**Лабораторная работа №2**

**Способы получения случайных чисел**

**с заданным законом распределения законом**

**(Часть 1. Дискретные случайные величины)**

по дисциплине

«Статистическое моделирование случайных процессов и систем»

|  |  |  |
| --- | --- | --- |
| Выполнил студент  гр. 33504/2 | Лелюхин Д. О. |  |
| Руководитель | Чуркин В.В. |  |

**Оглавление**

[**Цель работы** 3](#_Toc509697699)

[**Разработка программных датчиков дискретных случайных величин** 4](#_Toc509697700)

[**Равномерное распределение** 5](#_Toc509697701)

[**Биномиальное распределение** 6](#_Toc509697702)

[**Геометрическое распределение** 7](#_Toc509697703)

[**Распределение Пуассона** 9](#_Toc509697704)

[**Логарифмическое распределение** 11](#_Toc509697705)

[**Вывод** 12](#_Toc509697706)

## **Цель работы**

1. Практическое освоение методов получения случайных величин, имеющих дискретный̆ характер распределения.
2. Разработка программных датчиков дискретных случайных величин.
3. Исследование характеристик моделируемых датчиков:
   1. Оценка точности моделирования: вычисление математического ожидания и дисперсии, сравнение полученных оценок с соответствующими теоретическими значениями.
4. Графическое представление функции плотности распределения и интегральной функции распределения.

## **Разработка программных датчиков дискретных случайных величин**

Программный датчик случайных чисел реализован на языке Python 3.6 с использованием библиотеки numpy и random. Для построения графиков используется библиотека matplotlib.

В данной работе были реализованы генераторы случайных чисел по следующим законам распределения:

1. Равномерный
2. Биномиальный
3. Геометрический
4. Пуассоновский
5. Логарифмический

Каждый алгоритм протестирован на выборке размером 10000, построены графики функции распределения, плотности распределения и таблица для сравнения теоретический характеристик с экспериментальными.

## **Равномерное распределение**

Данное распределение получено при помощи функции IRNUNI, возвращающей равномерно распределенные целые числа в интервале [ilow, iup], где ilow и iup задаются пользователем.

clc

clearvars

syms x iup ilow N i p M D Mteor Dteor Fx;

Mteor=50.5;

Dteor=833.5;

ilow=1;

iup=100;

N=10000;

x=zeros(1,N);

p=zeros(1,N);

Fx=zeros(1,N);

for i=1:N

x(i) = IRNUNI(ilow, iup);

end

i=1:N;

p(i) = vpa(1/iup-ilow);

for i=1:N

if (ilow<=x(i)<=iup)

Fx(i)=(x(i)-ilow)/(iup-ilow);

elseif (x(i)<ilow)

Fx(i)=0;

elseif (x(i)>iup)

Fx(i)=1;

else

continue;

end

end

set(gcf, 'Position', [300, 100, 800, 600])

%Построение графика функции плотности

subplot(2,1,1);

histogram(x,'Normalization','probability','NumBins',20);

grid on;

xlim([ilow,iup]); ylim([0,1]);

title('График функции плотности f(x)','fontsize',15);

xlabel('x'); ylabel('f(x)', 'rotation', 1);

% Построение графика интегральной функции распределения

subplot(2,1,2);

stairs(sort(x),sort(Fx),'LineWidth',0.8,'MarkerFaceColor','c');

grid on;

xlim([ilow,iup]);

ylim([0,1]);

title('График интегральной функции распределения F(x)','fontsize',15);

xlabel('x'); ylabel('F(x)', 'rotation', 1);

%Вычисление матожидания

M = sum(x) / N;

%Вычисление дисперсии

D = sum(((x - M).^2)/N);

