УДК 004.032.24/94 EDN: INOXYB

# ОЦЕНИВАНИЕ ДОВЕРИТЕЛЬНЫХ ИНТЕРВАЛОВ ПРИ ИМИТАЦИОННОМ МОДЕЛИРОВАНИИ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ И БИБЛИОТЕКИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ Си++

**А.Ю. Быков** abykov@bmstu.ru
**А.Р. Поремский** poremskiyar@student.bmstu.ru

**МГТУ им. Н.Э. Баумана, Москва, Российская Федерация**

**Аннотация**

Рассмотрен подход к получению оценок в виде доверительных интервалов для искомых параметров системы массового обслуживания на имитационных моделях, требующих, как правило, значительных вычислительных ресурсов. Предложено использовать технологию параллельных вычислений для снижения времени имитационных экспериментов. Получение доверительных интервалов основано на многократных экспериментах с одной и той же моделью, которые выполняются независимо друг от друга, т. е. параллельно. Результаты экспериментов с одной и той же моделью при использовании генераторов случайных чисел различаются, если генераторы псевдослучайных чисел, основанные на линейном конгруэнтном генераторе, в каждом эксперименте инициализируются разными начальными значениями. Имитационное моделирование выполнено с помощью специальной библиотеки функций языка Си++, библиотека модернизирована для работы в многопоточном режиме — все функции и глобальные переменные включены в класс, каждый эксперимент с моделью проводили, используя свой объект класса. Эксперименты равномерно распределяются между потоками внутри многопоточной программы. Приведены пример моделирования типовой системы вида M/M/1, результаты экспериментов, зависимость времени моделирования от числа потоков и выводы

**Ключевые слова**

*Многопоточная программа, генератор псевдослучайных чисел, линейный конгруэнтный генератор, доверительная вероятность, классы, объект, Си++*

Поступила 10.03.2025
Принята 04.09.2026
© Автор(ы), 2026

**Введение.** Недостатком имитационного моделирования (ИМ) является необходимость наличия больших вычислительных ресурсов, из-за этого время выполнения имитационных экспериментов на ЭВМ может быть достаточно велико. Частично эту проблему можно решить, если использовать технологию параллельных вычислений на многопроцессорных (многоядерных) ЭВМ, в одном процессе запускаются несколько потоков, которые могут выполняться одновременно.

Для ИМ систем массового обслуживания (СМО) используется дискретно-событийный подход [1, 2], который реализован, например, в языке GPSS [2, 3] и в библиотеке функций языка Си++, разработанной для ИМ [4, 5]. Особенность подобных моделей заключается в переходе системы из одного состояния в другое в моменты наступления событий. Поскольку последовательность событий является важным фактором, то реализовать параллельный алгоритм для подобной модели достаточно проблематично, хотя попытки делаются, но с ограничениями. Распараллеливание можно проводить, когда требуется многократное выполнение одной и той же модели, например, при проведении интервальной оценки искомых параметров для получения доверительных интервалов при заданной доверительной вероятности.

Приведем краткий обзор некоторых работ из области применения параллельных вычислений при ИМ.

Проблема параллельного моделирования агентных систем рассмотрена в [6], предложен алгоритм синхронизации, специально разработанный для моделей с частым взаимодействием агентов, где каждый агент за раунд меняет состояние не более одного раза.

Проблема моделирования редких событий в биологических системах изучена в [7], где параллельно проверено множество возможных путей, ведущих к интересующему редкому событию.

В рамках дискретно-событийного подхода к ИМ СМО в [8] предложено вместо длительного моделирования одной реализации случайного процесса, протекающего в СМО, использовать моделирование множества коротких реализаций этого случайного процесса, т. е. можно применить распараллеливание.

Имитационное моделирование случайных потоков поступления грузов на склад и оттока (выдачи) грузов со склада рассмотрено в [9]. Исходные данные (время поступления и оттока) сформированы в разных потоках изначально для всего интервала времени моделирования в виде матриц, за счет этого выполнено распараллеливание. При таком подходе необходим

большой объем памяти. В связи с этим его применяют для ограниченных по длительности интервалов времени моделирования.

Проблемы параллельного моделирования рассмотрены в [10, 11]. Приведено предположение, что сложная система может быть разбита на подсистемы, которые можно имитировать параллельно, рассмотрены проблемы изменения модельного времени, выделены локальное время внутри задач (потоков выполнения) и глобальное время, предложено использовать механизмы синхронизации, рассмотрены проблемы балансировки нагрузки.

Оценка производительности API (Application Programming Interface) OpenMP и OpenACC на примере моделирования сушки зерна в пористой среде проведена в [12], где имитационная модель априорно разделена на подмодели, которые могут выполняться параллельно.

