Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО “РГРТУ”

Кафедра вычислительной и прикладной математики

Лабораторная работа №3
“Генерирование случайных величин с заданным законом распределения”

Выполнил:
Студент группы №843
Редько С.В.

Проверил:
Овечкин Г.В.

Рязань 2022

Вариант 10

Задание:

Составить подпрограмму генерирования случайных величин в соответствии с вариантом задания. По полученной с помощью подпрограммы выборке построить и проанализировать гистограмму частот и статистическую функцию распределения, оценить матожидание и дисперсию случайной величины. Соответствие эмпирических данных теоретическому распределению проверить с помощью критерия Пирсона или критерия Колмогорова. Объем выборки случайных величин не менее 1000. Количество интервалов разбиения k = 15 или k = 25.

№ вар.	Закон распределения	Способ построения
10.	$F(x)=\begin{cases}0.8x^2, & x\in[0;0.5);\\ 0.7x-0.15, & x\in[0.5;1);\\ 1-e^{-0.8x}, & x\in[1;\infty).\end{cases}$	Метод обратных функций

Решение:

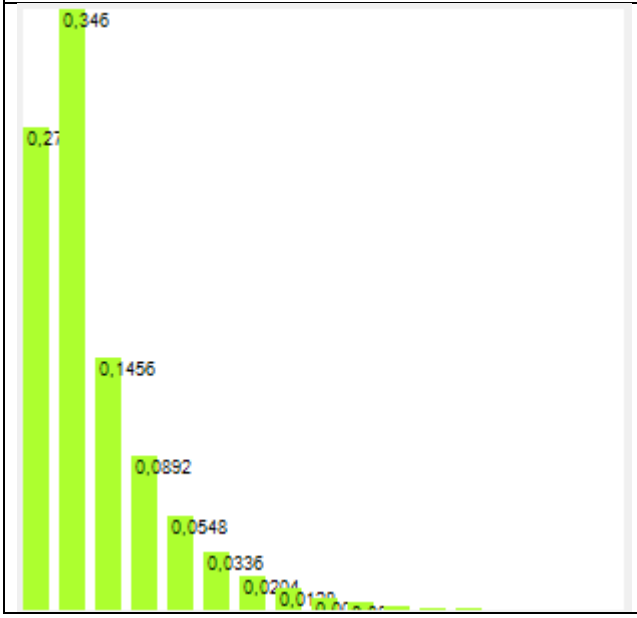
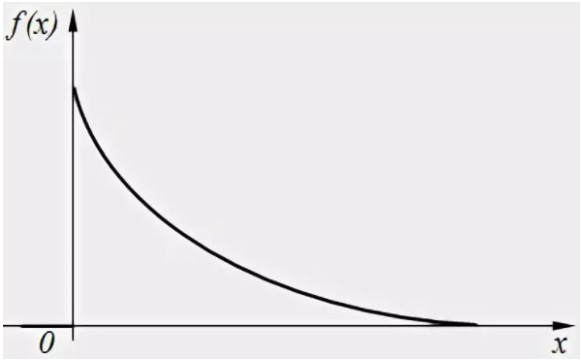
Для каждого подынтервала решим уравнение $r = F(x)$ относительно переменной x. Получим:

