МЕТОДИ МОНТЕ КАРЛО

Методът Monte Carlo е алгоритъм, който решава проблема на основата на статистически проби

- Първото му важно използване е през II световна война за създаването на атомната бомба
- Единственият метод за решаване на интеграли на произволни функции за 6 и повече измерения
- Използва се за прогнозиране на индекса Dow Jones

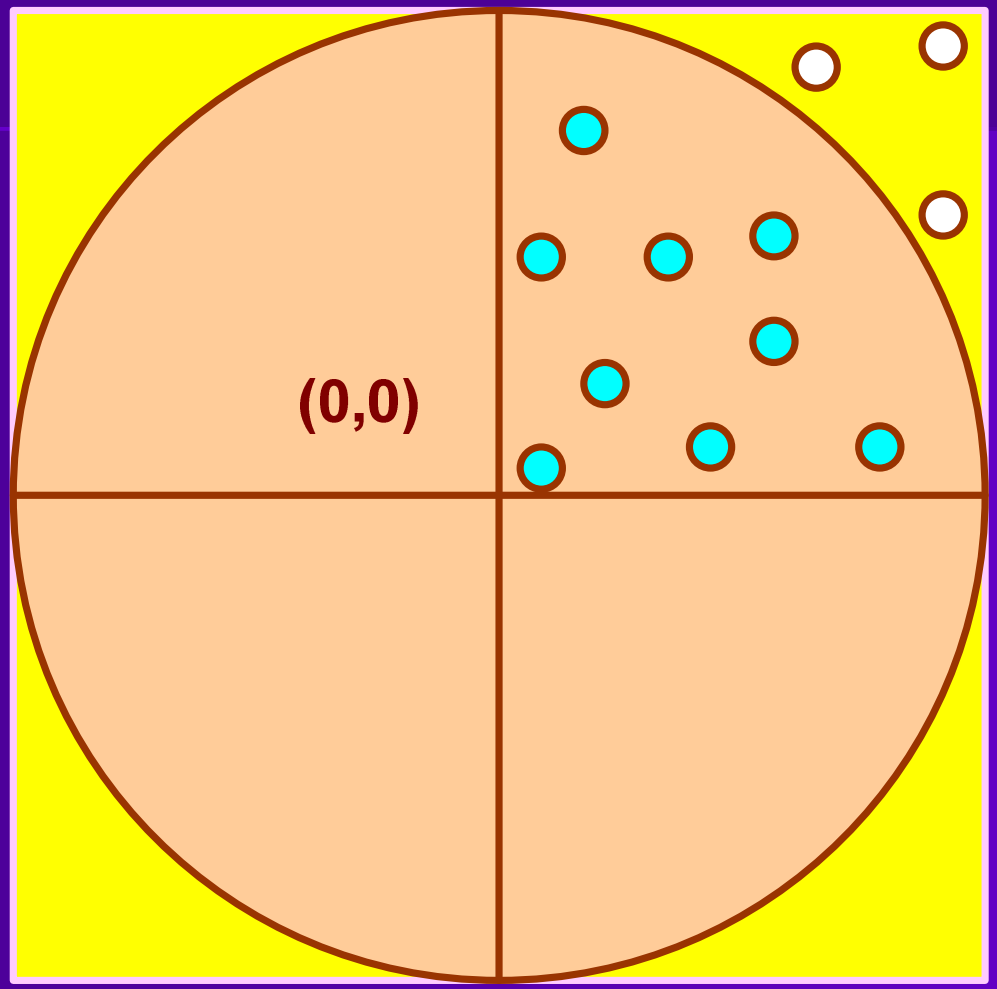
- Решаване на ЧДУ
- Обработка на образи от сателити
- Моделиране на популациите на определени региони
- Намиране на приблизителните решения на проблеми с NP сложност за полиномно време

Примерно приложение

Изчисляване на стойността на числото π

- Площта на кръг с диаметър D е $\pi D^2/4$
- Площта на квадрат със страна D е D^2
  - *Отношението на площите е* $\frac{\pi D^2/4}{D^2} = \frac{\pi}{4}$
- *За оценката използваме случайни числа*
- *Численото интегриране е по-добрата стратегия при малък брой на измеренията*
  - *Пълен кръг с радиус 1 има площ π*
    - *Площта на ¼ от кръга е π е π/4*

