УДК 004.9

Программная реализация генерирования последовательности псевдослучайных чисел при помощи метода обратной функции

Лаптева Анастасия Игоревна
Брянский государственный университет имени академика И.Г. Петровского
Студент

Аннотация
В статье рассмотрен способ генерации псевдослучайных чисел методом обратной функции и приводится его программная реализация.
Ключевые слова: случайные числа, псевдослучайные числа, метод обратной функции.

A software implementation of generating a sequence of pseudo-random numbers using the inverse function method

Lapteva Anastasiya Igorevna
Bryansk State University named by academician I.G. Petrovsky
Student

Abstract
In the article the way of generation of pseudo-random numbers by the inverse function method is considered and its program realization is given.
Keywords: random numbers, pseudo-random numbers, inverse function method.

Генерирование случайных последовательностей с заданным вероятностным законом и проверка их адекватности — одни из важнейших проблем современной криптологии. Генераторы случайных последовательностей используются в существующих криптосистемах для генерации ключевой информации и задания ряда параметров криптосистем.

Случайные числа – это такая последовательность чисел, для которой невозможно предсказать следующее даже зная все предыдущие. Псевдослучайные числа – это такая последовательность чисел, которая обладает свойствами случайных чисел, но каждое следующее число вычисляется по заданной формуле. Одним из преимуществ псевдослучайных чисел является их быстрая генерация, к тому же они не требуют запоминающих устройств, но запас псевдослучайных чисел ограничен.

Два основных требования к последовательности случайных чисел – это случайность и непредсказуемость.

Для данного случая генерации псевдослучайных чисел используется метод обратной функции, который рассмотрим далее. Суть заключается в том, что, используя h длина интервала и последовательность Y_n, программа

вычисляет формулу, по которой в дальнейшем вычисляется последующая часть последовательности.

Для использования метода обратной функции необходимо иметь в явном виде аналитическое выражение для функции распределения вероятностей $F_\zeta(x)$. Как правило, законы распределения задаются функцией плотности $f_\zeta(x)$, а функцию распределения получают интегрируя $f_\zeta(x)$:

$$F_\xi(x) = \int_{-\infty}^{x} f_\zeta(t)\,dt.$$

Для многих распределений использование метода обратной функции в аналитическом виде оказывается затруднительным или невозможным по ряду причин:

а) интеграл от $f_\zeta(x)$ не берётся, либо после интегрирования получается выражение, требующее больших затрат машинного времени;

б) описание вида распределения получено экспериментально в виде гистограммы.

В этих случаях используют универсальный способ получения случайных чисел, основанный на кусочной аппроксимации функции плотности.

Пусть требуется получить последовательность случайных чисел $\{x_i\}$ с функцией плотности распределения $f_\zeta(x)$.

Если область определения случайной величины ζ не ограничена, перейдём к усеченному распределению на интервале (*c*,*d*). Разобъём интервал (*c*,*d*) на *n* подинтервалов (рис. 1):

$$(a_0, a_1), (a_1, a_2), \ldots, (a_{n-1}, a_n), \quad a_0 = c, a_n = d.$$

Границы интервалов выбираются так, чтобы вероятность попадания в любой из подинтервалов (a_k, a_{k+1}) была постоянной, то есть не зависела от *k*. Для вычисления границы a_k воспользуемся следующим соотношением:

$$P_k = \int_{a_{k-1}}^{a_k} f(t)\,dt = \frac{1}{n}.$$

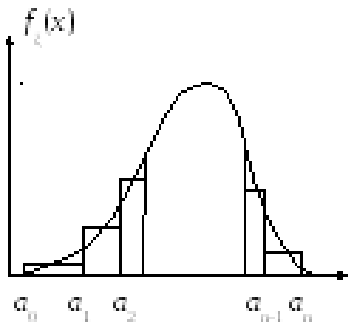

Рис.1. Кусочная аппроксимация функции плотности

В простейшем случае будем представлять $f_\zeta(x)$ в виде кусочно-постоянной функции, то есть считаем значение $f_\zeta(x)$ на каждом подинтервале постоянной. Тогда значение случайной величины ξ, попадающей в подинтервал (a_k, a_{k+1}) можно представить, как

$$x_k = a_k + \eta_k *,$$

где:

a_k – левая граница подинтервала;

$\eta_k *$ – РРСЧ, значение которого распределены от 0 до $(a_{k+1} - a_k)$.