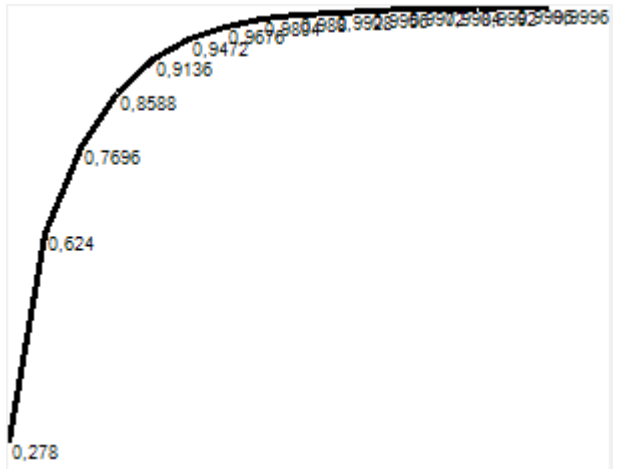
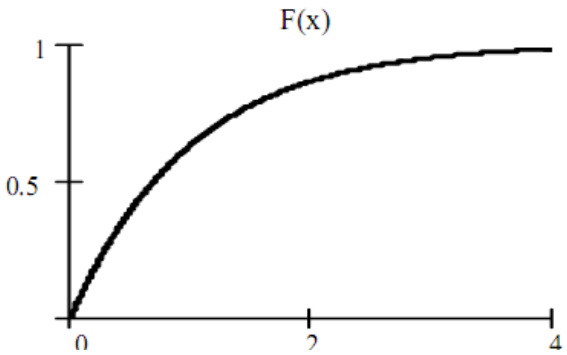
$$x=\begin{cases}\sqrt{\dfrac{r}{0.8}}, & 0\le r<0.2\\ \dfrac{r+0.15}{0.7}, & 0.2\le r<0.55\\ \dfrac{\ln(1-r)}{0.8}, & 0.55\le r<1\end{cases}$$

После генерации последовательности в 2500 значений оцениваем ее характеристики:

	Полученное	Параметр распределения
Математическое ожидание	0,1369	7,3046
Дисперсия	0,0145	8,3046

Построим гистограмму частот и статистическую функцию распределения:

Гистограмма частот	Предполагаемая функция плотности вероятности
	

Статистическая функция распределения	Предполагаемая функция распределения
	

Критерий Пирсона:

Требуется проверить, действительно ли случайная величина X имеет экспоненциальный закон распределения.

Количество интервалов	15
Уровень значимости	0,05
Число степеней свободы	13
Мера расхождения Хи^2	95,78
Критическое значение Хи^2	5,89

Так как мера расхождения Хи^2 **(95,78)** больше критического значения Хи^2 **(10,12)**, то гипотеза отвергается.

Количество интервалов	25
Уровень значимости	0,05
Число степеней свободы	
Мера расхождения $Хи^2$	167,14
Критическое значение $Хи^2$	13,11

Так как мера расхождения $Хи^2$ **(167,14)** больше критического значения $Хи^2$ **(13,11),** то гипотеза отвергается.

Количество интервалов	9
Уровень значимости	0,05
Число степеней свободы	7
Мера расхождения $Хи^2$	1,47
Критическое значение $Хи^2$	2,17

Так как мера расхождения $Хи^2$ **(1,47)** меньше критического значения $Хи^2$ **(2,17),** то гипотеза принимается.

Критерий Колмогорова:

Требуется проверить, действительно ли случайная величина X имеет экспоненциальный закон распределения.

Длина выборки	1000
D+ (max)	0,025
D- (max)	0,133
D (max)	0,133
Уровень значимости (расч)	4,194
Вероятность	<<0,001

Так как рассчитанный уровень значимости **(4,194)** больше заданного уровня вероятности **(0,5)**, то гипотеза **отвергается**.

Длина выборки	100
D+ (max)	0,025
D- (max)	0,138
D (max)	0,138
Уровень значимости (расч)	1,381
Вероятность	0,044

Так как рассчитанный уровень значимости **(1,381)** больше заданного уровня вероятности **(0,5)**, то гипотеза **отвергается**.