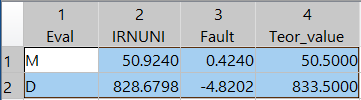
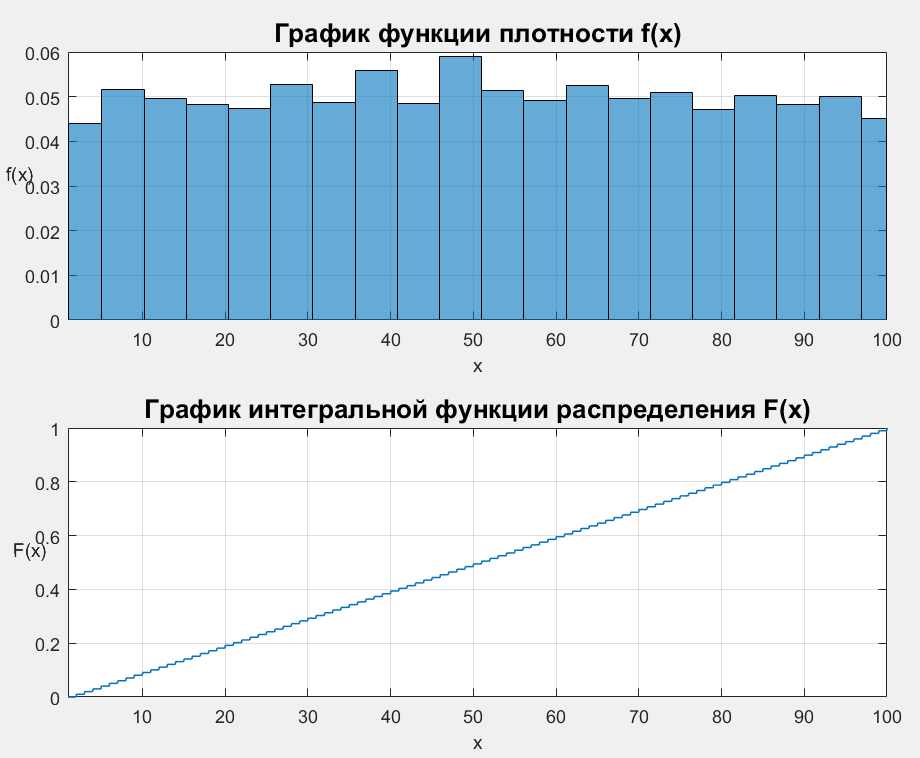
T=table(['M';'D'],[M;D],[M-Mteor;D-Dteor],[Mteor;Dteor]);

T.Properties.VariableNames = {'Eval' 'IRNUNI' 'Fault' 'Teor\_value'};

function IR = IRNUNI(ILOW, IUP)

IR = round(((IUP-ILOW+1)\*rand+ILOW), 0);

end



## **Биномиальное распределение**

Используется кумулятивный метод моделирования, где границы интервалов можно найти в соответствии с рекуррентной формулой:

p(0)=(1-p)\*\*N,

... ... ... ... ... ... ... ... ...

... ... ... ... ... ... ... ... ...

p(r)=p(r-1)\*[((N-r)/(r+1))\*(p/(1-p))],

Следует помнить, что предложенный алгоритм не следует использовать, если N принимает большие значения (например, N>=100). В этом случае необходимо использовать нормальную аппроксимацию:

IR = RNNORM(N\*p,SQRT(N\*p\*(1.0-p))) + 0.5

clc

clearvars

syms N r i p ;

N=10;

p=0.5;

n=10000;

Mteor=N\*p;

Dteor=N\*p\*(1-p);

r=zeros(1,n);

for i=1:n

r(i) = IRNBIN(N, p);

end

x=zeros(1,N+1);

x=unique(r);

fx=zeros(1,length(x));

Fx=zeros(1,length(x));

for i=1:length(x)

fx(i) = binomi(N,x(i))\*(p^x(i))\*((1-p)^(N-x(i)));

Fx(i) = sum(fx);

end

set(gcf, 'Position', [300, 100, 800, 600])

%Построение графика функции плотности

subplot(2,1,1);

histogram(r,'Normalization','probability','NumBins',N);

hold on

plot(x,fx,'ko-');

hold off

grid on;

xlim([min(r),max(r)]);

%ylim([0,1]);

title('График функции плотности f(x)','fontsize',15);

xlabel('x'); ylabel('f(x)', 'rotation', 1);

% Построение графика интегральной функции распределения

subplot(2,1,2);

stairs(sort(r), (1:length(r))/length(r),'LineWidth',0.8,'MarkerFaceColor','c');

grid on;

xlim([min(r),max(r)]);

ylim([0,1]);

title('График интегральной функции распределения F(x)','fontsize',15);

xlabel('x'); ylabel('F(x)', 'rotation', 1);

hold on

plot(x,Fx,'k-');

hold off

%Вычисление матожидания

M = sum(r) / n;

