

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ
ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

Журнал практики

Студента Махмудова Орхана Сакитовича

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра №805 «Математическая кибернетика»

Учебная группа 80-405Б-18

Направление подготовки (специальность) **01.03.04**

(шифр)

Прикладная математика

(название направления, специальности)



Вид практики **преддипломная**

(учебной, производственной, преддипломной или другой вид практики)

Руководитель практики от МАИ

Шестакова Ольга Владимировна

(фамилия, имя, отчество) (подпись)

 /  / “ 10 ” мая 2022 г.
(подпись студента) (дата)

1. Место и сроки проведения практики

Сроки проведения практики: -дата начала практики **9.02.22**

-дата окончания практики **10.05.22**

Наименование предприятия **МАИ**

Название структурного подразделения (отдел, лаборатория)

каф. 805

2. Инструктаж по технике безопасности

_____ / _____ / “9”.02.2022 г.
(подпись проводившего) (дата проведения)

3. Индивидуальное задание студенту

Разработать программное обеспечение для моделирования пуассоновских потоков различными методами. Провести сравнительный анализ метода моделирования, основанного на свойстве ординарности пуассоновского потока, и методов максимального сечения.

4. План выполнения индивидуального задания

Анализ постановки задачи. Изучение методов моделирования пуассоновских потоков

Разработка алгоритма решения задачи

Разработка программного обеспечения для моделирования пуассоновских потоков методом, основанном на свойстве ординарности

Разработка программного обеспечения для моделирования пуассоновских потоков методом максимального сечения и его модифицированным вариантом. Применение к задаче оценивания состояний динамических систем

Сравнительный анализ методов моделирования пуассоновских потоков

Руководитель практики от МАИ: Шестакова Ольга Владимировна _____/_____/

Руководитель от предприятия: Рыбаков Константин Александрович



/ Рыбаков К.А./



/ Махмудов О.С. / “9”02. 2022г.
(подпись студента) (дата)

5.Отзыв руководителя практики от предприятия

Работа выполнена в полном объеме и заслуживает оценки «Отлично».

Руководитель от предприятия:

Рыбаков Константин Александрович



«10»мая2022г.

М.П. (печать)

6. Отчет студента о практике

В данной работе предложено программное обеспечение для моделирования пуассоновских потоков разными методами. Предложена новая модификация моделирования метода моделирования, приведено сравнение модификации с другими методами и выявлена эффективность новой модификации.

Постановка задачи

Потоком событий будем называть однотипные события, происходящие в случайные моменты времени $\tau_k \in [t_0, +\infty)$, $k = 1, 2, \dots$. Обозначим через $P(t)$ число произошедших событий к определённому моменту $t > t_0$ – момент времени с которого ведётся наблюдение, т.е. $P(t)$ – считающий случайный процесс, $P(t_0) = 0$.

Предположим, что:

$$\mathbb{P}\{P(t + \Delta t) - P(t) = 1\} = \mu(t)\Delta t + o(\Delta t),$$

$$\mathbb{P}\{P(t + \Delta t) - P(t) > 1\} = o(\Delta t)$$

Вероятность наступления одного события в промежутке времени $\Delta t \rightarrow 0$ определяется интенсивностью потока $\mu(t)$, а вероятность наступления более одного события на этом же промежутке времени бесконечно мала. Интенсивность $\mu(t)$ определяет среднюю скорость появления событий, или среднее число событий, происходящих в единицу времени.

Обозначим через N_k – число событий потока на подмножестве $Y_k \subset [t_0, +\infty)$. Если эти подмножества не пересекаются, то случайные величины N_k независимы. Эти свойства потока событий называются ординарностью и отсутствием последействия. Поток событий, обладающий этими свойствами, называется пуассоновским потоком событий, а считающий процесс $P(t)$ – пуассоновским процессом.

Требуется смоделировать пуассоновский поток событий с помощью разных методов и провести их сравнительный анализ. Интенсивность потока и временные

промежутки задаются пользователем.

