Министерство науки и образования Украины

Харьковский Национальный Университет им. В.Н.Каразина

Факультет компьютерных наук

Кафедра моделирования систем и технологий

# Разработка программного продукта Delphi для моделирования логнормального распределения

**Курсовая работа по дисциплине**

***«Языки прикладного программирования»***

Исполнитель

студент \*\*\*\*\*\*

Руководитель

Ст. преп

Харьков 2007

***План***

1. Введение
2. Проектирование
3. Особенности реализации
4. Отладка и тестирование
5. Описание работы программного продукта
6. Заключение
7. Список используемой литературы

**Введение**

В данной работе рассматривается логнормальное распределение, его связь с другими распределениями. В статистике так называемое логнормальное распределение применяется в том случае, когда начинает изменяться цена актива в будущем – а это случайный процесс, который в принципе должен описываться нормальным распределением. В то же время для целей вероятностной оценки стоимости актива в теории пользуются не нормальным, а логнормальным распределением.

Это обусловлено следующими причинами. Во-первых, нормальное распределение симметрично относительно ее центральной оси и может иметь как положительные, так и отрицательные значения; однако цена актива не может быть отрицательной. Во-вторых, нормальное распределение говорит о равной вероятности для значений переменной отклониться вверх или вниз. В то же время на практике, например, имеет место инфляция, которая оказывает давление на цены в сторону их повышения, а также сама временная сущность денег: стоимость денег сегодня меньше, чем стоимость денег вчера, но больше, чем стоимость денег завтра. Кривая логнормального распределения всегда положительна и имеет правостороннюю скошенность (асимметрично), т.е. она указывает на большую вероятность цены отклониться вверх. Поэтому если, допустим, цена актива составляет 50 долл., то кривая логнормального распределения свидетельствует о том, что опцион пут с ценой исполнения 45 долл. должен стоить меньше опциона колл с ценой исполнения 55 долл., в то время как в соответствии с нормальным распределением они должны были бы иметь одинаковую цену. Хотя нельзя надеяться, что приведенные исходные предположения в точности выполняются во всех реальных рыночных ситуациях, тем не менее принято считать, что логнормальное распределение достаточно хорошо как первое приближение в случае активов, которыми торгуют на конкурентных рынках аукционного типа для длинных рассматриваемых периодов.

**Проектирование**

Перед началом работы в среде Delphi мною, я разработал макет программного продукта в письменном варианте, где я зарисовал какая должна быть главная форма (внешний вид ее), сколько и какие компоненты должны быть на этой форме для удобной работы пользователя с данным продуктом. Также на этом макете я разработал план создания текста программы.

После того, как была проделана выше указанная работа, я перешел непосредственно к созданию программного продукта на компьютере. Сначала я создал главную форму, соответствующую макету (Рис.1). На ней находятся:

* 2 колонки выводов значений: Теоретически, Критерий согласия;
* в Теоретической колонке: sigma, mu, a, b;
* в Критерии согласия: метод Неймана и метод обратных функций;
* поля для вывода мат.ожидания и дисперсии;
* кнопки управления программой и режимом просмотра;
* меню “Help” которое содержит подменю “About me” и “About the program”;
* поле время выполнения;
* A также кнопки “Вывести графики и вычислить” при нажатии которой программа считает все значения и выводит график на экран, “Выход”, для выхода из программы.
* A также кнопка Stop при нажатии которой программа считает значения, которые обработались до определенного момента.