%Вычисление дисперсии

D = sum(((r - M).^2)/n);

T=table(['M';'D'],[M;D],[M-Mteor;D-Dteor],[Mteor;Dteor]);

T.Properties.VariableNames = {'Eval' 'IRNBIN' 'Fault' 'Teor\_value'};

function IR=IRNBIN(N,p)

if (N>=100)

IR = round(RNNORM(N\*p,sqrt(N\*p\*(1-p)))+0.5);

else

alpha = rand;

p\_it = (1-p)^N;

IR = 0;

while ((alpha-p\_it)>=0)

alpha = alpha - p\_it;

p\_it = p\_it\*(p\*(N-IR))/((IR+1)\*(1-p));

IR = IR + 1;

end

end

end

function IR=RNNORM(mu,sigma)

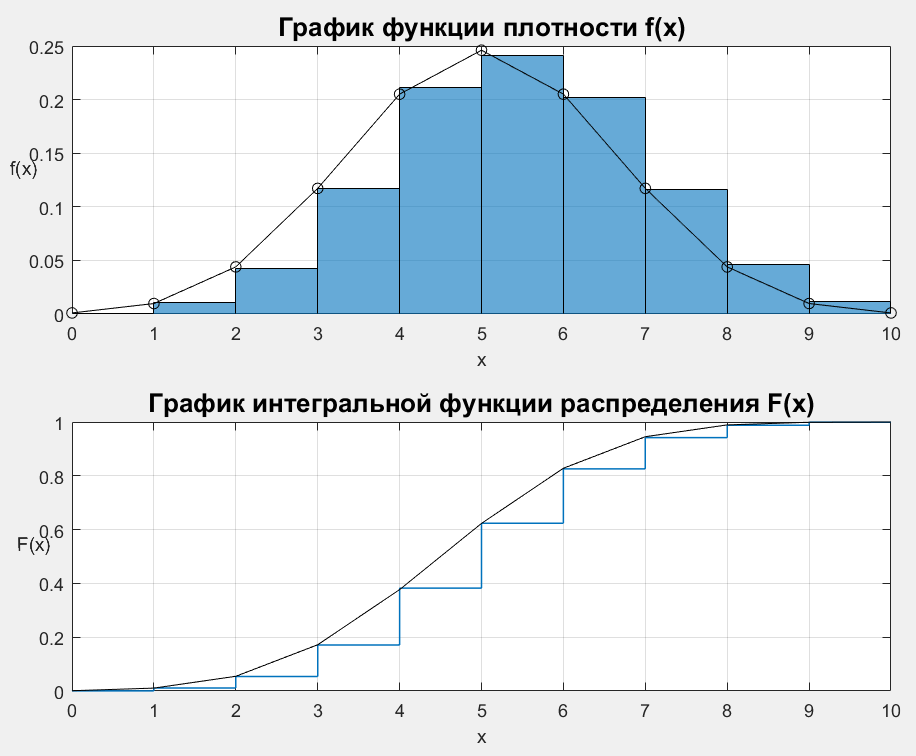
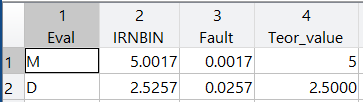
IR = normrnd(mu, sigma);

end

function IR=binomi(n,k)

IR=factorial(n)/(factorial(n-k)\*factorial(k));

end

## **Геометрическое распределение**

clc

clearvars

syms n p Mteor Dteor r1 r2 r3;

n=10000;

p=0.5;

q=1-p;

Mteor=1/p;

Dteor=(1-p)/(p^2);

r1 = zeros(1,n);

r2 = zeros(1,n);

r3 = zeros(1,n);

for i=1:n

r1(i) = IRNGEO\_1(p);

r2(i) = IRNGEO\_2(p);

r3(i) = IRNGEO\_3(p);

end

x1=unique(r1);

fx1=zeros(1,length(x1));

Fx1=zeros(1,length(x1));

for i=1:length(x1)

fx1(i) = p\*q^(x1(i)-1);

