МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Имитационное и статистическое моделирование

Отчёт по лабораторным работам

Подготовил

Студент 4 курса 2 группы

Минальд Андрей Анатольевич

Минск, 2018

Лабораторная работа №1

Условие

Используя метод Маклерена-Марсальи построить датчик БСВ (1 датчик должен быть мультипликативно конгруентный, второй – на выбор). Исследовать точность построенной БСВ.

1) Осуществить моделирование n = 1000 реализаций БСВ с помощью мультипликативного конгруэнтного метода (МКМ) с параметрами a0, β, M = 231.

2) Осуществить моделирование n = 1000 реализаций БСВ с помощью метода Макларена-Марсальи (один датчик должен быть мультипликативно конгруентный (п. 1), второй – на выбор). K – объем вспомогательной таблицы.

3) Проверить точность моделирования обоих датчиков (п. 1 и п. 2) с помощью критерия согласия Колмогорова и χ2-критерия Пирсона с уровнем значимости ε = 0.05.

a0 = β = 79 507, K = 6

Теория

Мультипликативный конгруэнтный метод (метод вычетов).

Псевдослучайная последовательность реализаций  БСВ  определяется по рекуррентным формулам: где  - параметры датчика (натуральные числа):  - множитель  модуль,  - стартовое значение (нечётное число).

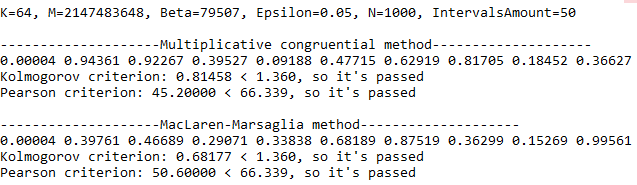
Операция  означает вычет числа  по модулю : где  - целая часть числа 

Метод Маклорена-Марсальи основан на комбинировании двух простейших программных датчиков БСВ. Пусть – псевдослучайные последовательности, порождаемые независимо работающими датчиками;  результирующая псевдослучайная последовательность реализаций БСВ;  вспомогательная таблица  чисел.

Процесс вычисления  включает следующие этапы:

* первоначальное заполнение таблицы 
* случайный выбор из таблицы: 
* обновление табличных значений: 

Результаты



Лабораторная работа №2

Условие

Смоделировать дискретную случайную величину (задания на стр. 18-22). Исследовать точность моделирования.

1) Осуществить моделирование n = 1000 реализаций СВ из заданных дискретных распределений.

2) Вывести на экран несмещенные оценки математического ожидания и дисперсии, сравнить их с истинными значениями.

3) Для каждой из случайных величин построить свой χ2-критерием Пирсона с уровнем значимость ε=0.05. Проверить, что вероятность ошибки I рода стремится к 0.05.

4) Осуществить проверку каждой из сгенерированных выборок каждым из построенных критериев

Теория

Дискретной случайной величиной (ДСВ) называется случайная величина , имеющая дискретное распределение вероятностей, определяемое дискретным множеством значений  и заданными вероятностями значений

 (1)

Основными функциональными и числовыми характеристиками ДСВ являются 

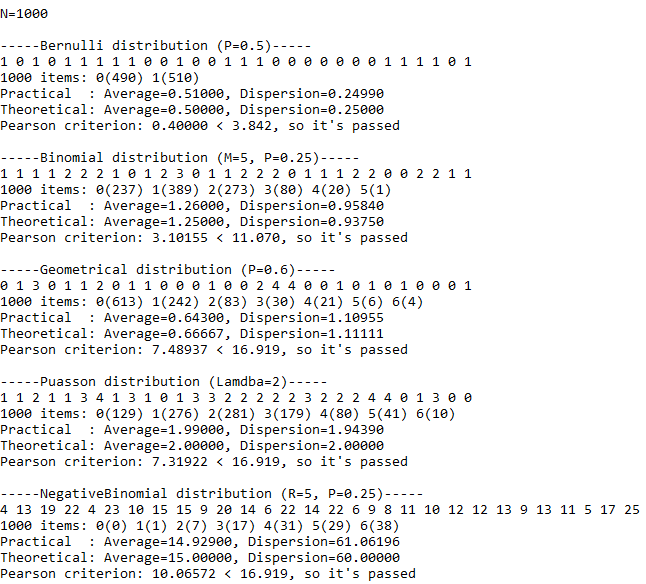
* функция распределения: где 
* функции вероятности:, где – символ Кронекера;
* математическое ожидание:

Алгоритм моделирования ДСВ , заданной распределением (1), состоит из вычисления вспомогательного вектора  и двух шагов, повторяющихся при каждом обращении к алгоритму:

1. Моделирование с помощью датчика БСВ реализации *a*.
2. Принятие решения о том, что реализацией является *x*, определяемое по правилу:

Результаты



Лабораторная работа №3

Условие

Смоделировать непрерывную случайную величину (задания на стр. 25-47). Исследовать точность моделирования.