Общий подход к методам моделирования

Подход основан на непосредственном моделировании моментов времени τ_k , $k = 1, 2 \dots$. Если поток событий является однородным $\mu(t) = \mu = \text{const}$, то промежуток времени $\Delta\tau_k$ между двумя последовательными событиями с номерами $k-1$ и k имеет показательное распределение с параметром μ

$$\tau_k = \tau_{k-1} + \Delta\tau_k, \quad \tau_0 = t_0,$$

где $\Delta\tau_k$ – реализация случайной величины, которая моделируется следующим образом:

$$\xi = \frac{\ln \alpha}{\mu},$$

где α – случайная величина, имеющая равномерное распределение на интервале $(0,1)$.

Так как для неоднородного потока событий случайная величина $\Delta\tau_k$ характеризуется сложной функцией распределения, чтобы точно смоделировать неоднородный поток применяют следующие методы.

Метод максимального сечения

Согласно методу максимального сечения промежуток времени $\Delta\tau_k$ определяется равенством:

$$\Delta\tau_k = \theta_N, \quad N = \min \left\{ l: \alpha_l \leq \frac{\mu(\tau_{k-1} + \theta_l)}{\mu^*} \right\}, \quad \theta_l = \sum_{i=1}^l \xi^i, (*)$$

где $\xi^1, \xi^2, \dots, \xi^i$ – последовательность независимых одинаково распределенных случайных величин, имеющих показательное распределение с параметром μ^* , ограничивающим сверху значения интенсивности $\mu(t): \mu(t) \leq \mu^* =$

$const, a \alpha_1, \alpha_2, \dots, \alpha_i$ — последовательность независимых случайных величин, имеющих равномерное распределение на интервале $(0,1)$.

Первая модификация метода максимального сечения

Существует более экономичная модификация метода максимального сечения, которая предполагает вычисление числа N в (*) по формуле:

$$N = \min \left\{ l: 1 - \alpha > \prod_{i=1}^l \left(1 - \frac{\mu(\tau_{k-1} + \theta_i)}{\mu^*} \right) \right\}$$

где α — случайная величина, имеющая равномерное распределение на интервале $(0,1)$. Использование данной модификации сокращает количество обращений к генератору псевдослучайных чисел, что в свою очередь уменьшает время моделирования.

Вторая модификация метода максимального сечения

Данная модификация основана на том, что мы обращаемся к генератору псевдослучайных чисел единожды, а далее обновляем случайные величины по следующей формуле:

$$\beta^{(1)} = \alpha, \beta^{(i+1)} = \begin{cases} \frac{\beta^{(i)}}{p^{(i)}}, & \text{если } \beta^{(i)} \in (0, p^{(i)}), \\ \frac{\beta^{(i)} - p^{(i)}}{1 - p^{(i)}}, & \text{если } \beta^{(i)} \in (p^{(i)}, 1), \end{cases}$$

где α — случайная величина, имеющая равномерное распределение на интервале $(0,1)$, а $p^{(i)} = \frac{\mu(\tau_i)}{\mu^*}$. Согласно формуле изложенной выше, случайные величины $\beta^{(1)}, \beta^{(2)}, \dots, \beta^{(i)}$ — также имеют равномерное распределение на интервале $(0,1)$. Далее вместо величин α_i мы по формуле метода максимального сечения используем величины $\beta^{(i)}$.

Линейный конгруэнтный генератор

Суть генератора заключается в вычислении последовательности случайных чисел X_i по следующему правилу:

$$X_{i+1} = (aX_i + c) \bmod m$$

где m – модуль (натуральное число, относительно которого вычисляется остаток от деления, $m \geq 2$), a – множитель ($0 \leq a \leq m$), c – приращение ($0 \leq c \leq m$), X_0 – начальное значение ($0 \leq X_0 \leq m$).

Метод генерации псевдослучайных чисел при $c = 0$ называют мультипликативным конгруэнтным методом, при $c \neq 0$ – смешанным конгруэнтным методом. В данной работе использованы следующие параметры: $c = 0$, $a = 2^{40}$, $m = 5^{17}$.