Fx1(i) = sum(fx1)-1;

end

x2=unique(r2);

fx2=zeros(1,length(x2));

Fx2=zeros(1,length(x2));

for i=1:length(x2)

fx2(i) = p\*q^(x2(i)-1);

Fx2(i) = sum(fx2)-1;

end

x3=unique(r3);

fx3=zeros(1,length(x3));

Fx3=zeros(1,length(x3));

for i=1:length(x3)

fx3(i) = p\*q^(x3(i)-1);

Fx3(i) = sum(fx3);

end

%%

set(gcf, 'Position', [200, 100, 900, 600])

%Построение графика функции плотности

subplot(2,3,1);

histogram(r1,'Normalization','probability','NumBins',max(r1));

grid on;

xlim([min(r1),max(r1)]);

%ylim([0,1]);

title('График функции плотности f(x)','fontsize',10);

xlabel('x'); ylabel('f(x)', 'rotation', 1);

subplot(2,3,2);

histogram(r2,'Normalization','probability','NumBins',max(r1));

grid on;

xlim([min(r2),max(r2)]);

%ylim([0,1]);

title('График функции плотности f(x)','fontsize',10);

xlabel('x'); ylabel('f(x)', 'rotation', 1);

subplot(2,3,3);

histogram(r3,'Normalization','probability','NumBins',max(r1));

grid on;

xlim([min(r3),max(r3)]);

%ylim([0,1]);

title('График функции плотности f(x)','fontsize',10);

xlabel('x'); ylabel('f(x)', 'rotation', 1);

%%

%Построение графика интегральной функции распределения

subplot(2,3,4);

stairs(sort(r1), (1:length(r1))/length(r1),'LineWidth',0.8,'MarkerFaceColor','c');

grid on;

xlim([min(r1),max(r1)]);

ylim([0,1]);

title('График интегральной функции распределения F(x)','fontsize',7);

xlabel('x'); ylabel('F(x)', 'rotation', 1);

subplot(2,3,5);

stairs(sort(r2), (1:length(r2))/length(r2),'LineWidth',0.8,'MarkerFaceColor','c');

grid on;

xlim([min(r2),max(r2)]);

ylim([0,1]);

title('График интегральной функции распределения F(x)','fontsize',7);

xlabel('x'); ylabel('F(x)', 'rotation', 1);

subplot(2,3,6);

stairs(sort(r3), (1:length(r3))/length(r3),'LineWidth',0.8,'MarkerFaceColor','c');

grid on;

xlim([min(x3),max(x3)]);

ylim([0,1]);

title('График интегральной функции распределения F(x)','fontsize',7);

xlabel('x'); ylabel('F(x)', 'rotation', 1);

%%

%Вычисление матожидания

M1 = sum(r1) / n;

M2 = sum(r2) / n;

M3 = sum(r3) / n;

%Вычисление дисперсии

D1 = sum(((r1 - M1).^2)/n);

D2 = sum(((r2 - M2).^2)/n);

D3 = sum(((r3 - M3).^2)/n);

T=table(['M';'D'],[M1;D1],[M2;D2],[M3;D3],[Mteor;Dteor]);

T.Properties.VariableNames = {'Eval' 'IRNGEO\_1' 'IRNGEO\_2' 'IRNGEO\_3' 'Teor\_value'};

Алгоритм 1

Используется кумулятивный метод моделирования, где границы интервалов, можно найти в соответствии с рекуррентной формулой:

p(0) = p

... ... ... ... ...

p(r) =p(r-1)\*(1-p), где r=1,2, ... .

function IR=IRNGEO\_1(p)

alpha = rand;

p\_it = p;

IR=0;

while ((alpha-p\_it)>=0)

alpha = alpha - p\_it;

p\_it = p\_it\*(1-p);

IR = IR + 1;

end

end

Алгоритм 2

Прямой метод заключается в получении псевдослучайной последовательности равномерно распределенных случайных чисел u[1], u[2],... в интервале [0,1], или иначе, так называемых "неудачных" исходов, до тех пор пока не найдется u[k] "успешный", который меньше или равен p.

function IR=IRNGEO\_2(p)

alpha = rand;

IR=0;

while (alpha > p)

alpha=rand;