Приложение:

```
private void buttonGenerate_Click(object sender, EventArgs e)
    {
      switch (comboBoxDistribution.SelectedItem)
      {
        case "Какой-то распределение":
          {
            for (int i = 0; i < CountNumbers; i++)
            {
              double Fx = (double)i / CountNumbers;
              SequenceNorm.Add(Formuls.X(Fx));
              if (i < 50)
                listNumber.Text += SequenceNorm[i].ToString() + "\n";
            }
            for (int i = 0; i < CountNumbers; i++)
            {
              Sequence.Add(SequenceNorm[i] / SequenceNorm.Max());
            }
            break;
          }
        default:
          break;
      }
      //Рассчет статистических данных
      Mx = Formuls.GetMx(Sequence);
      Dx = Formuls.GetD(Sequence);
      Moment2 = Formuls.Get2Moment(Sequence);
      Moment3 = Formuls.Get3Moment(Sequence);

      textBoxMx.Text = Mx.ToString();
      textBoxD.Text = Dx.ToString();
      textBox2Moment.Text = Moment2.ToString();
      textBox3Moment.Text = Moment3.ToString();

      //Тест серии единиц
      TestLengthSeria();

      SequenceSort = Sequence;
      SequenceSort.Sort();

      double min = SequenceSort.Min();
      double max = SequenceSort.Max();
      double lengthPart = (max - min) / CountIntervals;
      //Построение гистограммы частот
      for (int i = 0; i < CountIntervals; i++)
      {
        ProbabilityDensity.Add(0);
        for (int j = 0; j < SequenceSort.Count(); j++)
          if (Sequence[j] >= min + lengthPart * i &&
                  Sequence[j] < min + lengthPart * (i + 1))
            ProbabilityDensity[i]++;
      }
      for (int i = 0; i < CountIntervals; i++)
        ProbabilityDensityNorm.Add((double)ProbabilityDensity[i]             /
CountNumbers);
      DrawHistogramm(ProbabilityDensityNorm, CountIntervals);
      //Построение статистической функции распределения
      DistributionFunction.Add(ProbabilityDensity[0]);
      for (int i = 1; i < CountIntervals; i++)
        DistributionFunction.Add(DistributionFunction[i      -      1]      +
ProbabilityDensity[i]);
      for (int i = 0; i < CountIntervals; i++)
```

```
        DistributionFunctionNorm.Add((double)DistributionFunction[i]        /
CountNumbers);

      DrawGraph(DistributionFunctionNorm, CountIntervals);

      //Проверка по критерию Пирсона
      CheckPirson();
      //Проверка по критерию Колмогорова
      CheckKolmogorov();

    }
    /// <summary>
    /// Гистограмма
    /// </summary>
    /// <param name="parArr"></param>
    private void DrawHistogramm(List<double> parList, int parCount)
    {
      int intervalColumn = 5;
      int widthColumn = panel1.Width / parCount - intervalColumn;

      Graphics gPanel = panel1.CreateGraphics();

      gPanel.FillRectangle(new SolidBrush(Color.White),
              new Rectangle(0, 0, panel1.Width, panel1.Height));

      for (int i = 0; i < parCount; i++)
      {
        gPanel.FillRectangle(new SolidBrush(Color.GreenYellow),
            new Rectangle((widthColumn + intervalColumn) * i,
                          panel1.Height       -       (int)(parList[i]       *
(double)panel1.Height / parList.Max()),
                          widthColumn,
                          panel1.Height));
        gPanel.DrawString(parList[i].ToString(),
            new Font("Arial", 7),
            new SolidBrush(Color.Black),
            (widthColumn + intervalColumn) * i,
            panel1.Height   -   (int)(parList[i]   *   (double)panel1.Height   /
parList.Max()));
      }
    }
    /// <summary>
    /// График
    /// </summary>
    /// <param name="parArr"></param>
    private void DrawGraph(List<double> parList, int Count)
    {
      int intervalColumn = 5;
      int widthColumn = panel1.Width / Count - intervalColumn;

      Graphics gPanel = panel2.CreateGraphics();
      gPanel.FillRectangle(new SolidBrush(Color.White),
              new Rectangle(0, 0, panel2.Width, panel2.Height));

      for (int i = 0; i < Count; i++)
      {
        if (i != 0)
          gPanel.DrawLine(new Pen(Color.Black, 3),