- Генерираме серия от двойки *(x,y)*,
- x и y са с равномерно разпределение в интервала 0,1
- x,y са координати на точки вътре в квадрата
- Проследяваме частта f от точките, попадащи *вътре в ¼ от кръга*
  - За тези точки $x^2 + y^2 \leq 1$
    - $f = \pi/4 \rightarrow 4 = f.\pi$
- *Абсолютната грешка бавно намалява с увеличаването на броя на точките*



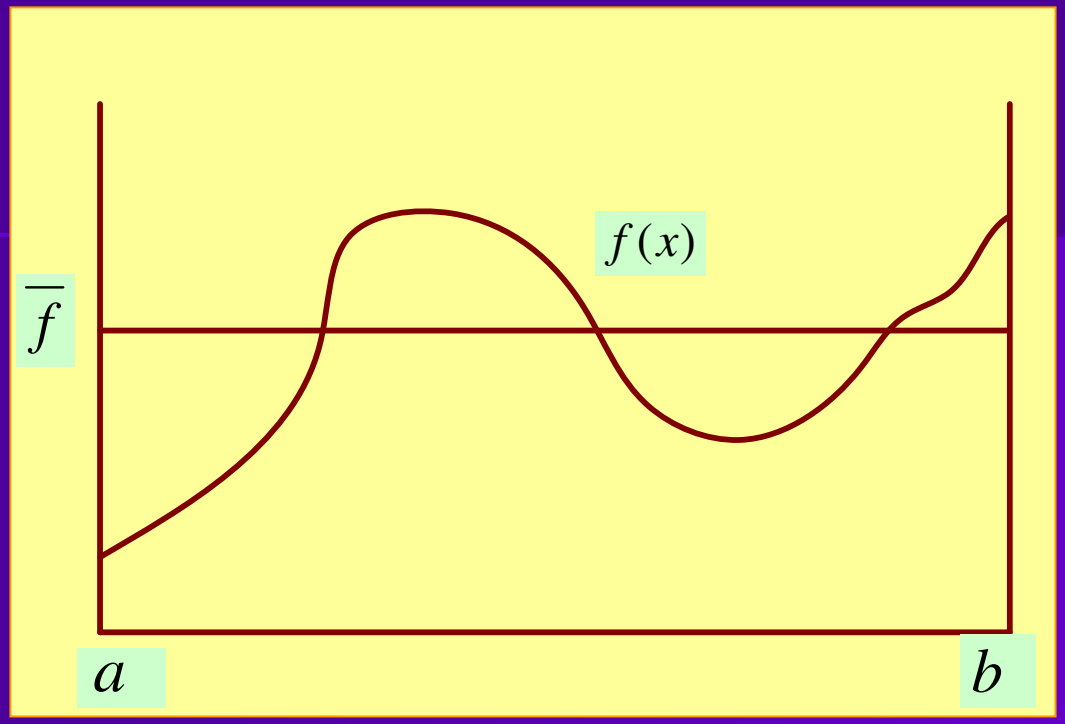
- Според теоремата за средната стойност

$$I = \int_a^b f(x)dx = (b-a)\overline{f}$$

- Оценяваме I чрез изчисляване на $f(x_i)$ в n точки, избрани случайно с равномерно разпределение в интервала [a,b]
- Очакваната стойност е

$$\overline{f} = \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)$$

$$I \approx (b-a)\frac{1}{n}\sum_{i=0}^{n-1} f(x_i)$$



Площта под кривата f(x) е еднаква с тази под $\overline{f}$ - средната стойност на f(x) в интервала [a,b]

- *Грешката при изчисляване на интеграла I намалява с фактора* $1/\sqrt{n}$
- *Скоростта на сходимост е независима от размерността на интегрираната величина*
- *Контраст с детерминистичните числени методи за интегриране, при които скоростта на сходимост намалява с увеличаването на размерността (правило на Simpson)*
- *Методът на Монте Карло превъзхожда детерминистичните методи при брой на измеренията > 6*

Брой на пробите	Оценка на π	Грешка	$1/(2\sqrt{n})$
10	2.40000	0.23606	0.15811
100	3.36000	0.06952	0.05000
1000	3.14400	0.00077	0.01581
10 000	3.13920	0.00076	0.00500
100 000	3.14132	0.00009	0.00158
1 000 000	3.14006	0.00049	0.00050
10 000 000	3.14136	0.00007	0.00016
100 000 000	3.14154	0.00002	0.00005
1 000 000 000	3.14155	0.00001	0.00002