Таким образом, вероятность всех чисел из любого подинтервала (a_k, a_{k+1}) одинакова.

Машинный алгоритм этого способа:

1) Генерируется РРСЧ η_1 из диапазона *(0, 1)* и выбирается номер *k* подинтервала из условия:

$$k \le \eta - \eta_1 < k, k = 0, \ldots n-1.$$

2) Генерируется РРСЧ η_2 (от 0 до 1) и формируется случайная величина выходной последовательности

$$x_k = a_k + (a_{k+1} - a_k)\eta_2.$$

Универсальный метод при кусочно-постоянной аппроксимации $f_\zeta(x)$ весьма прост в реализации и требует малого количества операций ЭВМ. Когда требуется обеспечить особо высокую точность преобразования случайных чисел могут оказаться полезными и другие виды аппроксимации, например, линейно-кусочная. Правда, выражение для получения x_k при этом существенно усложняется.

В случае, когда описание вида распределения задано в виде гистограммы, применяют модификацию универсального метода.

Пусть требуется произвести формирование случайной последовательности $\{x_i\}$, причем закон распределения для нее представлен в виде равноинтервальной гистограммы (рис. 2), содержащей n интервалов с границами $a_0, a_1, \ldots, a_{n-1}, a_n$.

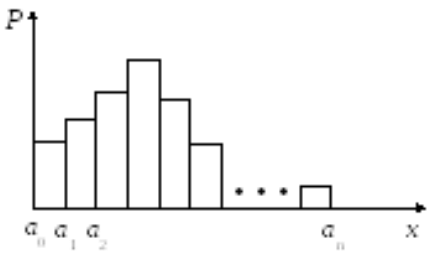

Рис. 2. Задание закона распределения гистограммой

Построим вспомогательную шкалу:

$0, c_1, \ldots, c_{n-1}, c_n, 1,$

где $c_i = \sum_{i=1}^{n-1} P_i$, P_i – относительная частота попадания в i-й интервал гистограммы.

Машинный алгоритм реализации способа:

1) Генерируется РРСЧ η_1 из диапазона (0,1) и выбирается номер k интервала вспомогательной шкалы из условия:

$$c_k \le \eta_1 < c_{k+1},\ k = 0,\ldots n-1.$$

2) Генерируется РРСЧ η_2 (от 0 до 1) и формируется случайная величина выходной последовательности

$$x_k = a_k + \left(a_{k+1} - a_k\right)\eta_2 .$$

Этот метод можно применить и в том случае, если задана $f\left(x\right)$ в аналитическом виде. Для этого предварительно рассчитывают значения P_i :

$$P_i = \int_{a_{i-1}}^{a_i} f\left(t\right)dt$$

или

$$P_i = \frac{a_n - a_0}{n} \cdot \frac{f\left(a_i\right) + f\left(a_{i+1}\right)}{2},$$

а далее по описанному выше алгоритму.

Исходными данными в программе являются: N (длина оси Ox), h (длина интервала) и последовательность Y_n (возрастающая последовательность).

Последовательность X_n получаем по алгоритму, описанному в коде программы.

Далее задаём рандомное заполнение Rand, случайными числами от 0 до 1, не включая сами 0 и 1. Теперь нужно проверить выполнение условия принадлежности промежутку от максимального Y_n до минимального Y_{-n}.

В конце остается подставить в уравнение

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

имеющиеся значения и вывести $X -$ следующее в последовательности случайное число.

Тестирование проводилось на основе последовательности чисел, полученных равномерным распределением.

На отрезке от a=1 до b=3 с начальными значениями:

1.256
1.4567
1.5689
2.45
2.4587
2.65789

Программы выдала последовательность (рис.3).