Подходы к параллельному моделированию процессов, происходящих в распределенных энергосетях, с целью получить выигрыш по времени, приведены в [13, 14]. Поскольку сети разделены на относительно независимые подсистемы, моделирование можно проводить параллельно. Некоторые аспекты реализации протоколов передачи данных в распределенных сетях в целях минимизации энергопотребления рассмотрены в [15].

В результате приведенного обзора литературы определено, что проблема распараллеливания задач для уменьшения общего времени проведения экспериментов при ИМ окончательно не решена. Выполнены отдельные попытки ее решить для систем с определенной структурой или с учетом некоторых ограничений и (или) упрощений.

*Цель настоящей работы* — предложить способ получения интервальных оценок (доверительного интервала при заданной доверительной вероятности) искомых параметров при ИМ СМО.

Получение интервальных оценок сводится к многократным экспериментам с одной и той же моделью. При использовании генераторов реально случайных чисел при каждом эксперименте результаты будут разными, эти эксперименты можно проводить независимо друг от друга, т. е. параллельно. Поскольку в модели применяются генераторы не случайных, а псевдослучайных чисел (ПСЧ), то в экспериментах разные последовательности получают при разных значениях начальной инициализации генераторов ПСЧ.

Имитационное моделирование проводят с помощью библиотеки функций языка Си++ и многопоточного программирования, эксперименты распределяют по нескольким потокам. Таким образом, используют модель параллельных вычислений с общей памятью, ориентированную на многопроцессорные архитектуры класса SMP (Symmetric MultiProcessor) [16].

**Материалы и методы.** Краткое описание библиотеки функций языка Си++ для ИМ с примерами типовых моделей приведено в [4, 5]. Библиотека создана на основе похожей библиотеки на языке Паскаль и реализует подходы и принципы, заложенные в языке GPSS. Рассмотрим пример модели типовой СМО M/M/1 (одноканальная СМО с неограниченной очередью и экспоненциальным распределением значений времени обслуживания и времени между заявками). Пусть среднее время обслуживания 8 у. е. (условных единиц), среднее время между заявками 10 у. е. Листинг модели с комментариями приведен ниже:

```
#include "SimC.h" // Файл заголовков функций библиотеки
int main()
{
    Start(10000); // Начальная инициализация, параметр - начальное
значение счетчика завершения
    InitGenerate(1, 0); // Выход транзакта, параметры: номер собы-
тия и время выхода
    PQUEUE pQ_1; // Указатель на очередь 1
    NewQueue(pQ_1, (char *)"1"); // Создать очередь 1
    PFACILITY pF_1; // Указатель на канал 1
    NewFac(pF_1, (char *)"1"); // Создать канал 1
    double lambda = 1.0 / 8, nu = 1. / 10; // Интенсивности пото-
ков: входного и обслуживания
    while (TG1) // Моделирование прекращается, когда счетчик за-
вершения 0
    {
        Plan(); // Протяжка модельного времени
        switch (SysEvent)
        {
        case 1: Generate(RandExp(0.1, V1)); break; // Создать
транзакт
        case 2: Queue(pQ_1); break; //Занимаем очередь 1
        case 3: Seize(pF_1); break; //Занять канал 1
        case 4: Depart(pQ_1); break; //Освобождаем очередь 1
        case 5: Advance(RandExp(lambda, V2)); break; // Задержать
транзакт
        case 6: Release(pF_1); break; //Освободить канал 1
        case 7: Terminate(1); break; // Удалить транзакт, счетчик
завершения уменьшается на 1
        }
    }
    PrintAll();
    return 0;
}
```

Результаты моделирования приведены на скриншоте листинга (рис. 1).

Моделирование прекращается, когда обслужено 10 000 заявок. Рассмотрим оценку одного параметра — среднего времени пребывания в очереди. Для подобной системы известны аналитические формулы [17] расчета точного значения среднего времени пребывания в очереди, точное значение равно 32 у. е., в листинге значение оценки составляет 30,659.