IR=IR+1;

end

end

Алгоритм 3

Первый алгоритм, в основу которого положен кумулятивный метод, может быть усовершенствован следующим образом:

если положить p[1]+p[2]+...+p[k] = u,

тогда k = int[ln(u)/ln(q)]+1

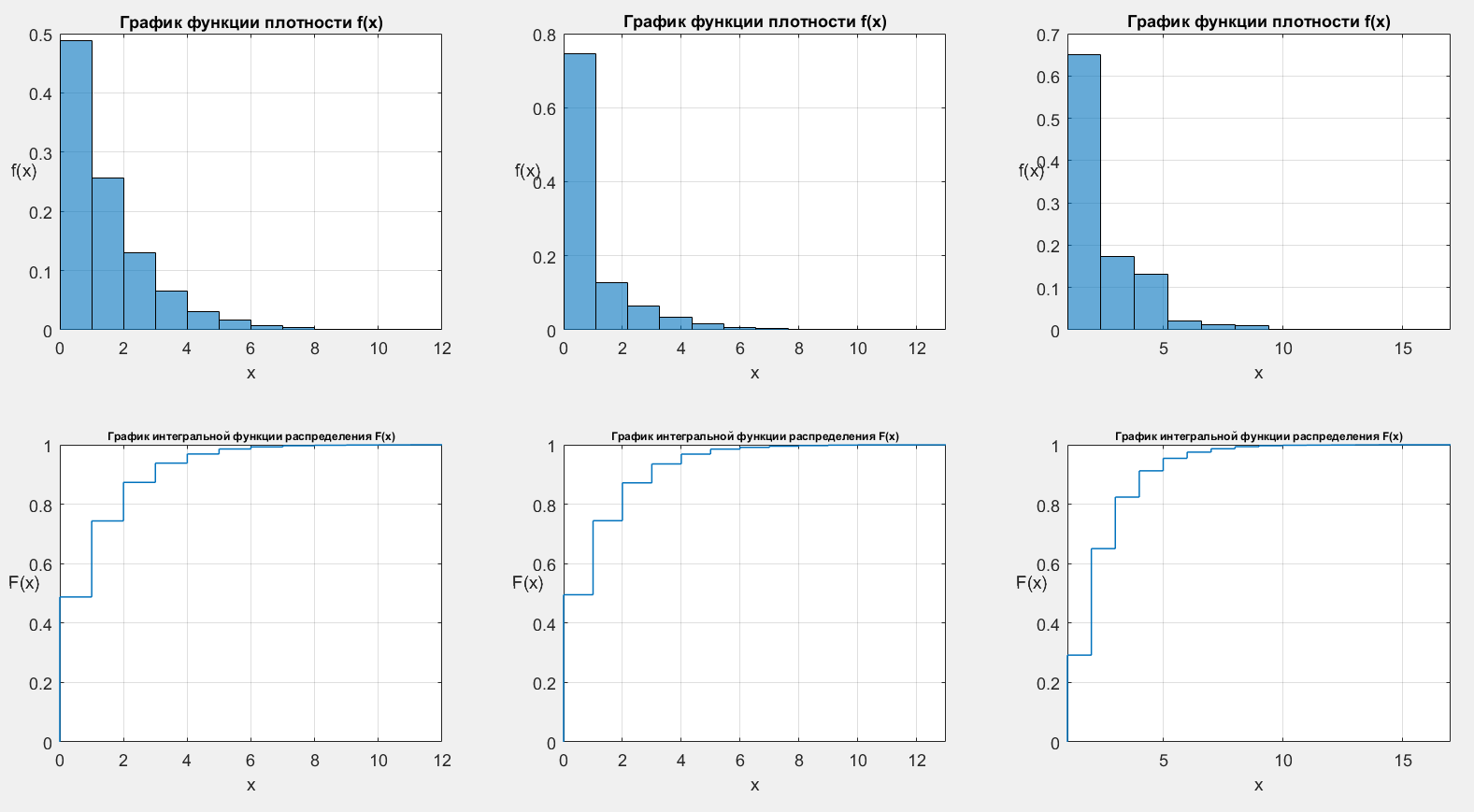
function IR=IRNGEO\_3(p)

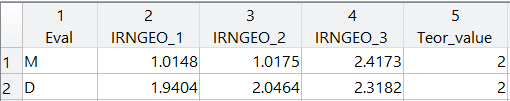
alpha=rand;

IR=round(log(alpha)/log(1-p))+1;

end

IR=IRNGEO(p) дает псевдослучайное число, имеющее геометрическое распределение. Если при обращении к IRNGEO() используются первые два алгоритма, следует помнить, что они крайне неэффекивны при небольших значениях параметра p, в то время как при использовании третьего алгоритма просматривается лишь слабая зависимость от выбранного значения этого параметра.