Програма за изчисляване на π по метода Монте Карло

```
#include <stdio.h>
#include <stdlib.h>


int main ( int argc, char *argv[] )
{
   int  count;                  /* точки в кръга */
   int   i;
   int  n;              /* брой на пробите */
   double pi;
   unsigned short xi[3] ; /* seed на случ. генератор */
   double x, y ;                /* координати на точката */
```

Програма за изчисляване на π по метода Монте Карло

```
if (argc != 5) {
    printf ("Correct command line:  ")
    printf ("%s <#samples>  <seed0> <seed1>
          <seed2>\n" , argv[0] ) ;
    return -1;
}
n = atoi (argv[1]);
for (i = 0; i < 3; i++)
     xi[i] = atoi (argv[i+2] ) ;
```

Програма за изчисляване на π по метода Монте Карло

```
    count = 0 ;
    for (i = 0; i < n; i++)   {
        x = erand48 (xi) ;
        y = erand48 (xi) ;
        if (x*x+y*y <= 1.0) count ++;
    }
    pi = 4.0 * (double) count / (double) n;
    printf ("Samples: %d Estimate of pi: %7.5f/n", n, pi);
}
```

Методът Монте Карло и паралелната обработка

- Алгоритмите лесно се трансформират за паралелни системи
- Характеризират се с пренебрежимо малко междупроцесорни комуникации
- Бързодействието при p процесора е p пъти и грешката при оценката се намалява с фактора $\sqrt{p}$
- При положение, че случайните числа са статистически независими

Методи Монте Карло

- ***Основно предизвикателство***
  → <u>добри паралелни генератори на случайни числа</u>
- ***Половината от машинното време на суперкомпютрите се използва за обработка по методите на Монте Карло***

Последователни генератори на случайни числа

- При съвременните компютри → ***генератори на псевдо – случайни числа***
- Тяхното действие е детерминистично → последователностите, които създават са предсказуеми
- Тези последователности са *разумна апроксимация на случайна последователност*
- **Под генератор на случайни числа ще разбираме генератор на псевдо – слуайни числа**

Свойства на идеалния генератор на случайни числа

- Равномерно разпределение
- Числата не са корелирани
- Числата никога не се повтарят
- Удовлетворяват всеки тест за случайност
- Машинно – независими – генераторът създава една и съща последователност на различни машини
- Лесно може да се променя последователността като се променя “seed”

Свойства на идеалния генератор на случайни числа

- Лесно се разцепва на множество независми последователности
- Последователността се генерира бързо
- Генераторът изисква ограничена компютърна памет
- Не съществуват реални генератори, удовлетворяващи всички изисквания
- Аритметика с ограничена точност
- Периодът на генератора е неговия цикъл

Линейно конгруентни генератори

- Най-популярните (50 години)
- Генерират последователност X_i от случайни числа по формулата: $X_i=(a.X_{i-1}+c) \bmod M$

  a – множител, *c* – адитивна константа,

  M – модул
- Трите стойности трябва да се подберат много внимателно за да има генераторът дълъг период и добри качества
- За 32-битови числа max период е 2^{32}, или около 4 милиарда

Линейно конгруентни генератори

- Използват се и за генериране на числа с плаваща точка
- Генерират се цели числа от *0* до *M-1*, X_i се дели на *M*, като се създават числа с плаваща точка x_i в интервала [0,1)
- Недостатъци:
  - Младшите битове са колерирани (при к-мерен куб се генерира решетка)
  - Влошават се при увеличаването на размерността
  - При 48-бита са добри за последователните компютри

Генератори Lagged Fibonacci

- Последователности с големи периоди
- Високо бързо действие
- Всеки елемент X_i на последователността се дефинира като $X_i = X_{i-p} \text{ ¤ } X_{i-q}$

p,q – lags, p>q, ¤ - произволна двоична операция – сума по модул M, изваждане по модул M (цели числа или числа с плаваща точка), умножение по модул M (нечетни цели числа) и побитово изключващо OR

Генератори Lagged Fibonacci

- Функцията random, извиквана от C, е Фибоначи генератор на случайни числа – default lag е 31
- Coddington – 31 е малко – препоръчва (p,q) да бъдат поне (1279, 1063)
- Увеличаването на p води до увеличаване на периода, но и завишава изискванията за памет