Для повышения точности необходимо увеличить время моделирования (можно задать абсолютное время или ограничить время числом обслуженных заявок), что может потребовать существенных вычислительных ресурсов, особенно для более сложных моделей.

```
    СИСТЕМНОЕ ВРЕМЯ         : SysTime=     101333.368
    СИСТЕМНОЕ СОБЫТИЕ       : SYSEVENT=  7
   TRANS=NULL
    ВСЕГО ВЫПОЛНЕНО СОБЫТИЙ : EVENTALL= 77967
    ЗАТРАЧЕНО ВРЕМЯ         : REALTIME=        0.00 СЕК
    СРЕДНЕЕ ВРЕМЯ ИСПОЛНЕНИЯ СОБЫТИЯ : MEANTIME=    0.000000 СЕК/СОБ

СОБ.#    ВСЕГО   СОБ.#    ВСЕГО   СОБ.#    ВСЕГО   СОБ.#    ВСЕГО   СОБ.#    ВСЕГО
    1     10004      3     10000      5     10000      7     10000
    2     10004      4     10000      6     10000      8         0

                    ****************************
                    *                          *
                    *  -----   ОЧЕРЕДИ   -----  *
                    *                          *
                    ****************************

 ОЧЕРЕДЬ  ЧИСЛО  ВХОДОВ МАКС. ДЛИНА СРЕДН. ВР. ОЖ. СРЕДНЯЯ     % ВХОДОВ
          ------------  -----------  --------------   ДЛИНА     В ПУСТУЮ
          С 0 ВРЕМ.ОЖ.  ТЕКУЩ.ДЛИНА  БЕЗ УЧ. 0 ВХ.              ОЧЕРЕДЬ |
        1       10004          23           30.659      3.026    20.442
                 2045           4           38.532

                    ****************************
                    *                          *
                    *  ----    ПРИБОРЫ    ----  *
                    *                          *
                    ****************************

ПРИБОР      ЧИСЛО       СРЕДНЕЕ ВРЕМЯ    ЗАГРУЗКА      ЧИСЛО      СОСТОЯНИЕ
            ВХОДОВ       ОБРАБОТКИ                   ЗАХВАТОВ
        1   10000              8.018       0.791         0       FREE
```

**Рис. 1.** Скриншот листинга с результатами моделирования

Полученная оценка является точечной, часто требуется получать интервальные оценки, используя доверительные интервал и вероятность. Для этого необходимо проводить несколько разных измерений с разными результатами, получаемыми за счет случайности процесса. В библиотеке используются генераторы ПСЧ с полным периодом, длина периода $2^{32}$ [1]. Для получения других результатов приведенной модели с теми же исходными данными достаточно изменить значения начальной инициализации генераторов ПСЧ. Например, в модели вызов функции для экспоненциального распределения — RandExp(0.1, V1), второй параметр V1 — целая беззнаковая 32-разрядная переменная, определенная в библиотеке и задающая состояние линейного конгруэнтного генератора, начальное значение, равное единице, можно использовать для задания состояний генераторов свои переменные.

Следовательно, можно проводить независимые эксперименты с одной и той же моделью, но с разными начальными значениями генераторов ПСЧ. Такие эксперименты для уменьшения времени можно осуществлять

независимо друг от друга, т. е. при технической возможности запускать выполнение моделей на разных ядрах процессора. В программе на языке Си++ в рамках одного процесса запускаются разные потоки [18].

**Результаты и их обсуждение.** Отметим, что изначально библиотека языка Си++ не предназначалась для проведения экспериментов в параллельном режиме. Процесс моделирования заключался в том, что транзакты — динамические объекты (в библиотеке тип транзакта задан специальной структурой языка Си++), моделирующие заявки на обслуживание, перемещались из одного списка в другой. Существуют следующие списки транзактов: список текущих событий (активные транзакты), будущих событий (у каждого одноканального или многоканального устройства есть свой список транзактов, ожидающих обслуживания), списки пользователей. Другие объекты системы моделирования, одноканальные или многоканальные устройства, очереди, гистограммы также помещались каждая в свой список. Указатели на эти списки объявлены в библиотеке как глобальные переменные. Глобальными переменными также являлись SysTime (текущее модельное время) и TG1 (значение счетчика завершения, по аналогии с GPSS). Таким образом, в рамках одного процесса все переменные существовали в одном экземпляре, что не позволяло в разных потоках запускать независимые эксперименты с одной моделью.

Для организации параллельного моделирования выполнена простая модернизация — все функции и глобальные переменные библиотеки включены в класс под названием Simulation. В модели в первую очередь создается объект этого класса и далее вызываются методы для этого объекта.

*Получение интервальных оценок.* В ходе экспериментов оценим математическое ожидание (МО) некоторого случайного значения. Полагаем, что ее дисперсия неизвестна. При работе с доверительными интервалами используют два основных термина.

1. Уровень значимости $\alpha$ — вероятность, с которой оценка параметра не попадает в заданный интервал.

2. Уровень доверия $\beta = 1 - \alpha$ — вероятность того, что оценка параметра попадает в заданный интервал.

Для оценки МО используем формулу

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad (1)$$

где $\overline{x}$ — оценка МО; $x_i$ — значение случайной величины, полученное в $i$-м эксперименте; $n$ — число экспериментов.

Вместо неизвестного стандартного квадратичного отклонения (СКО) используем его оценку, полученную по формуле [19, 20]

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}. \tag{2}$$

Для практических расчетов СКО удобнее рассчитывать как

$$s = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}{n-1}}. \tag{3}$$

При большой выборке ($n > 30$) доверительный интервал для среднего имеет вид

$$\left[\bar{x} - \frac{s}{\sqrt{n}} z_\alpha,\ \bar{x} + \frac{s}{\sqrt{n}} z_\alpha\right], \tag{4}$$

где $z_\alpha$ — квантиль нормального распределения уровня $1 - \alpha/2$ (определяется по специальным таблицам математической статистики).

