generowania punktów według rozkładu danego przez funkcję podcałkową.

▷ Czasem faza obliczeniowa połączona jest z fazą eksploracyjną i dla każdego podziału na warstwy obliczana jest całka, a następnie konstruowana jest tzw. **biegnąca średnia ważona** estymatora całki oraz jej błędu – procedura jest **zatrzymywana** po osiągnięciu **żądanej dokładności**.

⟶ Może być obarczona błędami systematycznymi z wczesnych etapów rachunowych, od których nie potrafi się uwolnić! (np. funkcja z wąskim pikiem ,,nieodkrytym" na wczesnych etapach)

Przykłady adaptacyjnych algorytmów MC: RIWIAD

▶ **Program RIWIAD Sheppeya i Lautrupa (ok. 1970) :** jeden z najwcześniejszych adaptacyjnych integratorów MC – używany do całkowania funkcji wielowymiarowych na hiperkostce $(0,1)^n$.

▷ **Schemat działania:**

- Na początku dzieli hiperkostkę na jednakowe hiperkostki i w każdej z nich oszacowuje całkę metodą podstawową MC (jednorodna stratyfikacja).
- W oparciu o wartości całek w podobszarach przesuwa ich granice tak, że objętości nowopowstałych hiperprostokątów są mniejsze tam, gdzie funkcja jest większa i odwrotnie.
- Powyższy proces jest kontynuowany i w każdym kroku obliczany jest estymator całki oraz jego wariancja – z nich konstruowana jest **biegnąca średnia ważona** estymatora całki funkcji oraz jego odchylenia standardowego.
- Procedura jest zatrzymywana po osiągnięciu wymaganej dokładności.

▷ **Wady:**

∗ Granice hiperprostąkątów są zawsze **równoległe** do pierwotnych osi parametrów i zawsze biegną wzdłuż **całej długości hiperkostki**, nawet jeśli poprzednie wyniki wskazywały, że pewne hiperprostokąty nie musiałyby być dzielone.

∗ Ważona średnia cząstkowych wyników może być obarczona **błędami systematycznymi** spowodowanymi **korelacjami** między estymatorami **wartości oczekiwanej** oraz **wariancji** – program nigdy nie wyzwala się z przypadkowo złych oszacowań w poprzednich krokach.

Przykłady: Adaptacyjny algorytm średniej ważonej Friedmana

▶ **Program J. Friedmana (lata 1970-te) (nieopublikowany):** przykład adaptacyjnego algorytmu MC korzystającego z metody średniej ważonej.

▷ **Schemat działania:**

1. Faza eksploracyjna

   Konstruowana jest **funkcja próbna** jako kombinacja liniowa funkcji Cauchy'ego (Breita–Wignera), w której pozycje i kształty pików odpowiadają odpowiednim pikom funkcji całkowanej. Do skonstruoawania funkcji próbnej używana jest procedura minimalizacji funkcji oraz analiza wektorów własnych kowariancji funkcji wokół każdego piku.

   ▷ Funkcja Cauchy'ego jest używana dlatego, że dąży do zera wolniej niż funkcja Gaussa, pozwalając unikać niestablilności.

2. Faza obliczeniowa

   Obliczana jest wartość całki i jej błąd statystyczny przy użyciu metody **średniej ważonej**, w oparciu o **funkcję próbną** skonstruowaną w **fazie ekploracyjnej**.

▷ **Wady:**

Nie nadaje się dla funkcji, które nie mogą być aproksymowane niewielką liczbą pików Cauchy'ego – niemało jest takich funkcji w praktyce!

Przykłady adaptacyjnych algorytmów MC: DIVONNE2

▶ **Program DIVONNE2, J. Friedman, SLAC CGTM No. 188 (1977): adaptacyjny algorytm MC oparty o rekursywny podział obszaru całkowania** (dostępny w bibliotece CERNLIB).