ПАРАЛЕЛНИ ГЕНЕРАТОРИ НА СЛУЧАЙНИ ЧИСЛА

Изисквани качества:

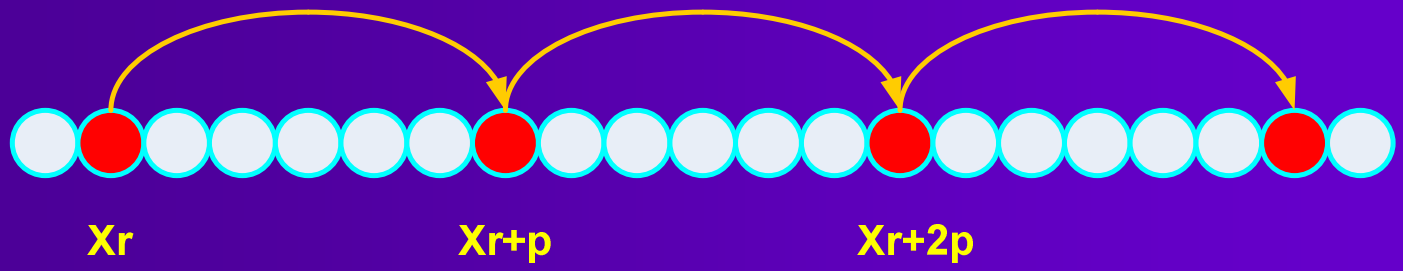
- *Не трябва да има корелации между отделните последователности*
- *Мащабируемост – голям брой процеси, всеки със свой поток от числа*
- *Локалност – всеки процес да бъде в състояние да генерира нова последователност от числа без необходимост от комуникация с други процеси*

Метод Manager-Worker

- Процес manager – генерира случайни числа и ги разпределя между процесите worker
  - Недостатъци:
    - Някои слуайни генератори създават последователности с широкообхватни корелации
    - Всички процеси получават числа от една и съща последователност
    - Риск: получаване на корелации в малък обхват на паралелните последователности

Метод "скок на жабата"

- Циклично разпределяне на данните между задачите
- Всичките p процеса използват един и същ последователен генератор на случайни числа
- Процес p с ранг r взема всеки p-ти елемент от последователността, започвайки с X_i



- *Линейният конгруентен генератор лесно се трансформира в генератор на Фибоначи*

  a се заменя с $a^p \bmod M$ и

  c със $c(a^p-1) \bmod M$

- Често се изисква да се генерират многомерни случайни стойности – модификация на метода leapfrog

  $(X_{2r}, X_{2r+1}, X_{2r+2p}, X_{2r+2p+1}, \ldots)$

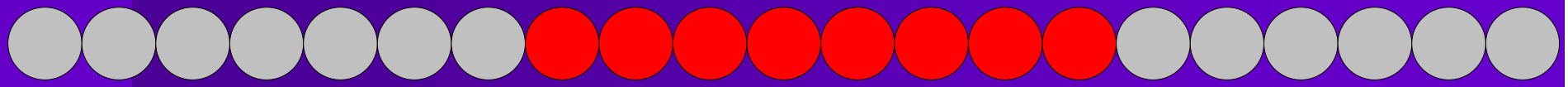
Метод "скок на жабата"

Недостатъци:

- Елементите в генерираната последователност могат да бъдат корелирани за определени стойности на p, дори когато елементите на изходната последователност имат ниска корелация
- Особено при използването на линейно конгруентния метод, p е степен на 2 и M е степен на 2
- Широкообхватните корелации → малообхватни корелации

Разцепване на последователността

- Аналогично на разпределението на данните по блокове между задачите
- Изходният генератор има период p
- Първите p числа от генератора се разделят на равни части между процесите, по 1 част на процес
- Риск от корелации между последователностите на процесите