При небольшой выборке ($n \le 30$) доверительный интервал для среднего можно записать так:

$$\left[\bar{x} - \frac{s}{\sqrt{n}} t_\alpha(n-1),\ \bar{x} + \frac{s}{\sqrt{n}} t_\alpha(n-1)\right],$$

где $t_\alpha(n-1)$ — квантиль распределения Стьюдента уровня $1-\alpha/2$ со степенью свободы $n-1$ (определяется по таблицам).

*Реализация функций для многопоточных экспериментов на языке Си++.* Результаты моделирования, приведенные в листинге (см. рис. 1), реализуем в виде функции языка Си++, она будет возвращать оценку среднего времени пребывания в очереди:

```
int N_Start = 100000; // Максимальное число обслуженных заявок
double Model(uint32_t* pRand) // Реализация модели, параметр -
указатель на массив инициализаторов ГПС
{
    Simulation obj;
    double lambda = 0.1, nu = 1. / 8; // Интенсивности потоков
заявок и обслуживания
    uint32_t v1, v2;
    v1 = *pRand; v2 = *(pRand + 1); // Начальные значения ГПС
    obj.Start(N_Start); // Начальная инициализация, N_Start -
начальное значение счетчика завершения
    obj.InitGenerate(1, 0.0); // Выход транзакта на событие 1,
момент времени 0.0
    PQUEUE pQ_1; // Указатель на очередь 1
```

```
    obj.NewQueue(pQ_1, (char *)"Queue 1"); // Создать очередь 1
    PFACILITY pF_1; // Указатель на канал 1
    obj.NewFac(pF_1, (char *)"Facility 1"); // Создать канал 1
    while (obj.TG1)
    {
        obj.Plan(); // Протяжка модельного времени
        switch (obj.SysEvent)
        {
        case 1: obj.Generate(obj.RandExp(lambda, v1)); break; //
Создать транзакт
        case 2: obj.Queue(pQ_1); break; //Занимаем очередь 1
        case 3: obj.Seize(pF_1); break; //Занять канал 1
        case 4: obj.Depart(pQ_1); break; //Освобождаем очередь 1
        case 5: obj.Advance(obj.RandExp(nu, v2)); break; // Задер-
жать транзакт
        case 6: obj.Release(pF_1); break; //Освободить канал 1
        case 7: obj.Terminate(1); break; // Удалить транзакт,
счетчик завершения уменьшается на 1
        }
    }
    double Rezult = pQ_1->MTIME; // Результат - среднее время
пребывания в очереди
    obj.Clear(); // Освобождение памяти
    return Rezult;
 }
```

Параметр функции (указатель на массив целых чисел) начальных значений для инициализации генераторов ПСЧ у данной модели имеет два значения в этом массиве, так как создаются две независимых последовательности ПСЧ — длительности интервалов времени между заявками и времени обслуживания.

В одном потоке будет проводиться $n$ экспериментов с этой моделью, при этом потоковая функция имеет вид:

```
// Потоковая функция
double ManyExperiments(int n, double& Sum2, uint32_t* pRand) // n ис-
пытаний в потоке
{
    double Sum = 0;
    Sum2 = 0; // Для суммы квадратов
    int k = 0; // Счетчик псевдослучайных чисел
    for (int i = 0; i < n; i++) // n экспериментов в цикле
    {
        double Rezult = Model(pRand + k);  // Запуск модели, пере-
даем 2 значения ГПСЧ
        k += 2; // Пропускаем по 2 инициализатора ГПСЧ
        Sum += Rezult; // Сумма результатов для оценки математиче-
ского ожидания
        Sum2 += Rezult*Rezult; // Сумма квадратов для оценки дис-
персии
    }
    return Sum;// Результат - сумма результатов n экспериментов
}
```

В этой функции два выходных параметра:

1) сумма значений на выходе модели (функция Model) для оценки МО;

2) сумма квадратов этих значений для оценки СКО.

Сумма квадратов записывается в переменную Sum2, передаваемую по ссылке через параметр функции. Функция Model вызывается в цикле, в каждом вызове передаются разные значения для инициализации генератора ПСЧ, которые находятся в массиве, передаваемом через указатель pRand в третьем параметре потоковой функции.

