Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО РГРТУ

Кафедра вычислительной и прикладной математики

Лабораторная работа №2

“Проверка качества генераторов псевдослучайных чисел”

Выполнил:

Студент группы №843

Редько С.В.

Проверил:

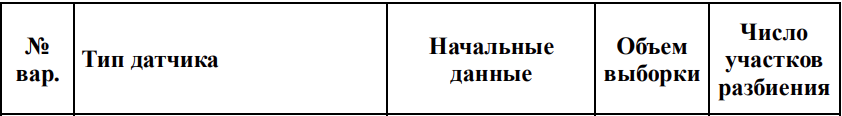
Овечкин Г.В.

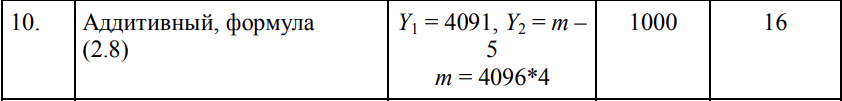
Рязань 2022

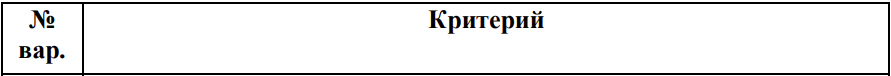
**Вариант 10**

**Задание:**

Составить и отладить программу (подпрограмму) генерирования псевдослучайных чисел с равновероятным распределением на интервале [0;1). Используя полученные результаты, проверить качество последовательности с помощью критерия Пирсона, Колмогорова, а также теста длины серии единиц.









**Решение:**

1. **Критерий Пирсона:**

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

Пусть Количество интервалов – 20; Уровень значимости – 0,05

Тогда мера расхождения Хи2 = 10,78; Число степеней свободы = 19;

Критическое значение Хи2 = 10,12

|  |  |
| --- | --- |
| Количество интервалов | 20 |
| Уровень значимости | 0,05 |
| Число степеней свободы | 19 |
| Мера расхождения Хи2 | 10,78 |
| Критическое значение Хи2 | 10,12 |

Так как мера расхождения Хи2 **(10,78)** больше критического значения Хи2 **(10,12)**, то гипотеза **отвергается**.

Если возьмем количество интервалов, равным 22, то:

|  |  |
| --- | --- |
| Количество интервалов | 22 |
| Уровень значимости | 0,05 |
| Число степеней свободы | 21 |
| Мера расхождения Хи2 | 8,918 |
| Критическое значение Хи2 | 11,92 |

Так как мера расхождения Хи2 **(8,918)** меньше критического значения Хи2 **(11,92)**, то гипотеза **принимается**.

1. **Критерий Колмогорова:**

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

|  |  |
| --- | --- |
| Уровень значимости | 0,5 |
| D+ (max) | 0,01398 |
| D- (max) | 0,01518 |
| D (max) | 0,01518 |
| Вероятность | 0,964 |

Так как статистика критерия D **(0,01518)** меньше вероятности **(0,964)**, гипотеза **принимается**.

1. **Критерий серии единиц:**

Требуется проверить независимость случайных величин равномерного закона распределения.

При разделительном элементе p = 0,25 получаем:

|  |  |
| --- | --- |
| Число единиц | 753 |
| Число серий единиц | 189 |
| Математическое ожидание числа единиц в серии | 3,0486 |
| Дисперсия числа единиц в серии | 12,3424 |
| Оценка средней длины серии единиц | 3,9841 |
| Надежность | 0,97 |
| Уровень значимости | 0,03 |
| Квантиль стандартного нормального распределения | 2,18 |
| Нижнее критическое значение | 2,4915 |
| Верхнее критическое значение | 3,6057 |

Так как оценка средней длины серии единиц **(3,9841)** не находится между нижним **(2,4915)** и верхним **(3,6057)** критическими значениями то гипотеза **отвергается**.

**Приложение:**

/// <summary>

/// Генерация псевдослучайных чисел с равновероятным распределением

/// </summary>

private void buttonGenerate\_Click(object sender, EventArgs e)

{

listNumber.Text = "";

numericUpDownCountIntervals.Enabled = true;

CountNumbers = (int)numericUpDownN.Value;

CountIntervals = (int)numericUpDownCountIntervals.Value;

switch (comboBoxDistribution.SelectedItem)

{

case "Равновероятное распределение":

{

Y0 = (int)numericUpDownY1.Value;

M = (int)numericUpDownM.Value;

Y1 = (int)numericUpDownY2.Value;

for (int i = 0; i < CountNumbers; i++)

{

Sequence.Add(RandomAdditive());

if (i < 50)

listNumber.Text += Sequence[i].ToString() + "\n";

}

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 = Formuls.Get2Moment(Sequence).ToString();

textBox3Moment.Text = Formuls.Get3Moment(Sequence).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>

/// Проверка по критерию Пирсона

/// </summary>

public void CheckPirson()

{

for (int i = 0; i < CountIntervals; i++)

EquallyProbabilityDensity.Add(1.0 / CountIntervals);