Числа подобраны таким образом, чтобы добиться максимальной периодичности и качества псевдослучайной последовательности. Период генератора равен 2^{38} .

RND128

RND128 работает на основе линейного конгруэнтного генератора. Однако в отличие от ЛКГ, RND128 оперирует числами, разрядность которых выше разрядности процессора, поэтому большие числа разбиваются на «подчисла» и хранятся в массиве.

RND128 использует следующие параметры: $c = 0$, $a = 2^{128}$, $m = 5^{100109}$. Период генератора равен 2^{126} .

Вихрь Мерсенна

Алгоритм генератора Вихря Мерсенна основан на следующей рекуррентной формуле:

$$x_{k+n} = x_{k+m} \oplus (x_k^u | x_{k+1}^l) A, \quad k = 0, 1, \dots,$$

где n – целое число, которое обозначает порядок рекуррентности,

m – целое число ($1 \leq m < n$),

A – матрица размера $w \times w$, с элементами из $F_2 = \{0, 1\}$,

$|$ – побитовое или (OR),

\oplus – сложение по модулю два (XOR),

Последовательность $x_{k+n}, x_{k+n-1}, \dots, x_k^u$ образует $(n \times w - r)$ – массив. Так как r битов отбрасываются, массив называется неполным массивом. Параметры n и r выбраны так, чтобы характеристический многочлен был примитивным, а $nw - r$ равно числу Мерсенна 19937. Период генератора равен $2^{19937} - 1$.

Сравнительный анализ методов моделирования

Программное обеспечение разработано на языке C++ с использованием компилятора GNUG++ и оптимизации компилятора O3.

В программе заданы следующие параметры:

- Интенсивность потока $\mu(t) = \mu^* e^{-2t}$, где $\mu^* = 10$ – верхняя оценка интенсивности.
- Начальное и конечное время моделирования процесса $t_0 = 0$ и $t_k = 2$
- Количество реализаций пуассоновских потоков $N = 100000000$
- Далее приведены результаты исследования в таблицах

Линейный когрозентный генератор (ЛКГ)

	Время работы программы (с)	Мат. ожидание и дисперсия	Выборочное среднее	Выборочная дисперсия
Метод максимального сечения	47.9251	4.90842	4.90846	4.90899
Первая модификация метода	51.2957	4.90842	4.90867	4.90842
Вторая модификация метода	54.4367	4.90842	4.90822	4.90813

RND128

	Время работы программы (с)	Мат. ожидание и дисперсия	Выборочное среднее	Выборочная дисперсия
Метод максимального сечения	74.3296	4.90842	4.90814	4.90825
Первая модификация метода	69.8353	4.90842	4.90876	4.90897
Вторая модификация метода	69.1230	4.90842	4.90889	4.90900

Вихрь Мерсенна

	Время работы программы (с)	Мат. ожидание и дисперсия	Выборочное среднее	Выборочная дисперсия
Метод максимального сечения	91.7894	4.90842	4.90878	4.90914
Первая модификация метода	78.8711	4.90842	4.90832	4.90960
Вторая модификация метода	77.0645	4.90842	4.90793	4.90668

Используя ЛКГ в качестве генератора псевдослучайных чисел (ГПСЧ) достигается минимальное время моделирования потоков. Однако модификации не дают выигрыша во времени по сравнению с обычным методом. Это связано с тем, что ЛКГ использует всего 3 математические операции, которые при оптимизации компилятора ОЗ работают быстрее других операций, использующихся в модификациях. ЛКГ обладает меньшим качеством и периодичностью последовательности, в отличие от RND128 и Вихря Мерсенна, поэтому стоит обратить внимание на результаты других ГСПЧ.

Используя RND128 или Вихрь Мерсенна, достигаются ожидаемые результаты. Модификации метода максимального сечения выигрывают во времени моделирования. Вторая модификация показывает наилучшее время по сравнению с другими методами.