## **Распределение Пуассона**

Распределение Пуассона описывает число событий r, происходящих за одинаковые промежутки времени, при условии, что события происходят независимо друг от друга с постоянной интенсивностью. При этом число испытаний N велико, а вероятность появления события в каждом испытании мала.

clc

clearvars

syms mu n Mteor Dteor r;

mu=10;

n=10000;

Mteor=mu;

Dteor=mu;

r1=zeros(1,n);

r2=zeros(1,n);

for i=1:n

r1(i) = IRNPOI(mu);

r2(i) = IRNPSN(mu);

end

x1=unique(r1);

fx1=zeros(1,length(x1));

Fx1=zeros(1,length(x1));

for i=1:length(x1)

fx1(i) = (exp(-mu)\*(mu^x1(i)))/(factorial(x1(i)));

Fx1(i) = sum(fx1);

end

x2=unique(r2);

fx2=zeros(1,length(x2));

Fx2=zeros(1,length(x2));

for i=1:length(x2)

fx2(i) = (exp(-mu)\*(mu^x2(i)))/(factorial(x2(i)));

Fx2(i) = sum(fx2);

end

set(gcf, 'Position', [200, 100, 900, 600])

%Построение графика функции плотности

subplot(2,2,1);

histogram(r1,'Normalization','probability','NumBins',max(r1));

grid on;

xlim([min(r1),max(r1)]);

%ylim([0,1]);

title('График функции плотности f(x)','fontsize',10);

xlabel('x'); ylabel('f(x)', 'rotation', 1);

subplot(2,2,3);

histogram(r2,'Normalization','probability','NumBins',max(r1));

grid on;

xlim([min(r2),max(r2)]);

%ylim([0,1]);

title('График функции плотности f(x)','fontsize',10);

xlabel('x'); ylabel('f(x)', 'rotation', 1);

%%

%Построение графика интегральной функции распределения

subplot(2,2,2);

stairs(sort(r1), (1:length(r1))/length(r1),'LineWidth',0.8,'MarkerFaceColor','c');

grid on;

xlim([min(r1),max(r1)]);

ylim([0,1]);

title('График интегральной функции распределения F(x)','fontsize',10);

xlabel('x'); ylabel('F(x)', 'rotation', 1);

subplot(2,2,4);

stairs(sort(r2), (1:length(r2))/length(r2),'LineWidth',0.8,'MarkerFaceColor','c');

grid on;

xlim([min(r2),max(r2)]);

ylim([0,1]);

title('График интегральной функции распределения F(x)','fontsize',10);

xlabel('x'); ylabel('F(x)', 'rotation', 1);

%%

%Вычисление матожидания

M1 = sum(r1) / n;

M2 = sum(r2) / n;

%Вычисление дисперсии

D1 = sum(((r1 - M1).^2)/n);

D2 = sum(((r2 - M2).^2)/n);

T=table(['M';'D'],[M1;D1],[M2;D2],[Mteor;Dteor]);

T.Properties.VariableNames = {'Eval' 'IRNPOI' 'IRNPSN' 'Teor\_value'};

Алгоритм 1

Используется кумулятивный метод моделирования, где границы интервалов, можно найти в соответствии с рекуррентной формулой

p(0) = e \*\*(-mu)

... ... ... ... ... ... ... ... ... ...