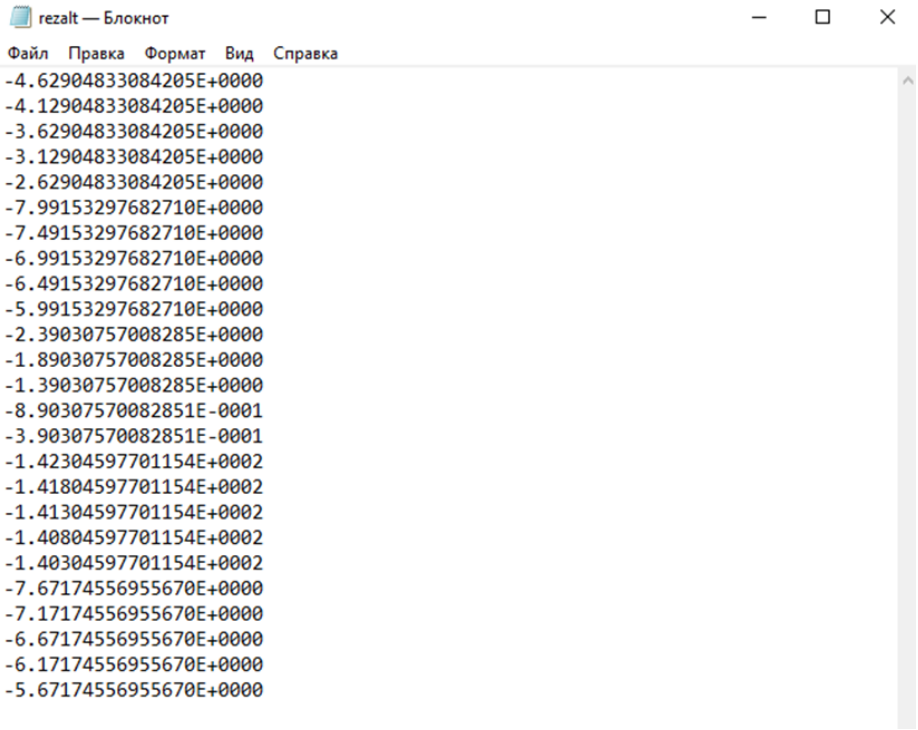

Рис. 3

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons;

type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    BitBtn1: TBitBtn;
    Label4: TLabel;
    Memo1: TMemo;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;

var
  Form1: TForm1;
  //s:integer;
  i,n,j,k,c,l: integer;
  b:array[1..10000] of real;
  a:array[1..10000] of real;
  Rand,h,k1,x:real;
  f: textfile;
implementation

{$R *.dfm}

procedure TForm1.BitBtn1Click(Sender: TObject);

begin
    n:=StrtoInt(Edit1.Text);
   h:=StrToFloat(Edit2.Text);
   assigntext(f,'source.txt');
   reset(f);
   For i:=1 to 2*n do
      readln(f,b[i]);
     closefile(f);

          Получение последовательности  X_n

 For j := 1 to 2*n do
     j:=1;
      For k := -n to n do
      begin
        a[j]:=h*k;
        j:=j+1;
      end;

          Получение рандомных значений

begin
      randomize;
      for c := 1 to n do
        Rand:=random(1);
      if (Rand<b[1]) or (Rand>b[2*n]) then
          for l := -n+1 to n do
```

```
        if (Rand<b[1]) or (Rand>b[2*n]) then
         k1:=l;
    end;

      Получение новых значений последовательности и вывод

assignfile(f,'rezalt.txt');
    rewrite(f);
    For i:=2 to 2*n do
        For j := 2 to 2*n do
        begin
           x:=((a[j]-a[j-1])*(Rand-b[i-1])+a[j-1]*(b[i]-b[i-1]))/(b[i]-b[i-1]);
           Memo1.Lines.Add(Floattostr(x));
           writeln(f,x);

        end;
  closefile(f);
    end;

end.
```

Библиографический список

1. Кнут Д. Э. Глава 3. Случайные числа // Искусство программирования. М.: Вильямс, 2000. 832 с.
2. Кнут Д. Э. Искусство программирования = The Art of Computer Programming. Получисленные алгоритмы. М.: Вильямс, 2017. Т. 2. 832 с.
3. Дроздова И. И., Жилин В. В. Генераторы случайных и псевдослучайных чисел [Текст] // Технические науки в России и за рубежом: материалы VII Междунар. науч. конф. (г. Москва, ноябрь 2017 г.). М.: Буки-Веди, 2017. С. 13-16. URL: https://moluch.ru/conf/tech/archive/286/13233/ (дата обращения: 20.06.2018).
4. Лаптева А.И. Программная реализация построения последовательности псевдослучайных значений непрерывной случайной величины с заданным законом распределения // Постулат. 2017. № 12. URL http://e-postulat.ru/index.php/Postulat/article/view/1037/1063