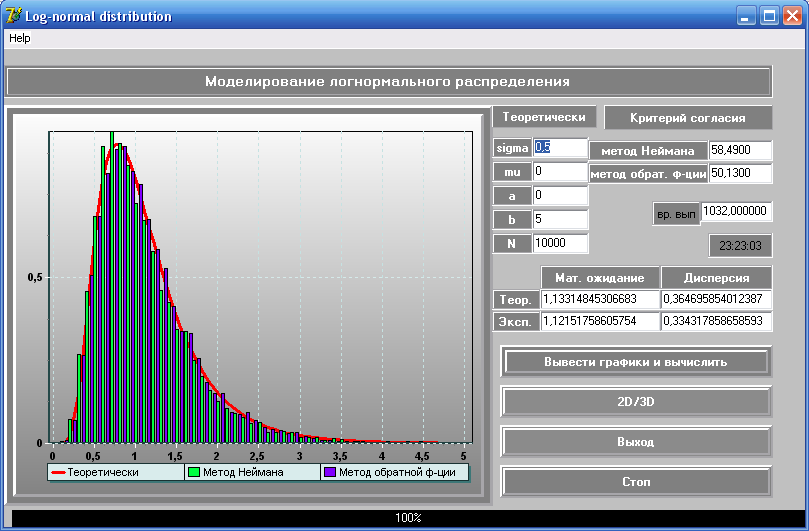


Рис. 1

**Особенности реализации**

var

Form1: TForm1;

kk:Int64;

flag:boolean;

implementation

Плотность распределения

function TForm1.PL(x:double):double; **//--density of distribution**

begin

if x<>0 then

result := exp(-(ln(x)-mu)\*(ln(x)-mu)/(2\*sigma\*sigma))/(x\*sigma\*Sqrt(2\*Pi))

else

result := 0;

end;

function TForm1.LogNorm() : double; **//--for a method of Neumann**

var

y : real;

x : double;

begin

repeat

x := a+random\*(b-a);

f := PL(x);

y := fmax\*random;

until y<f;

result := x;

end;

procedure TForm1.Clear; **//------------clear array---------**

const M=50;

var j : integer;

begin

for j:=0 to (M-1) do

begin

gist[j] := 0;

end;

end;

procedure TForm1.Panel1Click(Sender: TObject);

var

x, r, sr, h1, h2, Ob,g1,g2, chi2\_N, chi2\_12, chi2\_if, sum, Z : double;

p, y, Mat, Mat2, Disp : real;

M, j : integer;

N, i, u : longint;

begin

flag:=false;

Gauge1.Progress:=0;

**//-------\*\*All fields must be filled!\*\*---------**

if (E1.Text='') or (E2.Text='') or (E3.Text='') or (E4.Text='') or

(E5.Text='') then

begin

with Application do

begin

NormalizeTopMosts;

MessageBox('All of fields must be filled!', 'Error', MB\_OK);

RestoreTopMosts;

end;

exit;

end;

**//----------\*\*initialization\*\*--------------**

T := GetTime;

Clear;

Chart1.Series[0].Clear;

Chart1.Series[1].Clear;

Chart1.Series[2].Clear;

sigma := StrToFloat(E1.Text);

mu := StrToFloat(E2.Text);

a := StrToFloat(E3.Text);

b := StrToFloat(E4.Text);

kk:=StrToint64(E5.Text);

if kk>2000000000 then

begin

Showmessage ('Очень большое число, введите меньшее');

exit;

end;

N := StrToInt(E5.Text);

g1:=100/N;

g2:=0;

Randomize;

M := 50;

**//---------------\*\*theoretical method\*\*------------------**

for i:=1 to 100 do

begin

if (i mod 10) =0 then application.ProcessMessages;

x := a+i\*(b-a)/100;

//p := PL(x);

if x<>0 then

p := exp(-(ln(x)-mu)\*(ln(x)-mu)/(2\*sigma\*sigma))/(x\*sigma\*Sqrt(2\*Pi))

else

p := 0;

Chart1.Series[0].AddXY(x, p);

end; //---theoretical

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Метод Неймана

**//---------------\*\*method of Neumann\*\*------------------------**

fmax :=Chart1.Series[0].MaxYValue;