P(r) = P(r-1)\*mu/r для r = 1,2,... ,

function IR=IRNPOI(mu)

if (mu<88)

alpha=rand;

p\_it=exp(-mu);

IR=1;

while ((alpha-p\_it)>=0)

alpha=alpha-p\_it;

p\_it=p\_it\*(mu/IR);

IR=IR+1;

end

else

IR=RNNORM(mu,mu);

end

end

Алгоритм 2

Альтернативный метод получения псевдослучайных чисел, имеющих распределение Пуассона, заключается в перемножении равномерно распределенных случайных чисел до тех пор, пока выполняется условие

Ro -mu

П x[i] >= e

i=0

function IR=IRNPSN(mu)

if (mu<88)

alpha=rand;

p\_it=alpha;

IR=1;

while (p\_it>=exp(-mu))

alpha=rand;

p\_it=p\_it\*alpha;

IR=IR+1;

end

else

IR=RNNORM(mu,mu);

end

end

Алгоритм 3

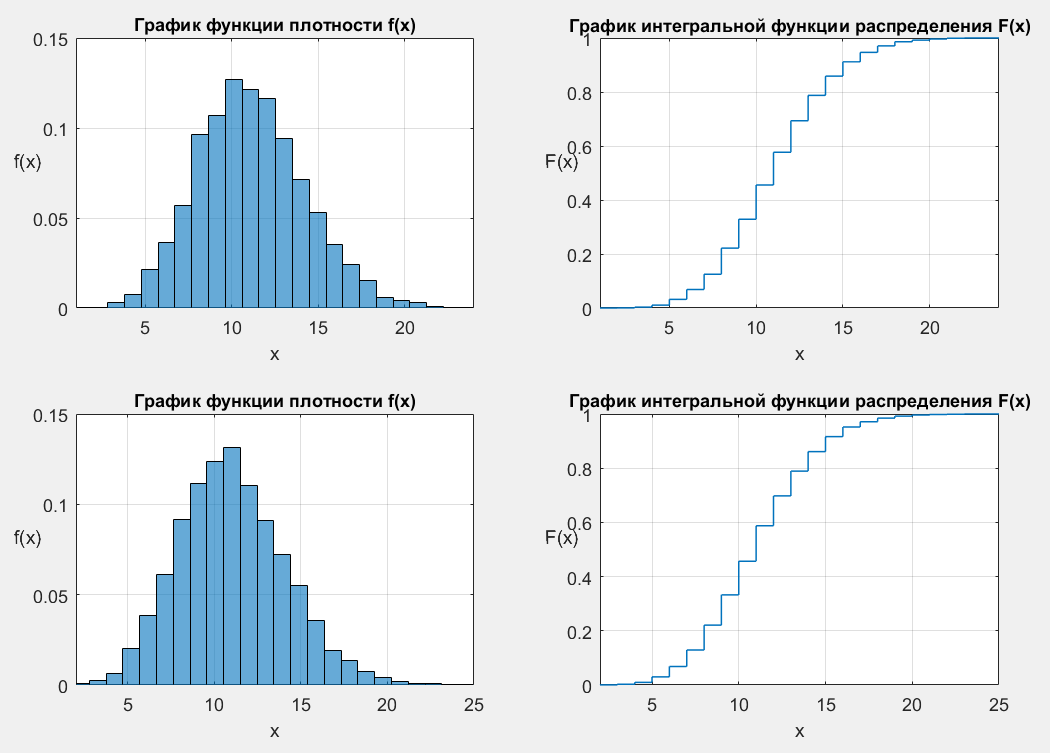
При больших значениях параметра mu возможна нормальная аппроксимация:

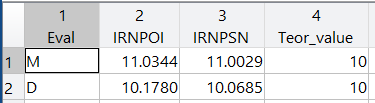
Ro = N(mu,mu)

function IR=RNNORM(mu,sigma)

IR = normrnd(mu, sigma);

end





## **Логарифмическое распределение**

clc

clearvars

q=0.5;

p=1-q;

n=10000;

alpha = 1/log(p);

Mteor=-alpha\*q/p;

Dteor=-alpha\*q\*(1+alpha\*q)/(p^2);

r=zeros(1,n);

for i=1:n

r(i)=IRNLOG(p);

end

x=unique(r);

fx=zeros(1,length(x));

Ffx=zeros(1,length(x));

for i=1:length(x)

fx(i)=-alpha\*(q^x(i))/x(i);

Fx(i)=sum(fx);

end

set(gcf, 'Position', [200, 100, 900, 600])