▷ **Schemat działania:**

1. Faza eksploracyjna

   Wykonywanie rekursywnego podziału wielowymiarowego obszaru całkowania (stratyfikacja), tak aby o trzymać podobszary, w których zakres wartości całkowanej funkcji jest jak najmniejszy. W tym celu używane są techniki minimalizacji funkcji.

2. Faza obliczeniowa

   Obliczanie wartości całki i jej błędu statystycznego w oparciu o metodę **losowania warstwowego**.

   ▷ Istnieje także możliwość generowania punktów losowych według wielowymiarowego rozkładu danego przez funkcję podcałkową.

▷ **Wady:**

Granice podziałów są równoległe do osi parametrów funkcji, podobnie jak w RIWIADzie.

⟶ Ale ponieważ podział jest rekursywny (tylko jeden podobszar jest dzielony w każdym kroku, a nie cały rząd), to algorytm wykazuje tendencję do uwalniania się od orientacji osi.

▶ **Program VEGAS, J. G. P. Lepage, J. Comp. Phys. 27 (1978) 192: adaptacyjny algorytm MC oparty o iteracyjny podział obszaru całkowania** (podobny do RIWIADa).

Algorytm jednowymiarowy

▷ Niech: $I = \int_0^1 f(x)dx$.

1. Generujemy M losowych punktów $x_i \in \mathcal{U}(0,1)$, tzn. z jednorodną gęstością prawdopodobieństwa $p(x) = 1$, i obliczamy w nich wartości funkcji: $f(x_i)$. Obliczamy całkę i jej błąd metodą podstawową MC.
2. Przedział całkowania dzielimy na N równych przedziałów:

$$0 = x_0 < x_1 < \ldots < x_N = 1,\ \Delta x_i = x_i - x_{i-1}.$$

Następnie każdy z przedziałow Δx_i dzielimy na $(m_i + 1)$ podprzedziałów, gdzie:

$$m_i = K\,\frac{\bar{f}_i \Delta x_i}{\sum_j \bar{f}_j \Delta x_j}\,, \qquad K = \text{const. (typowo } = 1000),$$

a

$$\bar{f}_i \equiv \sum_{x \in [x_{i-1}, x_i)} |f(x)| \quad \sim \quad \frac{1}{\Delta x_i} \int_{x_{i-1}}^{x_i} |f(x)|\, dx\,.$$

$\Rightarrow$ Nowy podział będzie gęstszy tam gdzie funkcja $|f(x)|$ jest ,,większa", a rzadszy tam, gdzie $|f(x)|$ jest ,,mniejsza".

3. Przywracamy początkową liczbę N podziałów przez sklejanie jednakowych ilości kolejnych podprzedziałów w większe przedziały.

   $\Rightarrow$ Nowe przedziały będą **węższe** tam gdzie funkcja $|f(x)|$ jest **większa** i odwrotnie.

   Generujemy M punktów według schodkowej gęstości prawdopodobieństwa:

$$p(x) = \frac{1}{N\Delta x_i}, \quad x_{i-1} \le x < x_i,\ i = 1, \ldots, N,$$

   i obliczamy całkę oraz jej błąd metodą średniej ważonej.

4. Powtarzamy powyższe iteracje do momentu, aż znajdziemy optymalny podział, tzn.

$$m_i \approx m_j,\ i, j = 1, \ldots, N.$$

   Po każdej iteracji obliczamy biegnącą średnią ważoną:

$$\sum_k \frac{I_k}{\sigma_k^2},$$

   gdzie I_k, σ_k – estymator całki i jego odchylenie standardowe w k-tej iteracji.