{for i:=1 to N do

begin

if (i mod 10) =0 then application.ProcessMessages;

x := a+i\*(b-a)/N;

f := PL(x);

if (f>fmax)then

fmax := f;

end;} //max

//------------------------------

Clear;

chi2\_N:=0;

Mat:=0;

Mat2:=0;

Disp:=0;

i:=0;

Clear;

chi2\_if := 0;

while true do

begin

if (i mod 10) =0 then application.ProcessMessages;

inc(i);

x := LogNorm();

Mat := Mat+x; **//expectation**

Mat2 := Mat2 +sqr(x);

if (x>b) or (x<a) then

continue;

u := trunc((x-a)/((b-a)/M));

gist[u] := gist[u]+1;

h1 := random;

h2 := random;

Ob := sqrt(-2\*ln(h1))\*cos(2\*Pi\*h2);

Ob := mu+Ob\*sigma;

x := exp(Ob);

if (x>b) or (x<a) then

continue;

u := trunc((x-a)/((b-a)/M));

gist1[u] := gist1[u]+1;

g2:=g2+g1;

Gauge1.Progress:=trunc(g2)+1;

if i>N then break;

if flag=true then

begin

N:=i;

break;

end;

end;

Mat := Mat/N;

Mat2 := Mat2/N;

Disp := Mat2 - sqr(Mat);

for j:=0 to (M-1) do **//------histogram**

begin

sum := (Power(N\*PL(a+(b-a)/M\*(j+0.5))\*(b-a)/M-gist[j], 2))/

(N\*PL(a+(b-a)/M\*(j+0.5))\*(b-a)/M);

chi2\_N := chi2\_N+sum;

Chart1.Series[1].AddXY((a+(j+0.5)\*(b-a)/M), gist[j]/N\*M/(b-a));

end;

E6.Text := FloatToStrF(chi2\_N, fffixed, 4, 4);**//--chi-square for a Neumann** //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Метод обратной функции

**//--------------\*\*method of inverse function\*\*-----------------**

Clear;

chi2\_if := 0;

{for i:=1 to N do

begin

h1 := random;

h2 := random;

Ob := sqrt(-2\*ln(h1))\*cos(2\*Pi\*h2);

Ob := mu+Ob\*sigma;

x := exp(Ob);

if (x>b) or (x<a) then

continue;

u := trunc((x-a)/((b-a)/M));

gist[u] := gist[u]+1;

end;}

for j:=0 to (M-1) do //------histogram

begin

sum := (Power(N\*PL(a+(b-a)/M\*(j+0.5))\*(b-a)/M-gist1[j], 2))/

(N\*PL(a+(b-a)/M\*(j+0.5))\*(b-a)/M);

chi2\_if := chi2\_if+sum;

Chart1.Series[2].AddXY((a+(j+0.5)\*(b-a)/M), gist1[j]/N\*M/(b-a));

gist1[j]:=0;

end;

E8.Text := FloatToStrF(chi2\_if, fffixed, 4, 4); **//chi-sq for a inverse function**

E10.Text := FloatToStr(exp(mu+sqr(sigma)/2)); **//--expectation (teor)**

E11.Text := FloatToStr(Mat); **//--expectation (experim)**

E12.Text := FloatToStr((exp(sqr(sigma))-1)\*exp(2\*mu+sqr(sigma)));

E13.Text := FloatToStr(Disp);

D := GetTime;

Z := MilliSecondSpan(D, T);

e5.Text:=IntTostr(N);