1) Осуществить моделирование n = 1000 реализаций СВ из нормального закона распределения N(m, s2) с заданными параметрами. Вычислить несмещенные оценки математического ожидания и дисперсии, сравнить их с истинными.

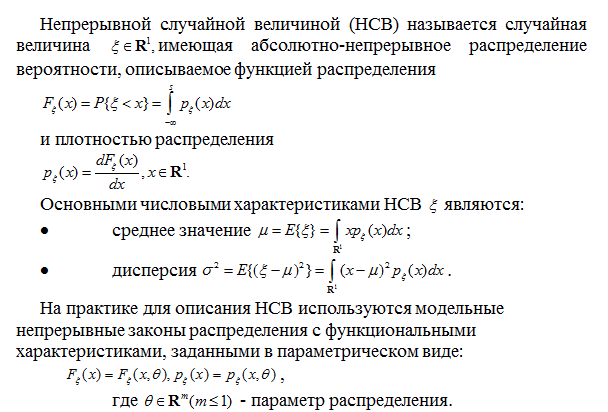
2) Смоделировать n = 1000 СВ из заданных абсолютно непрерывных распределений. Вычислить несмещенные оценки математического ожидания и дисперсии, сравнить их с истинными значениями (если это возможно).

3) Для каждой из случайных величин построить свой критерий Колмогорова с уровнем значимость ε=0.05. Проверить, что вероятность ошибки I рода стремится к 0.05.

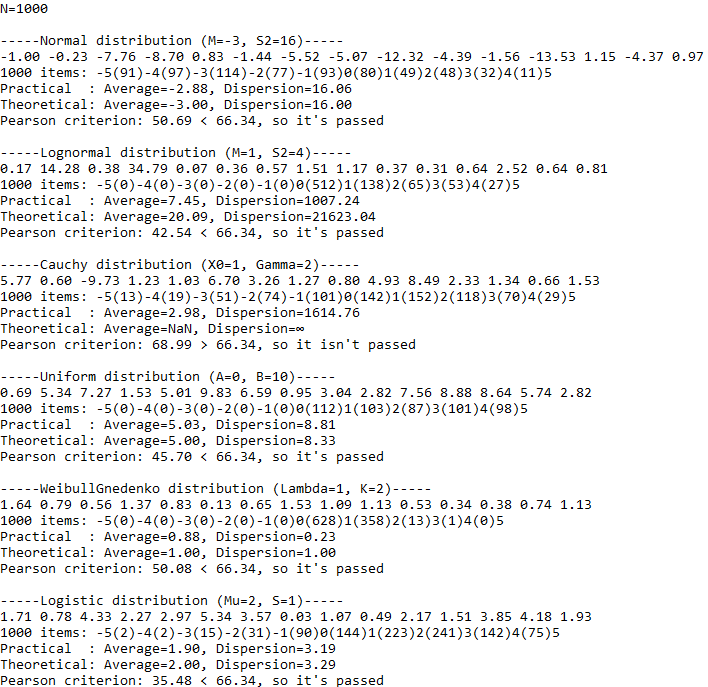
4) Для каждой из случайных величин построить свой χ2-критерий Пирсона с уровнем значимость ε=0.05. Проверить, что вероятность ошибки I рода стремится к 0.05.

5) Осуществить проверку каждой из сгенерированных выборок каждым из построенных критериев.

Теория



Результаты



Лабораторная работа №4

Условие

Вычислить значение интеграла, используя метод Монте-Карло. Оценить точность.

1. По методу Монте-Карло вычислить приближенное значения интегралов.

2. Сравнить полученное значение либо с точным значением (если его получится вычислить), либо с приближенным, полученным в каком-либо математическом пакете (например, в mathematica). Для этого построить график зависимости точности вычисленного методом Монте-Карло интеграла от числа итераций n.



Теория

Рассмотрим задачу приближенного вычисления интеграла  где - подмножество из  При n=1 имеем определенный интеграл вида 

Заметим, что схема вычислений как многомерных, так и одномерных интегралов , абсолютно аналогична.