5. Po zakończeniu wszystkich iteracji obliczamy całkę i jej błąd statystyczny według wzorów:

$$\bar{I} = \sigma_{\bar{I}}^2 \sum_k \frac{I_k}{\sigma_k^2}, \qquad \sigma_{\bar{I}} = \left[\sum_k \frac{1}{\sigma_k^2}\right]^{-\frac{1}{2}}.$$

▶ **Dalsze praktyczne ulepszenia:**

- Aby uniknąć szybkich, destabilizujących zmian podziału od iteracji do iteracji, liczba podziałów każdego przedziału dobierana jest według:

$$m_i = K \left\{ \left[\frac{\bar{f}_i \Delta x_i}{\sum_j \bar{f}_j \Delta x_j} - 1 \right] \frac{1}{\log[\bar{f}_i \Delta x_i / \sum_j \bar{f}_j \Delta x_j]} \right\}^{\alpha},$$

gdzie α determinuje stopień zbieżności i typowo jest ustawiane między 1 a 2.

- Kiedy funkcja ma wąskie piki, I_k oraz σ_k mogą być źle oszacowane we wczesnych iteracjach, co może dawać błędy systematyczne ostatecznego wyniku. Aby to poprawić, poprzedni estymator całki zastępowany jest następującym:

$$\bar{I} = \left[\sum_k \frac{I_k^2}{\sigma_k^2} \right]^{-1} \sum_k I_k \left(\frac{I_k^2}{\sigma_k^2} \right), \qquad \sigma_{\bar{I}} = \bar{I} \left[\sum_k \frac{I_k^2}{\sigma_k^2} \right]^{-\frac{1}{2}}.$$

- Liczba iteracji (minus jeden) nie powinna być znacznie mniejsza od (odpowiednio):

$$\chi^2 \simeq \sum_k \frac{(I_k - \bar{I})^2}{\sigma_k^2} \quad \text{lub} \quad \sum_k \frac{(I_k - \bar{I})^2}{\bar{I}^2} \frac{I_k^2}{\sigma_k^2}.$$

Jeżeli tak jest, to algorytmowi nie można ufać!

$\Rightarrow$ Ćw. A2*: Pokazać, że powyższe estymatory $\bar{I}$ są nieobciążone, a ich wariancje wynoszą $\sigma_{\bar{I}}^2$.

Algorytm wielowymiarowy

▷ Weźmy dla ilustracji: $I = \int_0^1 dx \int_0^1 dy\, f(x,y).$

Próbna gęstość prawdopodobieństwa jest brana w postaci:

$$p(x,y) = p_x(x)\, p_y(y)\,.$$

- Można pokazać (np. posługując się metodami analizy funkcjonalnej i używając mnożników Lagrange'a), że optymalne gęstości prawdopodobieństwa mają postać:

$$p_x(x) = \frac{\sqrt{\int_0^1 dy \frac{f^2(x,y)}{p_y(y)}}}{\int_0^1 dx \sqrt{\int_0^1 dy \frac{f^2(x,y)}{p_y(y)}}}\,,$$

wzór na $p_y(y)$ ma analogiczną postać.

- Zatem algorytm jednowymiarowy może być zastosowany wzdłuż każdej osi, z $\bar{f}_i$ zdefiniowaną dla osi x jako:

$$(\bar{f}_i)^2 = \sum_{x\in[x_{i-1},x_i)} \sum_y \frac{f^2(x,y)}{p_y^2(y)} \quad\sim\quad \frac{1}{\Delta x_i} \int_{x_{i-1}}^{x_i} dx \int_0^1 dy\, \frac{f^2(x,y)}{p_y(y)}$$

i analogicznie dla osi y.

Przykład zastosowania

- Sferycznie symetryczna funkcja Gaussa:

$$I_n = \left(\frac{1}{a\sqrt{\pi}}\right)^n \int_0^1 d^n x \, \exp\left[-\sum_{i=1}^{n} \frac{(x_i - 0.5)^2}{a^2}\right] \qquad = 1\,,$$

gdzie $n = 9,\ a = 0.1$, a stopień zbieżności $\alpha = 1$.