//Рассчитанный хи квадрат

double hi2 = Formuls.GetHi2(ProbabilityDensity, CountNumbers, EquallyProbabilityDensity);

textBoxHi2Nabl.Text = hi2.ToString();

//число степеней свободы

int NumberOfDegreesOfFreedom = Formuls.GetNumberOfDegreesOfFreedom(CountIntervals, 0);

textBoxNumberOfDegreesOfFreedom.Text = NumberOfDegreesOfFreedom.ToString();

//Проверка гипотезы

if (hi2 < (double)numericUpDownHi2Tabl.Value)

{

labelCheckPirson.BackColor = Color.GreenYellow;

labelCheckPirson.Text = "Гипотеза принимается";

}

else

{

labelCheckPirson.BackColor = Color.Red;

labelCheckPirson.Text = "Гипотеза отвергается";

}

}

/// <summary>

/// Проверка по критерию Колмогорова

/// </summary>

public void CheckKolmogorov()

{

for (int i = 1; i <= CountNumbers; i++)

{

KolmogorovPlus.Add((double)i / CountNumbers - SequenceSort[i - 1]);

KolmogorovMinus.Add(SequenceSort[i - 1] - (double)(i) / CountNumbers);

}

double Kolmogorov = Math.Max(KolmogorovPlus.Max(), KolmogorovMinus.Max());

double alpha = (double)numericUpDownLvlZna4.Value;

double p = Math.Sqrt(Math.Pow(Math.Log(alpha, 2.73), 2) / (2 \* CountNumbers));

p = 0.964;

textBoxDnPlus.Text = KolmogorovPlus.Max().ToString();

textBoxDnMinus.Text = KolmogorovMinus.Max().ToString();

textBoxDn.Text = Kolmogorov.ToString();

textBox4dn.Text = p.ToString();

if (Kolmogorov <= p)

{

labelCheckKolmogorov.BackColor = Color.GreenYellow;

labelCheckKolmogorov.Text = "Гипотеза принимается";

}

else

{

labelCheckKolmogorov.BackColor = Color.Red;

labelCheckKolmogorov.Text = "Гипотеза отвергается";

}

}

public void TestLengthSeria()

{

int CountOne = 0;

for (int i = 0; i < CountNumbers; i++)

{

if (Sequence[i] < 0.25)

SequenceOneZero.Add(0);

else

{

SequenceOneZero.Add(1);

CountOne++;

}

}

int CountSeriaOne = 0;

for (int i = 1; i < CountNumbers; i++)

if (SequenceOneZero[i - 1] == 1 && SequenceOneZero[i] == 0)

CountSeriaOne++;

if (SequenceOneZero[CountNumbers - 1] == 1)

CountSeriaOne++;

double MxOne = Formuls.GetMxOne(Sequence, CountOne);

double DOne = Formuls.GetDOne(Sequence, CountOne);

double AverageLengthOne = Formuls.GetAverageLengthOne(CountOne, CountSeriaOne);

double alpha = 0.999;

double betta = 1 - alpha;

double quantile = 3.30;

double Zdown = MxOne - quantile \* Math.Sqrt(DOne / CountSeriaOne);

double Zup = MxOne + quantile \* Math.Sqrt(DOne / CountSeriaOne);

textBoxCountOne.Text = CountOne.ToString();

textBoxCountSeriaOne.Text = CountSeriaOne.ToString();

textBoxMxOne.Text = MxOne.ToString();

textBoxDOne.Text = DOne.ToString();

textBoxAverageLengthOne.Text = AverageLengthOne.ToString();

textBoxZdown.Text = Zdown.ToString();

textBoxZup.Text = Zup.ToString();

if (AverageLengthOne >= Zdown && AverageLengthOne <= Zup)

{

labelTestLengthSeria.BackColor = Color.GreenYellow;

labelTestLengthSeria.Text = "Гипотеза принимается";

}

else

{

labelTestLengthSeria.BackColor = Color.Red;

labelTestLengthSeria.Text = "Гипотеза отвергается";

}

}

/// <summary>

/// Хи квадрат

/// </summary>

public static double GetHi2(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;

}

/// <summary>

/// Число степеней свободы

/// </summary>

public static int GetNumberOfDegreesOfFreedom(int parK, int parM)

{

return parK - parM - 1;

}

/// <summary>

/// МатОжидание серии единиц

/// </summary>

public static double GetMxOne(List<double> parArr, int parCountOne)

{

double p = (double)parCountOne / parArr.Count();

return p / (1 - p);

}

/// <summary>

/// Дисперсия серии единиц

/// </summary>

public static double GetDOne(List<double> parArr, int parCountOne)

{

double p = (double)parCountOne / parArr.Count();

return p / Math.Pow(1 - p, 2);

}

/// <summary>

/// Оценка средней длины серии единиц

/// </summary>

public static double GetAverageLengthOne(int parCountOne, int parCountSeriaOne)

{

return (double)parCountOne / parCountSeriaOne;

}