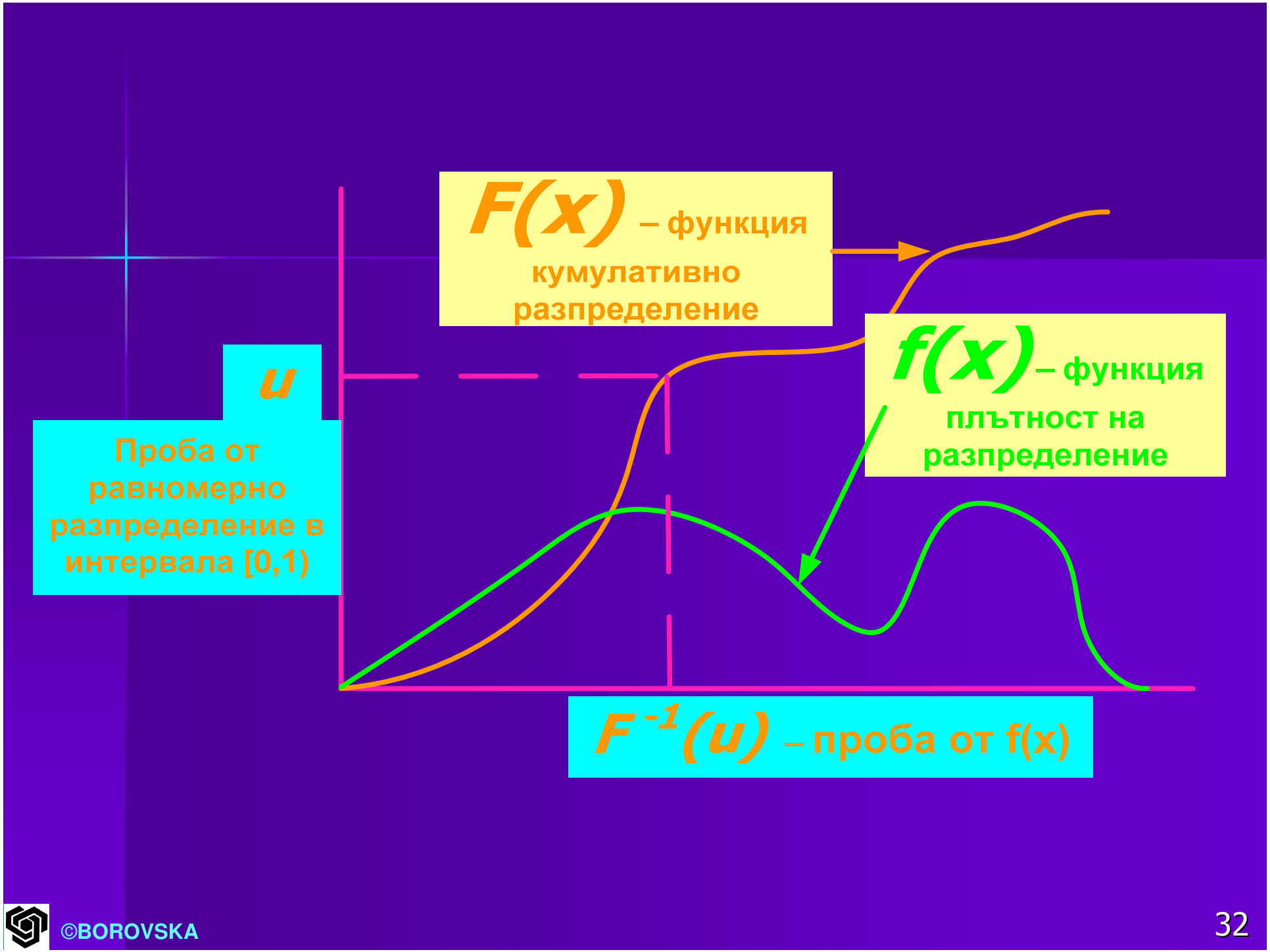
Параметризация

- Всеки процес изпълнява отделен последователен случаен генератор
- Инициализацията на генератора се осъществява с различни параметри
- Линейните конгруентни генератори използват различни конгруентни константи за генерирането на различни потоци
- Генераторите Фибоначи използват различни начални таблици с lags
- Отделен генератор създава “seeds”

Други разпределения на случайните числа

- Нека ***u*** преставлява стойност на случайна функция с равномерно разпределение в интервала **[0,1)**
- Целта е да се генерира последователност от случайни числа от функцията на плътността на разпределението на вероятностите ***f(x)***

- Определяме кумулативната функция на разпределението ***F(x)*** и я инвертираме
  - ***Тогава $F^{-1}(u)$ е случайна променлива с функция на плътността на разпределение f(x)***
- На основата на тази трансформация се извежда ***формула, която дава проба от експоненциално разпределение***