VEGAS

Iteracja	I_k	σ_k	$\bar{I}$	$\sigma(\bar{I})$	Liczba obliczeń funkcji
1	0.007	0.005	0.007	0.005	10^4
3	0.643	0.070	0.612	0.064	$3 \cdot 10^4$
5	1.009	0.041	0.963	0.034	$5 \cdot 10^4$
10	1.003	0.008	1.001	0.005	10^5
Metoda podstawowa MC			0.843	0.360	10^5

Porównanie z kwadraturami numerycznymi

- Sferycznie symetryczna funkcja Gaussa:

$$I_n = \left(\frac{1}{a\sqrt{\pi}}\right)^n \int_0^1 d^n x \, \exp\left[-\sum_{i=1}^{n} \frac{(x_i - 0.5)^2}{a^2}\right] \qquad = 1\,,$$

gdzie $n = 9,\ a = 0.1$.

Numeryczne kwadratury Gaussa–Legendre'a

Liczba punków na oś	Wartość całki	Liczba obliczeń funkcji
5	71.364	$2 \cdot 10^6$
6	0.017	10^7
10	0.774	10^9
15	1.002	$3.8 \cdot 10^9$

Przykłady adaptacyjnych algorytmów MC: VEGAS

▶ **Wady algorytmu VEGAS:**

- Podział obszaru całkowania odbywa się wzdłuż osi parametrów, co zakłada przybliżoną faktoryzację funkcji całkowanej na funkcje jednowymiarowe – dla niektórych funkcji wielowymiarowych może to nie być dobre przybliżenie!
- Nie radzi sobie z funkcjami, które mają piki lub krawędzie ukośnie do osi parametrów.
- Ponieważ jest optymalizowany na minimum wariancji estymatora całki, więc nie jest zbyt wydajny do generowania punktów losowych według rozkładu danego przez całkowaną funkcję (długie ogony wagi), czyli jest raczej **integratorem** niż generatorem MC.

▶ **Dalsze udoskonalenia algorytmu VEGAS:**

▷ S. Kawabata, Comput. Phys. Commun. **88** (1995) 309:

Możliwość wyodrębnienia podzbioru zmiennych, dla których funkcja jest silnie zmienna i ,,wyśredniowania" po pozostałych ,,łagodnych" zmiennych $\longrightarrow$ użyteczne dla bardzo dużej liczby wymiarów ~ 100.

▷ T. Ohl, Comput. Phys. Commun. **120** (1999) 13:

Zamiast przybliżać funkcję podcałkową iloczynem n funkcji, można przybliżać ją sumą takich iloczynów z automatycznym dopasowaniem względnych wkładów poszczególnych składników.

$\Rrightarrow$ Ćw. N1* (nadobowiązkowe):

Zaimplementować algorytm VEGAS i zastosować go do obliczenia całek Φ i Ψ z wykładu 2. Porównać efektywność tego algorytmu oraz klasycznych metod całkowania Monte Carlo.

Przykłady adaptacyjnych algorytmów MC: FOAM

▶ **Program FOAM, S. Jadach,** Comp. Phys. Commun. **130** (2000) 244, arXiv:physics/9910004; Comp. Phys. Commun. **152** (2003) 55; arXiv:physics/0203033: **adaptacyjny generator MC oparty o rekursywny podział obszaru całkowania na komórki.**

▷ **Dwa algorytmy** podziału obszaru całkowania na komórki:

- **Symplektyczny:** Komórki w postaci **sympleksów** (hipertrójkątów) – ograniczony do niezbyt dużej liczby wymiarów (≤ 5).
- **Kubiczny:** Komórki w postaci **hiperprostokątów** – może być stosowany do większej liczby wymiarów (≤ 20).

▶ **Schemat $\longrightarrow$ dwie fazy:**

1. **Faza eksploracyjna:**

   Obszar całkowania w postaci **wielowymiarowej kostki** dzielony jest **rekursywnie na komórki** – **w każdym kroku tylko jedna komórka jest dzielona na dwie niekoniecznie jednakowe części**.