Потоки создаются в функции:

```
// Функция для создания нескольких потоков
double ManyThreads(int n, int m, double& sko, uint32_t* pRand) //
Для многопоточного режима, n испытаний, m – потоков
{ // n - общее число экспериментов, m - число потоков
    int N1 = n / m; // Число испытаний в одном потоке, минимум
    int n1 = n % m; // Осталось число испытаний, не попавших
в потоки, разбрасываем по потокам
    vector<double> Sum2(m, 0); // Сюда потоки записывают суммы
квадратов
    vector<future<double>> fut; // Вектор для хранения future
(потоков)
    for (int i = 0; i < m; i++) // Цикл создания потоков
    {
        int num = N1; // Число испытаний в одном потоке
        if (i < n1) num++; // Учитываем оставшиеся испытания
        // Создаем поток
        fut.push_back(async(ManyExperiments, num, ref(sum2[i]),
pRand));
        pRand += 2 * num; // Изменяем указатель на псевдослучайные
числа
    }
    double Sum = 0; // Здесь МО
    for (int i = 0; i < m; i++) // Сбор результатов потоков
    {
        Sum += fut[i].get(); // Получаем результат потока
    }
    double Sum2 = accumulate(Sum2.begin(), Sum2.end(), 0); // Сум-
ма квадратов
    sko = sqrt((Sum2 - Sum * Sum / n) / (n - 1)); // СКО
    return Sum / n; // МО
}
```

В этой функции параметрами являются: требуемое число экспериментов; число потоков; ссылка на переменную, в которую записывается оценка СКО; указатель на массив с начальными значениями для инициализации генераторов ПСЧ. Для каждого эксперимента должны быть свои значения для инициализации генераторов ПСЧ. В данном примере для каждого эксперимента два значения, размерность массива с инициализаторами $2n$, возвращаемое значение — оценка МО оцениваемого параметра.

В функции эксперименты равномерно распределяются между потоками. Если число экспериментов не делится на число потоков без остатка, то остаток также равномерно распределяется между потоками. В некоторых потоках на один эксперимент больше. Получив результаты потоков, находим сумму оцениваемых значений и сумму их квадратов. Оценки МО и СКО находим по формулам (1) и (3).

*Результаты экспериментов.* Функция main, задающая проведение экспериментов в многопоточном режиме, имеет вид

```
int main()
{
    std::cout << "N Threads=" <<
std::thread::hardware_concurrency() << endl; // Число потоков,
которые могут выполняться одновременно
    int n = 100; // Число экспериментов
    int m = 12; // Число потоков
    srand(5); // Инициализация генератора ПСЧ
    vector<uint32_t> vRand(n * 2); // Вектор инициализаторов
для генераторов ПСЧ
    double Rezult; // Для оценки МО
    // Заполнение вектора инициализаторов для генераторов ПСЧ
    for (int i = 0; i < vRand.size(); i++) vRand[i] = rand();
    double sko; // СКО
    Rezult = ManyThreads(n, m, sko, vRand.data()); // Запуск моде-
лей в потоках
    cout << "MO=" << Rezult << " sko=" << sko << endl; // МО и СКО
    // Печать доверительного интервала для доверительной вероятно-
сти 0.9
    cout << "90 %: " << Rezult - 1.645 * sko / sqrt(n) << " - " <<
Rezult + 1.645 * sko / sqrt(n) << endl;
    return 0;
}
```

В функции число экспериментов 100, число потоков 12 (процессор поддерживает одновременное выполнение 12 потоков). Оцениваемый параметр — среднее время пребывания в очереди. Вектор инициализаторов генераторов ПСЧ заполняется с помощью стандартной функции rand(). Доверительный интервал для уровня доверия 0,9 считаем по (4). Квантиль нормального распределения уровня $1-\alpha/2 = 0{,}95$ находим по таблицам математической статистики, он равен 1,645. Результат работы (МО, СКО и 90%-ный доверительный интервал) следующий:

$$\text{МО} = 31{,}8749; \quad \text{СКО} = 1{,}02471;$$

$$90\ \%: 31{,}7064–32{,}0435.$$

Среднее время пребывания в очереди, рассчитанное на аналитической модели, равно 32 и попадает в доверительный интервал.

Измерена зависимость времени проведения экспериментов от числа используемых потоков. Измерения выполнены на ноутбуке с процессором AMD Ryzen 5 4600H, Radeon Graphics, 3000 МГц, ядер 6, логических процессоров 12. С учетом загрузки процессора фоновыми процессами операционной системы время выполнения при одинаковых параметрах может незначительно изменяться. В связи с этим проведено три теста, результаты в виде зависимостей времени выполнения экспериментов от числа используемых потоков в трех тестах приведены в таблице и на рис. 2.