Edit1.Text := FloatToStrF(Z, fffixed, 6, 6);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

end;

procedure TForm1.Panel7Click(Sender: TObject);

begin

Close;

end;

procedure TForm1.E1KeyPress(Sender: TObject; var Key: Char);

begin

if (key='-')

then begin

if Pos ('-', (Sender as TEdit).Text)=0 then Begin (Sender as TEdit).SelStart:=0; key:='-'; end

else key:=#0;

end;

if Sender is TEdit then

begin

if Not((Key in ['0'..'9'])or (Key=Chr(vk\_Back))

or (Key=DecimalSeparator) or (Key='-')) then

Key:=#0

else

begin

if Key = DecimalSeparator then

if Pos(DecimalSeparator,(Sender as TEdit).Text)>0 then

Key:=#0;

end;

end;

end;

procedure TForm1.Aboutme1Click(Sender: TObject);

begin

AboutBox.Show;

end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin

Panel19.Caption := TimeToStr(Time);

end;

procedure TForm1.E1KeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (ssShift in Shift)then

key:=0;

end;

procedure TForm1.Panel20Click(Sender: TObject);

begin

flag:=true;

end;

end.

**Отладка и тестирование программы**

В процессе отладки я вводил различные значения успешной вероятности и количество успехов и сравнивал форму полученного графика при разных значениях.

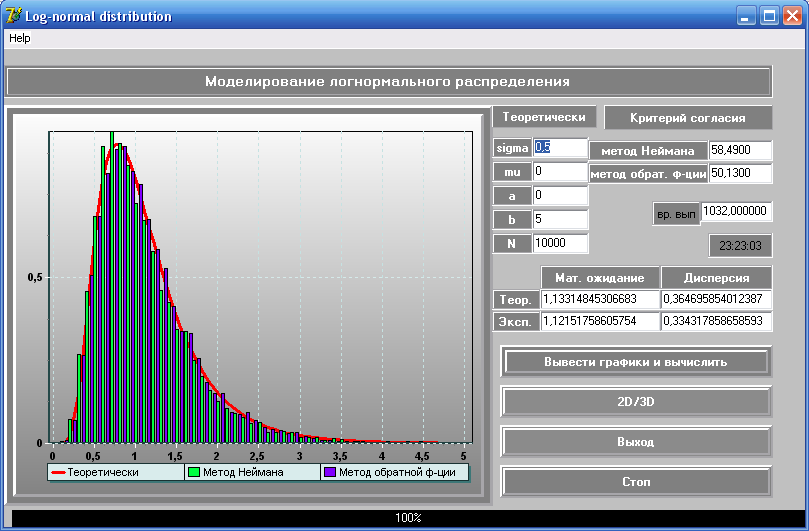


Рис. 2

Так, же пробовал вводить другие значения, график при этом не сильно менялся

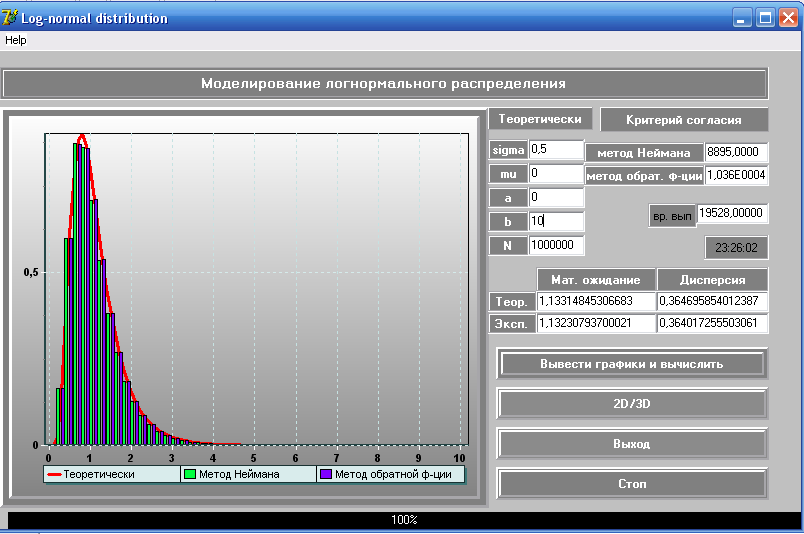


Рис. 3

Пользователь может увидеть полученные графики в трехмерном и в двухмерном пространстве: для этого надо нажать кнопку 2D\3D. Результат 3D графика можно увидеть на рис. 3