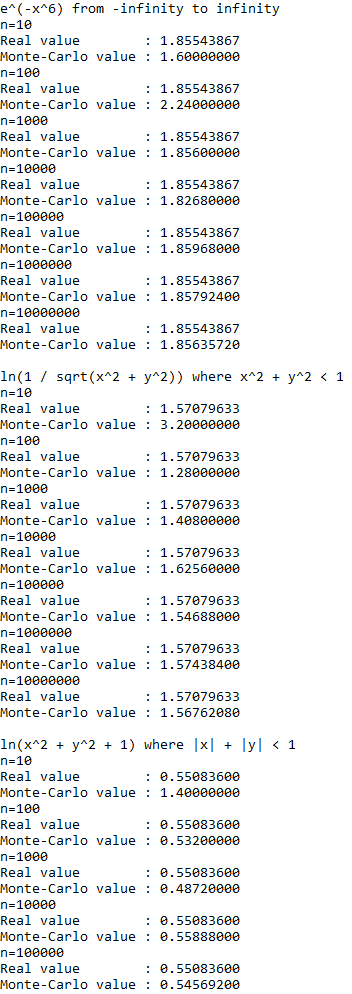
Пусть η – произвольная случайная величина с плотностью распределения вероятностей  Предполагается только, что существуют моменты случайных величин, встречающиеся ниже. Рассмотрим случайную величину, являющуюся функциональным преобразованием случайной величины η.



Можно показать, что  Поэтому в качестве приближенного значения интеграла можно использовать статистическую оценку , построенную в выборке из n независимых случайных величин 



Результаты



Лабораторная работа №5

Условие

Решить систему линейных уравнений, используя метод Монте-Карло.

1. Решить систему линейных алгебраических уравнений методом Монте-Карло.

2. Сравнить с решением данного уравнения, полученным в произвольном математическом пакете.

3. Построить график зависимости точности решения от длины цепи Маркова и числа смоделированных цепей Маркова.

Теория

Пусть система алгебраических уравнений задана в виде  (1), где – вектор-столбец неизвестных,  – вектор правых частей и ,  – матрица системы.

Предположим, что наибольшее по модулю характеристическое число матрицы *А* меньше единицы, так что сходиться метод последовательных приближений

,  (2)

Достаточным условием для того, чтобы все характеристические числа матрицы *А* лежали внутри единичного круга на комплексной плоскости, то есть, чтобы , , может служить неравенство  или неравенство .

Рассмотрим задачу о вычислении скалярного произведения , где *h* – заданный вектор.

Мы будем связывать с системой (1) и вектором *h* некоторую фиксированную цепь Маркова из множества цепей, определяемых парой :

, (3)

, (4)

для которых выполнены условия:

, если ,

, если ;  (5)

Положим

  (6)

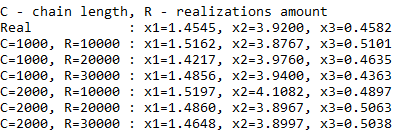
Зададимся некоторым целым *N* , и будем рассматривать траектории цепи Маркова длины *N >*0. Движущейся частице приписывается "вес" , который изменяется при движении ее по траектории  следующим образом.

, . (7)

Введем случайную величину , определенную на траекториях Марковской цепи длины *N* , (8)

Теперь можем построить СВ  по формуле: , где  - номер реализации цепи Маркова. Приближенное значение для  имеет вид  в зависимости от вектора h.

Результаты

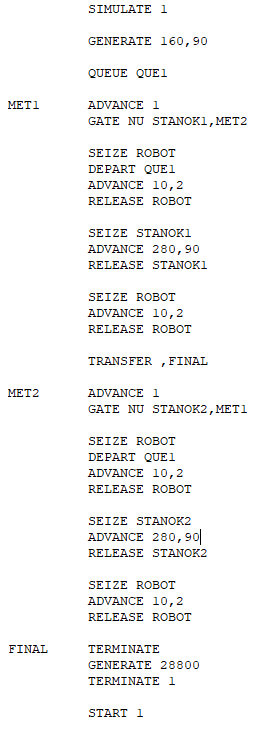


Лабораторная работа №6

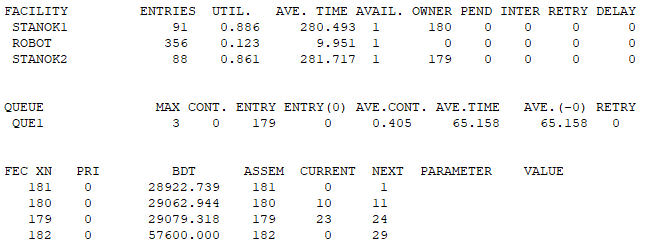
Условие

ЗАДАНИЕ 12. Моделирование процесса функционирования участка механообработки ГАП Участок механообработки включает входной конвейер, транспортный робот, два однотипных независимо работающих станка с ЧПУ и выходной конвейер. Время поступления деталей на обработку и время обработки детали на станке распределены равномерно на интервалах [a ± δ] сек., [b ± δ] сек. соответственно. Робот переносит деталь с входного конвейера на свободный станок и затем со станка на выходной конвейер, время каждого переноса распределено равномерно на интервале [c ± ν] сек. Разработать GPSSV - модель для анализа процесса функционирования участка в течение k смен (продолжительность одной смены 8 часов ). Первоначальный перечень экспериментов: k = 1, = 160, δ = 90, b = 280, ε = 30, c = 10, ν = 2.