Листинг программы

```

#include <iostream>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include <random>

using namespace std;

// double random_alfa()
// {
//     const unsigned long long int __RAND_Multiplier = 762939453125; // 5^17
//     const unsigned long long int __RAND_Divider = 1099511627776; // 2^40
//     static unsigned long long int __RAND_Seed = 1;

//     __RAND_Seed = (__RAND_Seed * __RAND_Multiplier) % __RAND_Divider;
//     return static_cast<double>(__RAND_Seed) / __RAND_Divider;
// }

// double random_alfa()
// {
//     static long long int u[5] = { 1, 0, 0, 0, 0 };
//     const long long int m[5] = { 14919573, 10735332, 23806020, 21375263, 16382667 };
//     const double x[5] = {
//         pow(2.0, -128),
//         pow(2.0, -102),
//         pow(2.0, -76),
//         pow(2.0, -50),
//         pow(2.0, -24) };
//     longlong int c0, c1, c2, c3, c4, n;

//     c0 = m[0] * u[0];
//     c1 = m[0] * u[1] + m[1] * u[0];
//     c2 = m[0] * u[2] + m[1] * u[1] + m[2] * u[0];
//     c3 = m[0] * u[3] + m[1] * u[2] + m[2] * u[1] + m[3] * u[0];
//     c4 = m[0] * u[4] + m[1] * u[3] + m[2] * u[2] + m[3] * u[1] + m[4] * u[0];

//     u[0] = c0 - ((c0 >> 26) << 26);
//     n = c1 + (c0 >> 26);
//     u[1] = n - ((n >> 26) << 26);
//     n = c2 + (n >> 26);
//     u[2] = n - ((n >> 26) << 26);
//     n = c3 + (n >> 26);
//     u[3] = n - ((n >> 26) << 26);
//     n = c4 + (n >> 26);
//     u[4] = n - ((n >> 24) << 24);

//     return u[0] * x[0] + u[1] * x[1] + u[2] * x[2] + u[3] * x[3] + u[4] * x[4];

```

```

// }

double random_alfa()
{
    static mt19937_64 gen(1);
    uniform_real_distribution<>dist(0., 1.);
    return dist(gen);
}

double beta_simulation(double beta, double p)
{
    if (beta > p)
        return beta = (beta - p) / (1 - p);
    else
        return beta = beta / p;
}

double simulation_exponential_distribution(double
mu_above) //моделированиеслучайныхвеличинспоказательнымраспределением
{
    return -log(random_alfa()) / mu_above;
}

double simulation_mu(double t, double mu_above) //функция mu(t) (интенсивностьпотока)
{
    return mu_above * exp(-2 * t);
}

double mathematical_expectation(double t, double mu_above) //точноематематическоеожидание
{
    return (mu_above / 2) * (1 - exp(-2 * t));
}

double maximum_cross_section_method(double t0, double tk, double mu_above) //методмаксимальногосечения
{
    double teta = t0 + simulation_exponential_distribution(mu_above);
    while ((random_alfa() > (simulation_mu(teta, mu_above) / mu_above)) && (teta < tk))
    {
        teta += simulation_exponential_distribution(mu_above);
    }
    if (teta <= tk)
        return teta;
    else
        return 0;
}

double modified_maximum_cross_section_method(double t0, double tk, double
mu_above) //модификацияметодамаксимальногосечения
{

```

```

double teta = t0 + simulation_exponential_distribution(mu_above);
double composition = 1 - simulation_mu(teta, mu_above) / mu_above;
double alfa = random_alfa();
while ((1 - alfa <= composition) && (teta < tk))
{
    teta += simulation_exponential_distribution(mu_above);
    composition *= (1 - simulation_mu(teta, mu_above) / mu_above);
}
if (teta <= tk)
    return teta;
else
    return 0;
}

double maximum_cross_section_method_with_beta(double t0, double tk, double mu_above) //метод максимального сечения (с
beta)
{
    // static double eps = pow(2, -20);
    // static double l = 0.0;
    double teta = t0 + simulation_exponential_distribution(mu_above);
    static double beta = random_alfa();
    double p = simulation_mu(teta, mu_above) / mu_above;
    // if (beta < p)
    //     l *= p;
    // else
    //     l *= (1 - p);
    // if (l >= eps)
    //     beta = random_alfa();
    while ((beta > p) && (teta < tk))
    {
        beta = beta_simulation(beta, p);
        teta += simulation_exponential_distribution(mu_above);
        p = simulation_mu(teta, mu_above) / mu_above;
        // l *= (1 - p);
        // if (l >= eps)
        //     beta = random_alfa();
    }
    beta = beta_simulation(beta, p);
    if (teta <= tk)
        return teta;
    else
        return 0;
}

int main()
{
    double t0 = 0,           //начальное время
    tk = 2,                 //конечное время
    mu_above = 10;          //верхняя оценка mu