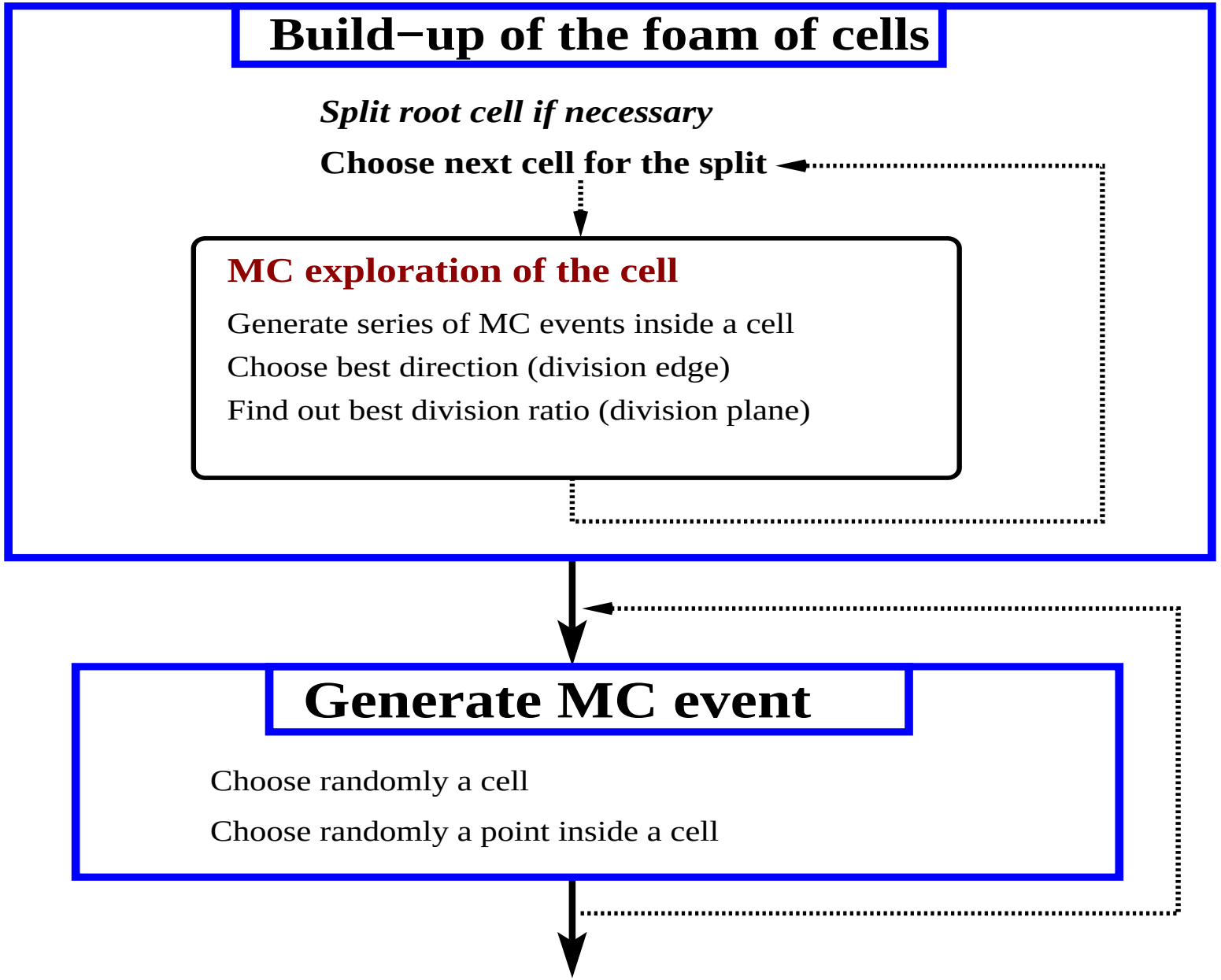
   Podział jest **zatrzymywany** po osiągnięciu ustalonej z góry **maksymalnej liczby komórek**.

   Konstruowana jest **funkcja aproksymująca** i obliczana jest **przybliżona wartość całki**.