F(x) – функция кумулативно разпределение
f(x) – функция плътност на разпределение
u
Проба от равномерно разпределение в интервала [0,1)
F -1(u) – проба от f(x)

Експоненциално разпределение

- Разпадането на радиоактивните атоми
- Дистанцията на пътя на неутрона в твърдо тяло преди да взаимодейства с атом
- Времето за пристигане на нов клиент в център за обслужване
- ***Функцията на експоненциалната плътност на вероятностите с очаквана стойност m е*** $f(x) = (1/m)e^{-x/m}$
- ***Може да интегрираме*** $f(x)$ ***за да намерим кумулативната функция на разпределение*** $F(x) = 1 - e^{-mx}$

Експоненциално разпределение

- Инвертираме F(x)
- Получаваме инверсната функция

$$F^{-1}(u) = -m \ln(1-u)$$

- Поради равномерното разпределение на u в интервала [0,1)

няма разлика между u и 1-u

- $F^{-1}(u) = -m \ln(u)$

е с експоненциално разпределение със средна стойност m

Пример – да се генерират 4 проби с експоненциално разпределение и очаквана средна стойност 3

- Започваме с 4 проби с равномерно разпределение: 0.540; 0.619; 0.462; 0.095;
- Изчисляваме натурален алгоритъм от всяка стойност и я умножаваме по 3

-3ln(0.540); -3ln(0.619); -3ln(0.462); -3ln(0.095)

- Получаване 4 проби с експоненциално разпределение с очаквана средна стойност 3:

1.850; 1.440; 2.317; 7.072

Задача

- Стъпката на симулация е 1 сек.
- Вероятността за настъпването на събитие е с експоненциално разпределение и очаквана средна стойност 5 сек.
- Каква е вероятността събитието да се случи през следващата стъпка на симулация?
- Как да се определи дали събитието ще се случи през следващата стъпка?

Решение

- Вероятността събитието да възникне през следващата стъпка е 1/5.
- За да определим дали събитието ще се случи през следващата стъпка генерираме случайно число с равномерно разпределение в интервала [0,1).
  - Ако случайното число е <1/5, събитието ще се случи.

Трансформация на Box-Muller

- Целта е да се генерират числа с нормално (Гаусово) разпределение.

$$f(x)=\frac{1}{\sqrt{2\pi}}\ell^{-x^2/2}$$

- Трансформацията дава възможност да се генерират двойка стандартни отклонения g_1 и g_2 от двойка равномерни отклонения u_1 и u_2

repeat

$v_1 \leftarrow 2u_1 - 1$

$v_2 \leftarrow 2u_2 - 1$

$r \leftarrow v_1^2 + v_2^2$

until $r>0$ and $r<1$

$f \leftarrow SQRT(-2\ln r/r)$

$g_1 \leftarrow f\, v_1$

$g_2 \leftarrow f\, v_2$

Трансформацията на Box-Muller може да се използва за създаването на функция, която връща стойност със стандартно отклонение.

При нечетните извиквания (1,3,5,...) на функцията се съхранява g_2 и се връща g_1. При четните извиквания (2,4,6,...) се връща стойността на g_2, формирана при предходното извикване.

Пример: да се определят 4 проби с нормално разпределение със средна стойност 0 и стандартно отклонение 1.

u_1	u_2	v_1	v_2	r	f	g_1	g_2
0.234	0.784	-0.532	0.568	0.605	1.290	-0.686	0.732
0.824	0.039	0.648	-0.921	1.269			
0.430	0.176	-0.140	-0.648	0.439	1.935	-0.271	-1.254

Пример: да се определят 4 проби с нормално разпределение със средна стойност 0 и стандартно отклонение 1.