**Зависимость времени выполнения экспериментов от числа используемых потоков**

| Число потоков | Время выполнения в тестах, мс | | | Число потоков | Время выполнения в тестах, мс | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | | 1 | 2 | 3 |
| 1 | 4541 | 4559 | 4546 | 11 | 678 | 701 | 691 |
| 2 | 2353 | 2323 | 2337 | 12 | 652 | 671 | 672 |
| 3 | 1612 | 1583 | 1639 | 13 | 689 | 698 | 678 |
| 4 | 1213 | 1284 | 1494 | 14 | 696 | 741 | 704 |
| 5 | 1048 | 1267 | 1426 | 15 | 674 | 698 | 707 |
| 6 | 926 | 1179 | 1189 | 16 | 688 | 672 | 676 |
| 7 | 858 | 1038 | 983 | 17 | 661 | 677 | 656 |
| 8 | 855 | 893 | 923 | 18 | 671 | 655 | 678 |
| 9 | 798 | 789 | 789 | 19 | 660 | 670 | 668 |
| 10 | 731 | 749 | 746 | 20 | 690 | 692 | 723 |

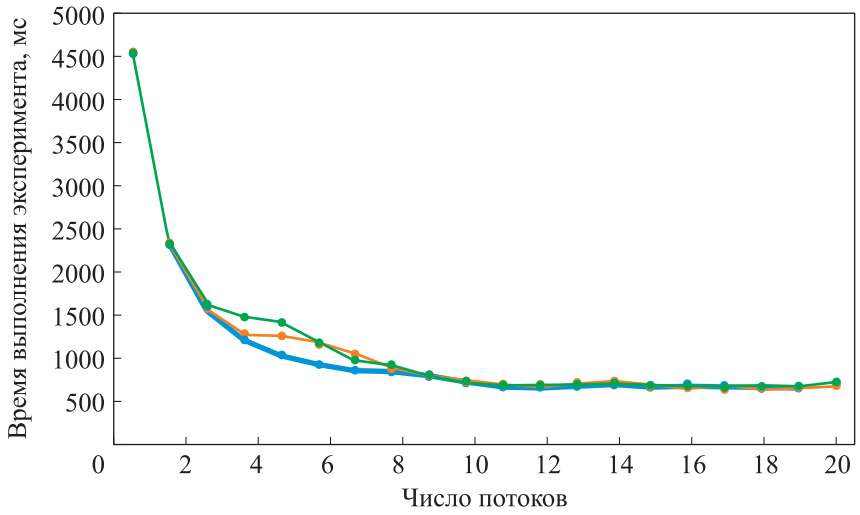



**Рис. 2.** Зависимости времени выполнения экспериментов от числа используемых потоков в тестах 1 (•), 2 (•), 3 (•)

**Заключение.** Отметим, что нет смысла использовать число параллельных потоков для моделирования большее, чем число потоков, одновременно выполняемых на конкретном процессоре (в проведенных экспериментах процессор поддерживал 12 потоков). При превышении этого числа параллельных потоков время моделирования перестает уменьшаться, при использовании максимального числа потоков, которые могут выполняться одновременно на процессоре, получено увеличение в ~ 7 раз ускорения выполнения имитационных экспериментов при 12 потоках, что не противоречит закону Амдала [21].

Рассмотрено применение параллельных вычислений для получения доверительных интервалов оцениваемых параметров при ИМ СМО с использованием библиотеки функция языка Си++. Поскольку получение доверительного интервала сводится к выполнению многократных независимых имитационных экспериментов с одной и той же моделью, то предложено эксперименты проводить в разных потоках и при этом использовать разные значения для инициализации генераторов ПСЧ в модели.

Начальная библиотека функций языка Си++ для ИМ СМО ориентирована на однопоточное выполнение из-за того, что основные системные переменные, например системное время и другие, были глобальными переменными языка Си++, т. е. существовали в единственном экземпляре. Для многопоточного выполнения библиотека модернизирована, все функции и глобальные переменные включены в класс. Отдельный эксперимент с моделью проводился с использованием объекта класса. Объекты классов не зависят друг от друга.

Использование многопоточного режима ИМ для получения доверительных интервалов позволило существенно уменьшить машинное время выполнения экспериментов, например, в ~ 7 раз при 12 параллельных потоках.

## ЛИТЕРАТУРА

[1] Рудаков И.В., Смирнов А.А. Исследование сложных дискретных систем на базе агентного метода. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2009, № 3 (76), с. 33–40. EDN: KXDRJH

[2] Постников В.М., Гребенников Н.А. Имитационное моделирование систем обработки информации. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2000, № 2 (39), с. 71–82.

[3] Рудаков И.В. Программный комплекс иерархического исследования сложных дискретных структур. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2012, № 1 (86), с. 121–124. EDN: OWBRWN

[4] Медведев Н.В., Быков А.Ю., Гришин Г.А. Имитационное моделирование систем массового обслуживания с использованием межплатформенной библиотеки функций языка Си++. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2005, № 4 (61), с. 85–93.