```

```

long long int N = 100000000;    //число реализаций

long long int events = 0, average_sample = 0;
double sample_variance = 0;
double teta;
double math_expectation = mathematical_expectation(tk, mu_above);
long long int work_time = clock();
for (int i = 0; i < N; i++)
{
    events = 0;
    teta = maximum_cross_section_method(t0, tk, mu_above);
    while (teta > 0)
    {
        events += 1;
        teta = maximum_cross_section_method(teta, tk, mu_above);
    }
    average_sample += events;
    sample_variance += pow((events - math_expectation), 2);
}
work_time = clock() - work_time;
cout<<endl;
cout<< "Maximum cross section method\n";
cout<< "Simulation time: " << (double)work_time / 1000000 <<endl;
cout<< "Exact value of mathematical expectation: " <<math_expectation<<endl;
cout<< "Average sample: " << (double)average_sample / N <<endl;
cout<< "Sample variance: " <<sample_variance / N <<endl;
cout<< "Max estimation of the flow intensity: " <<mu_above<<endl;
cout<< "Number of iterations: " << N <<endl<<endl;

events = 0;
average_sample = 0;
sample_variance = 0;
work_time = clock();
for (int i = 0; i < N; i++)
{
    events = 0;
    teta = modified_maximum_cross_section_method(t0, tk, mu_above);
    while (teta > 0)
    {
        events += 1;
        teta = modified_maximum_cross_section_method(teta, tk, mu_above);
    }
    average_sample += events;
    sample_variance += pow((events - math_expectation), 2);
}
work_time = clock() - work_time;
cout<< "Modified maximum cross section method\n";
cout<< "Simulation time: " << (double)work_time / 1000000 <<endl;
cout<< "Exact value of mathematical expectation: " <<math_expectation<<endl;

```



```

cout<< "Average sample: " << (double)average_sample / N <<endl;
cout<< "Sample variance: " <<sample_variance / N <<endl;
cout<< "Max estimation of the flow intensity: " <<mu_above<<endl;
cout<< "Number of iterations: " << N <<endl<<endl;

events = 0;
average_sample = 0;
sample_variance = 0;
work_time = clock();
for (int i = 0; i< N; i++)
{
    events = 0;
    teta = maximum_cross_section_method_with_beta(t0, tk, mu_above);
    while (teta> 0)
    {
        events += 1;
        teta = maximum_cross_section_method_with_beta(teta, tk, mu_above);
    }
    average_sample += events;
    sample_variance += pow((events - math_expectation), 2);
}
work_time = clock() - work_time;
cout<< "Maximum cross section method (with beta)\n";
cout<< "Simulation time: " << (double)work_time / 1000000 <<endl;
cout<< "Exact value of mathematical expectation: " <<math_expectation<<endl;
cout<< "Average sample: " << (double)average_sample / N <<endl;
cout<< "Sample variance: " <<sample_variance / N <<endl;
cout<< "Max estimation of the flow intensity: " <<mu_above<<endl;
cout<< "Number of iterations: " << N <<endl<<endl;

return 0;
}

```

Заключение

Результатом данной работы является программное обеспечение для моделирования пуассоновского потока событий разными методами. Представлен сравнительный анализ методов моделирования. Приведена новая модификация метода максимального сечения и показана эффективность использования данного метода на примерах разных ГПСЧ.