%Построение графика функции плотности

subplot(2,1,1);

histogram(r,'Normalization','probability','NumBins',max(r));

hold on

plot(x,fx,'ko-');

hold off

grid on;

xlim([min(r),max(r)]);

%ylim([0,1]);

title('График функции плотности f(x)','fontsize',10);

xlabel('x'); ylabel('f(x)', 'rotation', 1);

%Построение графика интегральной функции распределения

subplot(2,1,2);

stairs(sort(r), (1:length(r))/length(r),'LineWidth',0.8,'MarkerFaceColor','c');

hold on

plot(x,Fx,'ko-');

hold off

grid on;

xlim([min(r),max(r)]);

ylim([0,1]);

title('График интегральной функции распределения F(x)','fontsize',15);

xlabel('x'); ylabel('F(x)', 'rotation', 1);

Алгоритм 1.

Используется стандартный кумулятивный метод.

function IR=IRNLOG(p)

alpha=rand;

p\_it= -(1/log(p))\*(1-p);

IR=1;

while ((alpha-p\_it)>=0)

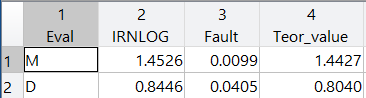
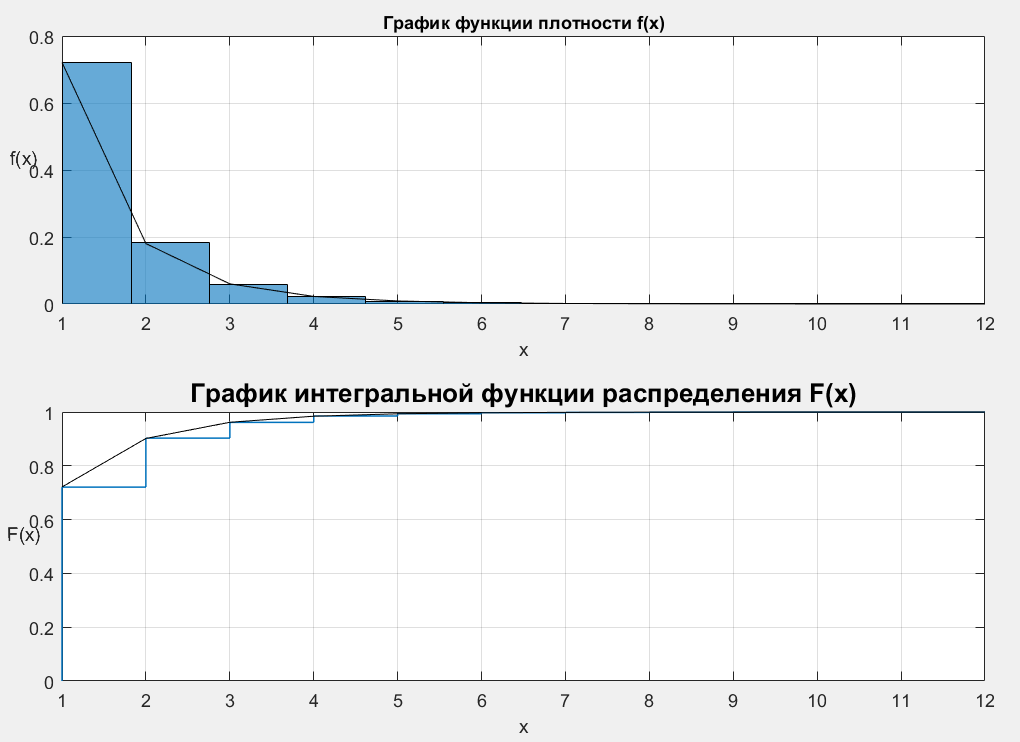
alpha=alpha-p\_it;

p\_it=p\_it\*(IR/(IR+1))\*(1-p);

IR=IR+1;

end

end



## **Вывод**

В ходе выполнения лабораторной работы были разработаны программные датчики случайных величин. Исследование характеристик моделируемых датчиков показало, что точность моделирования высока, так как вычисленные значения математического ожидания и дисперсии совпали с теоретическими значениями лишь с небольшой погрешностью. К каждой программе представлены графики функции плотности распределения и интегральной функции распределения.