2. **Faza generacji/obliczeń:**

   Generowane są **punkty losowe** według rozkładu danego przez **wielowymiarową funkcję** i **obliczana** jest odpowiednia **całka** metodą **średniej ważonej**, w oparciu o funkcję aproksymującą i jej całkę wyznaczoną w **fazie eksploracyjnej**.

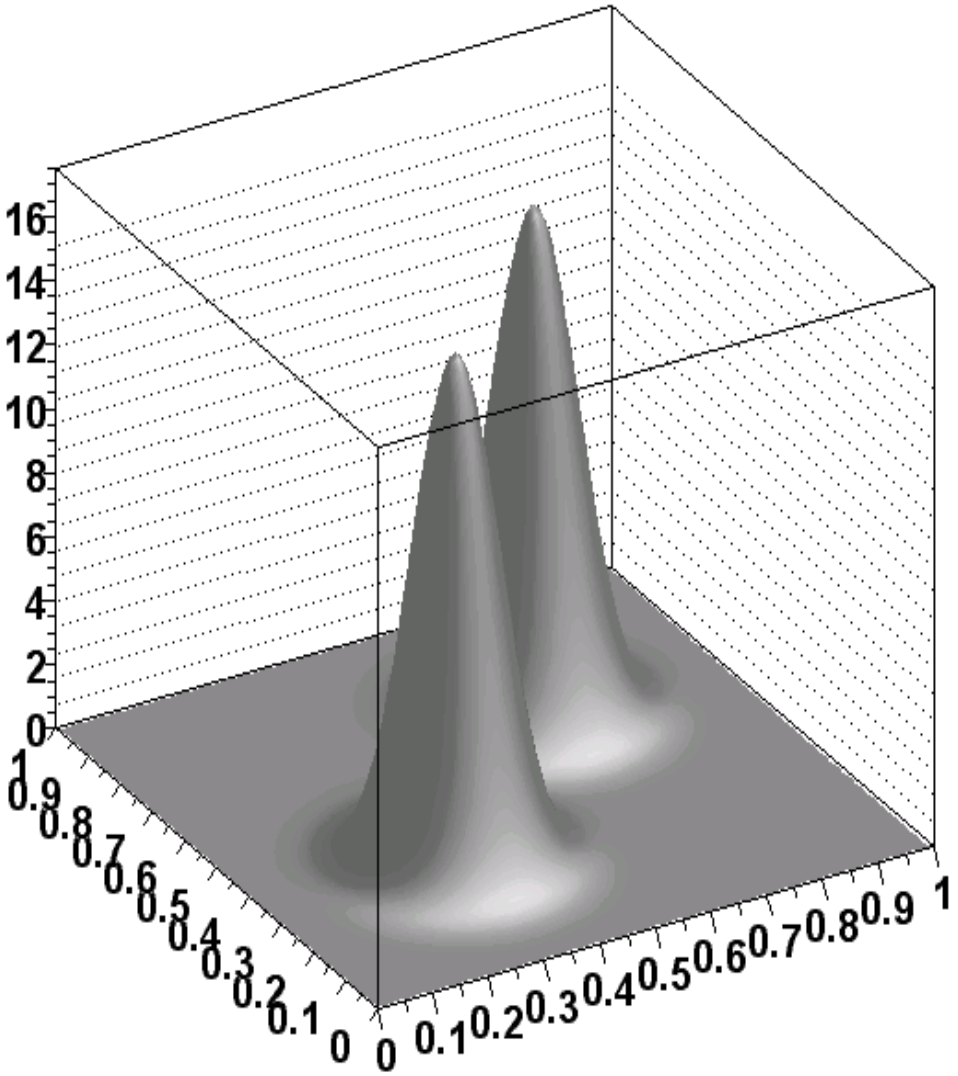
Schemat algorytmu FOAM

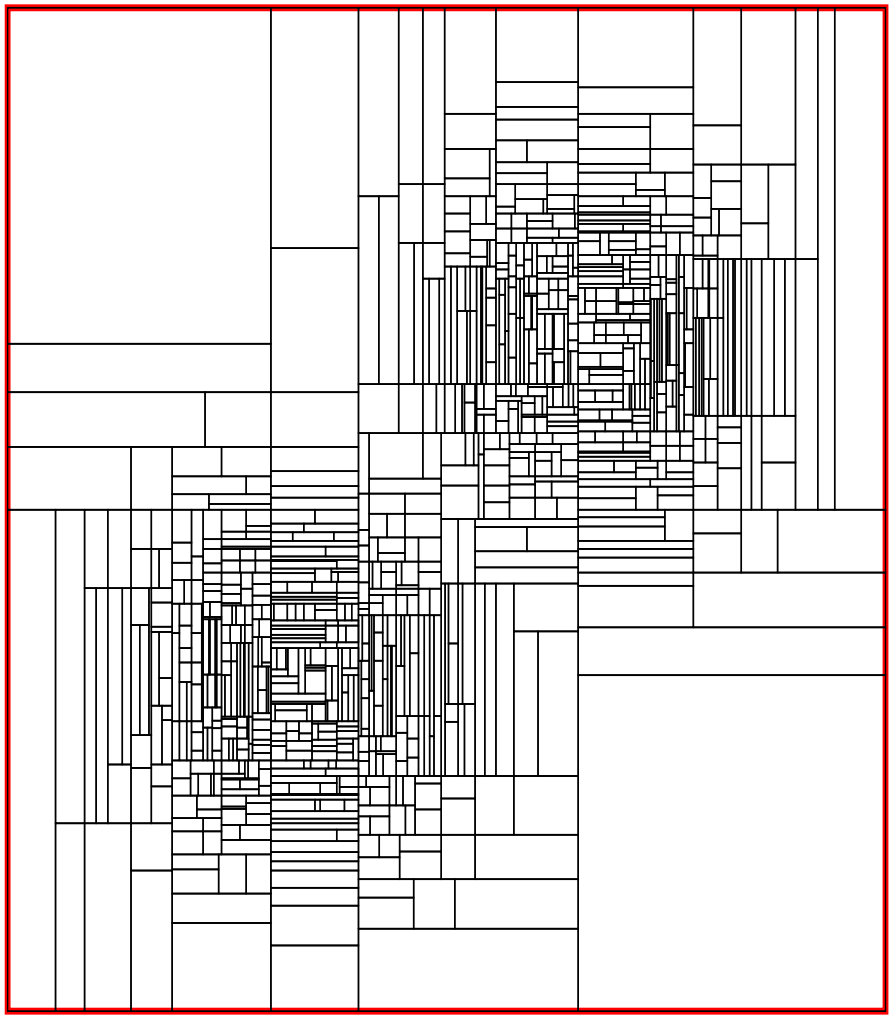


Cechy algorytmu FOAM

- Możliwe dwa kryteria podziału komórek:
  (1) Optymalizacja wariancji estymatora całki – optymalne do całkowania.
  (2) Optymalizacja wagi maksymalnej – optymalne do generowania przypadków nieważonych.
- Możliwość generowania zmiennych dyskretnych.
- Możliwość wyłączenia podziału komórek w wybranych kierunkach (dla których funkcja jest prawie stała).
- Możliwość predefiniowania punktów podziału – dla wąskich pików.
- Możliwość zastosowania wielokanałowego schematu generacji MC.
- Oszczędne wykorzystanie pamięci dla algorytmu kubicznego (do zapamiętywania wierzchołków każdej komórki używane są dwie 2-bajtowe liczby całkowite zamiast $2n$ 8-bajtowych liczb zmiennopozycyjnych).
- Program jest dostępny w dwóch językach programowania: C++ i Fortran 77.
- Tzw. wersja ,,kompakt" o nazwie mFOAM dostępna jest w systemie ROOT– szczegóły w:
  S. Jadach, P. Sawicki, Comput. Phys. Commun. **177** (2007) 441; arXiv:physics/0506084.