              new PointF((widthColumn + intervalColumn) * i, panel1.Height -
(int)(parList[i] * panel1.Height / parList.Max())),
              new   PointF((widthColumn   +   intervalColumn)   *   (i   -   1),
panel1.Height - (int)(parList[i - 1] * panel1.Height / parList.Max())));
        gPanel.DrawString(parList[i].ToString(),
            new Font("Arial", 7),
```

```
            new SolidBrush(Color.Black),
            (widthColumn + intervalColumn) * i,
            panel1.Height  -  (int)(parList[i]  *  (double)panel1.Height  /
parList.Max()));
      }
    }
    /// <summary>
    /// Проверка по критерию Пирсона
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    public void CheckPirson()
    {
      //Рассчитанный хи квадрат
      double hi2 = 0;
      //число степеней свободы
      int NumberOfDegreesOfFreedom = 0;
      switch (comboBoxDistribution.SelectedItem)
      {
        case "Какой-то распределение":
          {
            double lambda1 = 1.0 / Mx;
            double lambda2 = Math.Sqrt(1.0 / Dx);
            double lambda = (lambda1 + lambda2) / 2;
            for (double i = 0; i < 1; i += 1.0 / CountIntervals)
              ExponentialProbabilityDensity.Add(lambda  *  Math.Exp(-lambda  *
i));

            double sum = ExponentialProbabilityDensity.Sum();
            for (int i = 0; i < CountIntervals; i++)
              ExponentialProbabilityDensity[i] /= sum;

            //DrawGraph(ExponentialProbabilityDensity, CountIntervals);

            hi2 = Formuls.GetHi2Exponential(ProbabilityDensity, CountNumbers,
ExponentialProbabilityDensity);
            NumberOfDegreesOfFreedom                                          =
Formuls.GetNumberOfDegreesOfFreedom(CountIntervals, 1);
            break;
          }
        default:
          break;
      }
      textBoxHi2Nabl.Text = hi2.ToString();
      textBoxNumberOfDegreesOfFreedom.Text                                    =
NumberOfDegreesOfFreedom.ToString();
      checkPirson.BackColor  =  hi2  <  (double)numericUpDownHi2Tabl.Value  ?
Color.GreenYellow : Color.Red;
    }
    /// <summary>
    /// Проверка по критерию Колмогорова
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    public void CheckKolmogorov()
    {
      switch (comboBoxDistribution.SelectedItem)
      {
        case "Какой-то распределение":
          {
            for (int i = 1; i <= CountNumbers; i++)
            {
              KolmogorovPlus.Add((double)i  /  CountNumbers  -  (1  -  Math.Exp(-
SequenceSort[i - 1] / Mx)));
```

```
              KolmogorovMinus.Add(1 - Math.Exp(-SequenceSort[i - 1] / Mx) -
(double)(i - 1) / CountNumbers);
            }
            break;
          }
        default:
          break;
      }

      double Kolmogorov = Math.Max(KolmogorovPlus.Max(),
KolmogorovMinus.Max());
      double alpha = (double)numericUpDownLvlZna4.Value;
      double alphaCalc;
      alphaCalc = Kolmogorov * Math.Sqrt(CountNumbers);


      textBoxDnPlus.Text = KolmogorovPlus.Max().ToString();
      textBoxDnMinus.Text = KolmogorovMinus.Max().ToString();
      textBoxDn.Text = Kolmogorov.ToString();
      textBox4dn.Text = alphaCalc.ToString();

      checkKolmogorov.BackColor = alphaCalc <= alpha ? Color.GreenYellow :
Color.Red;
    }

    public static double GetMx(List<double> parList)
    {
      return parList.Sum() / parList.Count();
    }
    public static double GetD(List<double> parList)
    {
      double sum = 0;
      double N = parList.Count();
      for (int i = 0; i < N; i++)
        sum += Math.Pow(parList[i] - GetMx(parList), 2);
      return sum / N;
    }
    public static double GetHi2Exponential(List<int> parM, int
parCountNumbers, List<double> parP)
    {
      double sum = 0;
      double a;
      for (int i = 0; i < parM.Count(); i++)
      {
        a = parCountNumbers * parP[i];
        sum += (double)((parM[i] - a) * (parM[i] - a)) / a;
      }
      return sum;
    }
    public static int GetNumberOfDegreesOfFreedom(int parK, int parM)
    {
      return parK - parM - 1;
    }
```