[5] Журавлев А.М., Медведев Н.В., Быков А.Ю. Использование межплатформенной библиотеки функций языка Си++ для реализации имитационных моделей типовых систем массового обслуживания. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2008, № 4 (73), с. 3–15. EDN: JXLLEL

[6] Andelfinger P., Uhrmacher A. Optimistic parallel simulation of tightly coupled agents in continuous time. *IEEE/ACM 25th DS-RT*, 2021.
DOI: https://doi.org/10.1109/DS-RT52167.2021.9576156

[7] Taylor L., Israelsen B., Zhang Z. Cycle and commute: rare-event probability verification for chemical reaction networks. *Proc. 23rd FMCAD,* 2023, pp. 284–293.
DOI: https://doi.org/10.34727/2023/isbn.978-3-85448-060-0_37

[8] Мещеряков Р.В., Моисеев А.Н., Демин А.Ю. и др. Применение параллельных вычислений в имитационном моделировании сетей массового обслуживания. *Известия Томского политехнического университета,* 2014, т. 325, № 5, с. 99–109.
EDN: TGGRZL

[9] Яркова О.Н., Москаленко Л.П. Параллельный алгоритм имитационного моделирования динамики объема груза на складе хранения. *Инженерный вестник Дона*, 2024, № 6 (114), с. 190–203. EDN: CIXSWZ

[10] Алексанян А.С., Аракелян С.М., Прокошев В.В. и др. Система имитационного моделирования работы сети на основе высокопроизводительных параллельных вычислений: архитектура и перспективы развития. *Информатизация образования и науки*, 2009, № 4, с. 88–101. EDN: KXEMKZ

[11] Юдин Е.Б. О распараллеливании вычислений при построении системы имитационного моделирования. *Динамика систем, механизмов и машин*, 2014, № 3, с. 247–250. EDN: SZFITJ

[12] Lucca N., Schepke C., Tremarin G. Parallel directives evaluation in porous media application: a case study. *31st PDP*, 2023, pp. 275–282.
DOI: https://doi.org/10.1109/PDP59025.2023.00050

[13] Wei S., Yang Z., Lou G., et al. Transmission line decoupling based parallel simulation method of an active distribution network. *IEEE iSPEC*, 2021, pp. 226–230.
DOI: https://doi.org/10.1109/iSPEC53008.2021.9735969

[14] Carreño M.A., Rodríguez M.D.F., Gers J.M. A simple application for executing dynamic simulations in parallel on cluster machines. *IEEE ISGT*, 2022.
DOI: https://doi.org/10.1109/ISGT50606.2022.9817560

[15] Колесников А.В., Басараб М.А., Ключарев П.Г. Анализ влияния протоколов передачи данных на энергопотребление NB-IoT-устройств. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2024, № 4 (149), с. 56–75. EDN: WHJCKD

[16] Ежова Н.А., Соколинский Л.Б. Обзор моделей параллельных вычислений. *Вестник Южно-Уральского государственного университета. Сер. Вычислительная математика и информатика*, 2019, т. 8, № 3, с. 58–91.
DOI: https://doi.org/10.14529/cmse190304

[17] Григорьев Ю.А., Плужников В.Л. Модель обработки запросов в параллельной системе баз данных. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2010, № 4 (81), с. 78–90. EDN: NCTAHD

[18] Бабкин А.А., Махов Н.М., Пономарев К.И. Многопоточность и параллельные вычисления в Qt Си++: QThreadPool и Qt Concurrent. *Научный лидер*, 2025, № 36 (237), с. 16–17. EDN: EFOAHM

[19] Боровков В.Е., Ключарёв П.Г., Денисенко Д.И. Методика оценивания результативности функционирования систем обнаружения веб-бэкдоров. *Информатика и автоматизация*, 2025, т. 24, № 1, с. 125–162. DOI: https://doi.org/10.15622/ia.24.1.6

[20] Басараб М.А., Шелухин О.И., Коновалов И.А. Оценка влияния трешолдинга на достоверность обнаружения аномальных вторжений в компьютерные сети статистическим методом. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2018, № 5 (122), с. 56–67. EDN: VMJYWC

[21] Амдал Д. Компьютерная архитектура и закон Амдала. *Открытые системы. СУБД*, 2014, № 2, с. 46–50.