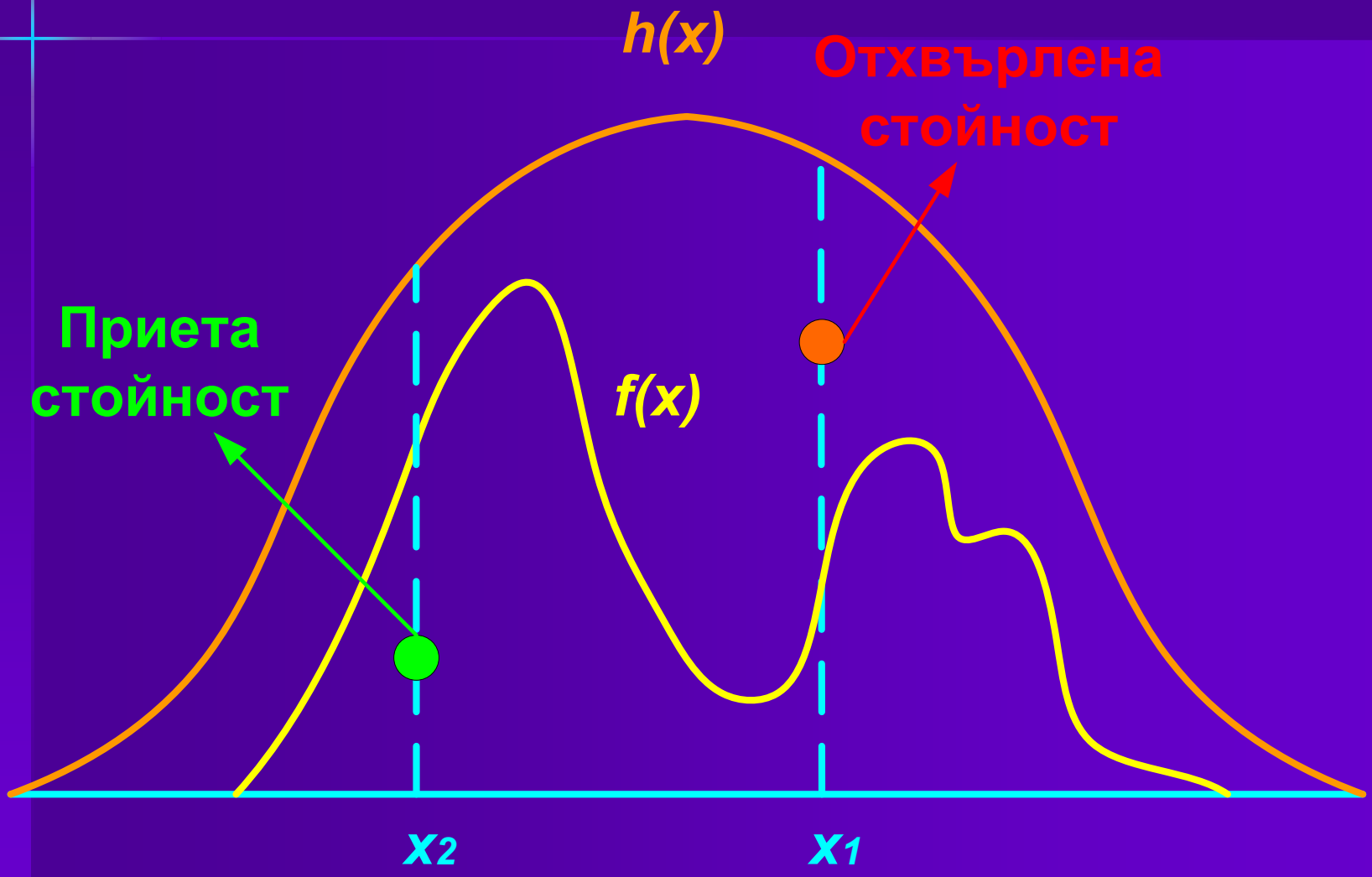
Решение: Пробите с равномерно разпределение 0.234 и 0.784 преобразуваме в две проби с нормално разпределение -0.686 и 0.732. При преобразуването на следващата двойка 0.824 и 0.039 се получава $r<1$, така че пренебрегваме тази двойка и генерираме нова. При преобразуването на двойката 0.430 и 0.176 получаваме двойката проби -0.271 и -1.254.

МЕТОД НА ОТХВЪРЛЯНЕТО (REJECTION METHOD)

- *Предложен от Джон фон Нойман*
- Възможност да се създават стойности с функция на плътност на разпределението f(x), които не можем да интегрираме и/или инвертираме по аналитичен път
- Генерираме проби с плътност на разпределение *h(x)*
- Можем да намерим константа δ такава, че за всички x

$$f(x) \leq \delta h(x)$$

МЕТОД НА ОТХВЪРЛЯНЕТО (REJECTION METHOD)