Код программы



Результаты

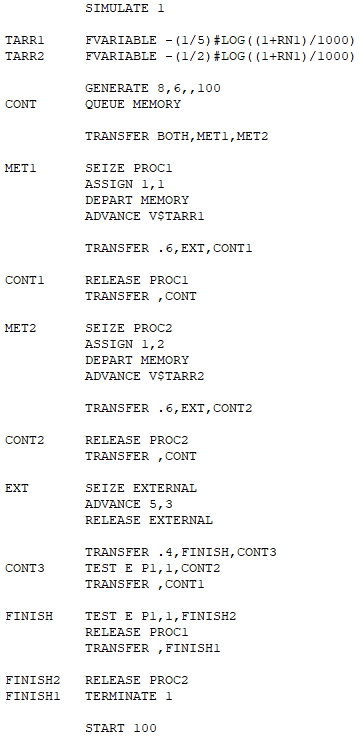


Лабораторная работа №7

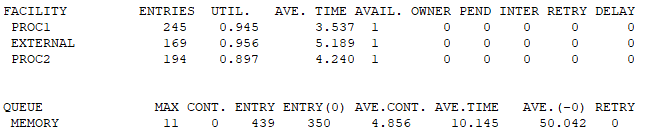
Условие

ЗАДАНИЕ 46. Моделирование процесса функционирования двухпроцессорной ЭВМ Двухпроцессорная ЭВМ имеет общую память, разделенную на n, блоков. Каждой задаче при ее решении отводится один блок. Длины временных промежутков между поступлениями задач случайны, независимы и одинаково распределены по равномерному закону [τ-, τ+]. Время обработки порции информации в процессоре Пj случайно и подчинено экспоненциальному закону Е(µj), j=1,2. Между обработкой порций с вероятностью p возможно обращение к внешней памяти со случайным временем обращения, подчиненным равномерному закону R[t-, t+]. C вероятностью q = 1 - p задача оказывается «решенной» и покидает ЭВМ. Разработать GPSSV - модель для анализа, функционирования ЭВМ при решении N задач. Первоначальный перечень экспериментов: τ- = 2, τ+ = 14, µ-= 0.2, µ+ = 0.5, p = 0.6, t- = 2, t+ = 8, N = 100.

Код программы на GPSS



Результаты программы на GPSS



Код программы на C#

public class Processor : IDevice

{

private BernulliRandom \_rand;

public QueueEx ProcQueue { get; protected set; }

public MicroProcessor Proc1 { get; protected set; }

public MicroProcessor Proc2 { get; protected set; }

public Processor(double procTime1, double procTime2)

{

ProcQueue = new QueueEx();

Proc1 = new MicroProcessor(procTime1);

Proc2 = new MicroProcessor(procTime2);

\_rand = new BernulliRandom(0.6);

}

public void EnQueueTransact(Transact transact)

{

ProcQueue.EnQueueTransact(transact);

}

private void DeQueueTransact(Transact transact)

{

ProcQueue.DeQueueTransact(transact);

}

public bool ProcessTransact(Transact transact)

{

if (Proc1.Time <= transact.Time)

{

DeQueueTransact(transact);

Proc1.ProcessTransact(transact);

PostProcess(transact);

return true;

}

if (Proc2.Time <= transact.Time)

{

DeQueueTransact(transact);

Proc2.ProcessTransact(transact);

PostProcess(transact);

return true;

}

transact.Time = (Proc1.Time < Proc2.Time) ? Proc1.Time : Proc2.Time;

return false;

}

private void PostProcess(Transact transact)

{

var r = \_rand.Next();

if (r == 0)

{

transact.Status = TransactStatus.ReProc;

transact.Device = CompSystem.Memory;

}

else

{

transact.Status = TransactStatus.Finished;

transact.Device = CompSystem.Memory;

}

}

}

public class TransactGenerator

{

private readonly UniformRandom \_rand;

public double Time { get; set; }

public double LowerBound

{

get { return \_rand.LowerBound; }

set { \_rand.LowerBound = value; }

}

public double UpperBound

{

get { return \_rand.UpperBound; }

set { \_rand.UpperBound = value; }

}

public TransactGenerator(double lowerBound, double upperBound)

{

Time = 0;

\_rand = new UniformRandom(lowerBound,upperBound);

}

public Transact Generate()

{

Time += \_rand.Next();

return new Transact { Time = Time, Status = TransactStatus.Prepared};

}

public List<Transact> Generate(int count)

{

var transacts = new List<Transact>(count);

for(var i = 0; i < count;i++)

{

transacts.Add(Generate());

}

return transacts;

}

}

Результаты программы на C#