Функция для 3D\2D записана так

procedure TForm1.Panel12Click(Sender: TObject);

begin

Chart1.View3D:=Not Chart1.View3D;

end;

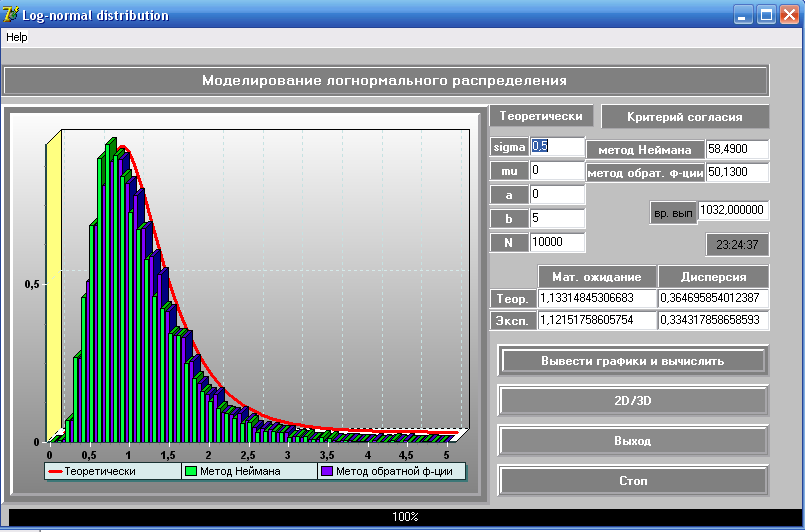


Рис. 4

**Описание работы программного продукта**

При запуске программы перед пользователем открывается форма, на которой есть поля ввода параметров, поля вывода посчитанных значений, поле для вывода графика и кнопки, при нажатии на которые происходит то или иное событие.

Справа в разделе "Теоретически пользователь может ввести значение *sigma* и *mu*, те значения которые он считает нужными; *a* и *b* это интервал в пределах которого меняется график. И значение *N* – (количество єксперементов) – в зависемости от того сколько раз мы будем проводить єксперемент . В зависимости от выбора данных параметров пользователь может получить различные формы графика плотности вероятности.

В разделе "Критерий согласия" выводятся значения оценки Хи-квадрат для двух указанных методов. Ниже вывод математического ожидания и дисперсии, посчитанных теоретически и экспериментально.

Справа внизу формы выводится системное время и время выполнения расчётов в миллисекундах.

При нажатии на кнопку «Вывести графики и вычислить» слева выводятся график плотности логнормального распределения (построенный теоретически), гистограммы распределения случайной величины по логнормальному закону, смоделированные при помощи метода Неймана и метода обратной функции.

При нажатии на кнопку «Стоп» программа прекращаются свою работу и начинает считывать значения которые обработались до определенного момента и записует значения в поля.

При нажатии на кнопку «2D/3D» пользователь может наблюдать изменение графика из 2D в 3D и наоборот.

При нажатии на кнопку «Выход» программа будет завершена.

В закладке «About» пользователь может узнать о создателях данного программного продукта и краткое описание программного продукта.

**Заключение**

В данной курсовой работе была достигнута поставленная цель: я изучил и создал программный продукт, который представляет моделирование на компьютере логнормального распределения. Научился применять на практике свои знания полученные в процессе изучения Delphi.

В данном программном продукте реализованы работа с графиками, с таблицами, таймерами, файлами, различными математическими функциями.

Этот программный продукт, на мой взгляд, представляет собой законченную рабочую и отлаженную программу.

**Список используемой литературы**

1. <http://en.wikipedia.org>
2. Шеффе Г. Дисперсионный анализ. - М.: Физматгиз, 1980. - 628 с.
3. «Delphi 2005: «Секреты программирования»», Михаил Фленов.