FOAM: przykład podziału na komórki





Trzy testowe funkcje dwu i trzech zmiennych

Wydajności generowania przypadków: $\boxed{\langle w \rangle / w^{\varepsilon}_{\max}}$ dla $\varepsilon = 0.0005$ (tzn. $\mathcal{P}(w > w^{\varepsilon}_{\max}) < \varepsilon$).

Wyniki FOAMa dla 5000 komórek i 200 przypadków MC na komórkę w fazie eksploracyjnej.

Funkcja 2-wymiarowa	FOAM-symplektyczny	FOAM-kubiczny	VEGAS
$\rho_a(x)$ (cienka diagonala)	0.93	0.86	0.03
$\rho_b(x)$ (brzeg koła)	0.82	0.82	0.16
$\rho_b(x)$ (brzeg kwadratu)	1.00	1.00	0.53
Funkcja 3-wymiarowa	**FOAM-symplektyczny**	**FOAM-kubiczny**	**VEGAS**
$\rho_a(x)$ (cienka diagonala)	0.74	0.66	0.002
$\rho_b(x)$ (cienka sfera)	0.47	0.53	0.11
$\rho_b(x)$ (powierzchnia sześcianu)	0.95	1.00	0.30

$\Rightarrow$ Ćw. N4.1: Zastosować program mFOAM z systemu ROOT do obliczenia całek Φ i Ψ z wykładu 2.
Porównać efektywność tego algorytmu oraz klasycznych metod całkowania Monte Carlo.

Porównanie całkowania MC z kwadraturami numerycznymi

- Wszystkie formuły kwadraturowe przebliżają wartość całki przez liniową kombinację wartości funkcji:

$$I_Q = \sum_{i=1}^{m} w_i f(x_i) \, .$$

Różne metody odpowiadają różnym wyborom punktów (węzłów) x_i i wag w_i. Z reguły wycałkowują dokładnie wielomian pewnego stopnia (np. reguła trapezów wycałkowuje dokładnie wielomian 1-go stopnia, reguła Simpsona – wielomian 3-go stopnia itd.).

▷ Metoda podstawowa Monte Carlo może być uważana za formułę kwadraturową z jednostkowymi wagami i punktami wybieranymi równomiernie, ale losowo.

- Wydajności metod całkowania:

Niepewność jako funkcja liczby punktów n	1 wymiar	d wymiarów
Monte Carlo	$n^{-1/2}$	$n^{-1/2}$
Reguła trapezów	n^{-2}	$n^{-2/d}$
Reguła Simpsona	n^{-4}	$n^{-4/d}$
m-punktowa reguła Gaussa	n^{-2m}	$n^{-2m/d}$ $(m \leq n)$

Wnioski:

- **W jednym wymiarze metoda MC jest znacznie wolniej zbieżna od kwadratur numerycznych** (nawet od najprostszej metody trapezów)!
- **W wielu wymiarach efektywność metody MC względem kwadratur poprawia się** (w tym sensie, że zbieżność metody MC jest taka sama, a zbieżność kwadratur pogarsza się).

  ▷ Metoda MC jest **szybciej zbieżna** od:

  reguły trapezów dla $d > 4$, reguły Simpsona dla $d > 8$,

  3-punktowej reguły Gaussa dla $d > 12$, 10-punktowej reguły Gaussa dla $d > 40$.

  Uwaga: Dla 10-punktowej reguły Gaussa w 40 wymiarach potrzeba obliczyć 10^{40} wartości funkcji – **praktycznie niemożliwe!**

Wady kwadratur numerycznych:

- Trudne do zastosowania w wielu wymiarach (> 2) – z reguły całkuje się niezależnie w każdym wymiarze. Brak kwadratur prawdziwie wielowymiarowych!
- Trudne do zastosowania dla skomplikowanych obszarów całkowania.
- Nie wszystkie funkcje dają się dobrze przybliżać wielomianami, np. funkcje nieciągłe, nieróżniczkowalne w punktach itd.
- Błędy całkowania są trudne do oszacowania!