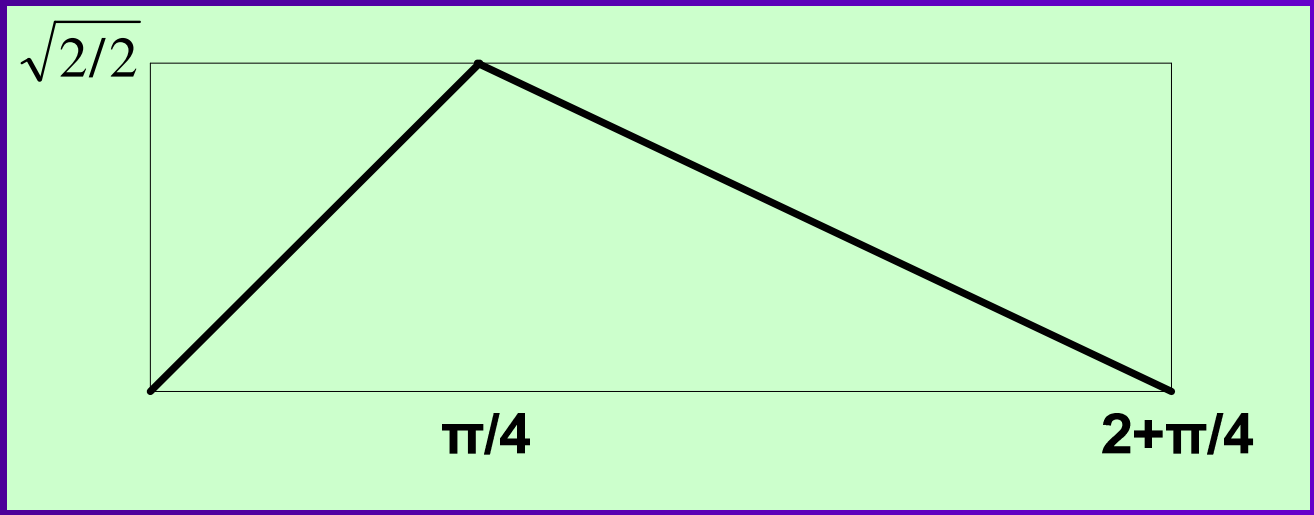


- Приемаме само тези точки, които са под кривата f(x)
  - *Методът с отхвърлянето е най-добър, когато грешката между f(x) и δh(x) е относително малка → колкото е по-голяма площта на грешката, толкова по-често ще отхвърляме пробите – забавяне на процеса*
- Ефективността на метода рязко намалява при увеличаване на броя на измеренията
- Напр., ако за едномерно интегриране са приети 75% от пробите, то за 6-мерен интеграл само 18% ще бъдат приети

Пример:

- **Случайна величина има плътност на разпределение:**

$$f(x)=\begin{cases}\sin x & 0\le x\le \pi/4\\ (-4x+\pi+8)/(8\sqrt{2}) & \pi/4<x\le 2+\pi/4\\ 0 & \end{cases}$$



- Избираме функция с равномерна плътност ***h(x)*** за която:

$$h(x) = \begin{cases} 1/(2+ \pi/4), \text{ за } 0 \le x \le 2+\pi/4 \\ 0, \text{ в останалите случаи} \end{cases}$$

- Ако умножим ***h(x)*** с ***δ=(2+ π/4)(sqrt(2)/2),*** то ще получим ***δ.h(x) >=f(x)*** за всички x

$$\delta.h(x) = \begin{cases} sqrt(2)/2, \text{ за } 0 \le x \le 2+\pi/4 \\ 0, \text{ в останалите случаи} \end{cases}$$

- Генерираме случайно число с равномерно разпределение между 0 и 1 и го умножаваме с $(2+\pi/4)$ $\rightarrow$получаваме случайно число x_i с равномерно разпределение между 0 и $2+\pi/4$
- Генерираме случайно число u_i с равномерно разпределение между 0 и 1
- Приемаме x_i като проба от $f(x)$ ако

$$u_i \delta h(x_i) \leq f(x_i)$$

- В противен случай генерираме нова двойка (x_i, u_i) и повтаряме теста

x_i	u_i	$u_i \delta h(x_i)$	$f(x_i)$	Резултат
0.860	0.975	0.689	0.681	отхвър-лена
1.518	0.357	0.252	0.448	приета
0.357	0.920	0.650	0.349	отхвър-лена
1.306	0.272	0.192	0.523	приета

КРАЙ