**Быков Александр Юрьевич** — канд. техн. наук, доцент кафедры «Информационная безопасность» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

**Поремский Александр Романович** — аспирант кафедры «Информационная безопасность» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

**Просьба ссылаться на эту статью следующим образом:**

Быков А.Ю., Поремский А.Р. Оценивание доверительных интервалов при имитационном моделировании систем массового обслуживания с использованием параллельных вычислений и библиотеки на языке программирования Си++. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2026, № 3 (156), с. 85–101.
EDN: INOXYB

# ESTIMATION OF CONFIDENCE INTERVALS IN SIMULATION MODELING OF QUEUEING SYSTEMS USING PARALLEL COMPUTING AND A LIBRARY IN THE C++ PROGRAMMING LANGUAGE

**A.Yu. Bykov** abykov@bmstu.ru
**A.R. Poremskiy** poremskiyar@student.bmstu.ru

**BMSTU, Moscow, Russian Federation**

**Abstract**

The article considers an approach to obtaining interval estimates in the form of confidence intervals for the desired parameters of the queuing system on simulation models, which, as a rule, require significant computing resources. To reduce the time of simulation experiments, it is proposed to use parallel computing technology. Obtaining confidence intervals is based on multiple experiments with the same model, which can be performed independently of each other, that is, performed in parallel. When using random number generators, the results in each experiment with the same model will be different. To simulate this, pseudorandom number generators based on a linear congruent generator are initialized with different initial values in each experiment. Simulation modeling was performed using a special library of C++ language functions, the library was upgraded to work in multithreaded mode — all functions and global variables are included in the class, each experiment with the model was performed using its own class object. Experiments are evenly distributed between threads within a multithreaded program. An example of modeling a typical M/M/1 system is given, experimental results are presented, graphs of the dependence of simulation time on the number of flows are plotted, and conclusions are drawn

**Keywords**

*Multithreaded program, pseudorandom number generator, linear congruential generator, confidence probability, classes, objects, C++*

Received 10.03.2025
Accepted 04.09.2026
© Author(s), 2026

## REFERENCES

[1] Rudakov I.V., Smirnov A.A. Study of complex discrete systems on the basis of agent method. *Herald of the Bauman Moscow State Technical University. Series Instrument Engineering*, 2009, no. 3 (76), pp. 33–40 (in Russ.). EDN: KXDRJH

[2] Postnikov V.M., Grebennikov N.A. Technology of fast development of imitation models for information processing systems. *Herald of the Bauman Moscow State Technical University*. *Series Instrument Engineering*, 2000, no. 2 (39), pp. 71–82 (in Russ.). EDN: JXLIEL

[3] Rudakov I.V. Program complex of hierarchical study of complex discrete structures. *Herald of the Bauman Moscow State Technical University*. *Series Instrument Engineering*, 2012, no. 1 (86), pp. 121–124 (in Russ.). EDN: OWBRWN

[4] Medvedev N.V., Bykov A.Yu., Grishin G.A. Simulation of mass service systems using a cross-platform C++ function library. *Herald of the Bauman Moscow State Technical University*. *Series Instrument Engineering*, 2005, no. 4 (61), pp. 85–93 (in Russ.).

[5] Zhuravlev A.M., Medvedev N.V., Bykov A.Yu. Using cross-platform library of C++ functions for implementation of imitation models of standard queueing systems. *Herald of the Bauman Moscow State Technical University*. *Series Instrument Engineering*, 2008, no. 4 (73), pp. 3–15 (in Russ.). EDN: JXLIEL

[6] Andelfinger P., Uhrmacher A. Optimistic parallel simulation of tightly coupled agents in continuous time. *IEEE/ACM 25th DS-RT*, 2021.
DOI: https://doi.org/10.1109/DS-RT52167.2021.9576156

[7] Taylor L., Israelsen B., Zhang Z. Cycle and commute: rare-event probability verification for chemical reaction networks. *Proc. 23rd FMCAD*, 2023, pp. 284–293.
DOI: https://doi.org/10.34727/2023/isbn.978-3-85448-060-0_37

[8] Meshcheryakov R.V., Moiseev A.N., Demin A.Yu., et al. Using parallel computing in queueing network simulation. *Izvestiya Tomskogo politekhnicheskogo universiteta* [Bulletin of the Tomsk Polytechnic University], 2014, vol. 325, no. 5, pp. 99–109 (in Russ.).
EDN: TGGRZL

[9] Yarkova O.N., Moskalenko L.P. Parallel algorithm for simulating the dynamics of cargo volume in a storage warehouse. *Inzhenernyy vestnik Dona* [Engineering Journal of Don], 2024, no. 6 (114), pp. 190–203 (in Russ.). EDN: CIXSWZ

[10] Aleksanyan A.S., Arakelyan S.M., Prokoshev V.V., et al. A network simulation engine based on high-performance parallel computing: system design and development plans. *Informatizatsiya obrazovaniya i nauki* [Informatization of Education and Science], 2009, no. 4, pp. 88–101 (in Russ.). EDN: KXEMKZ

[11] Yudin E.B. About parallel computing in constructing simulation system. *Dinamika sistem, mekhanizmov i mashin* [Dynamics of System, Mechanisms and Machines], 2014, no. 3, pp. 247–250 (in Russ.). EDN: SZFITJ

[12] Lucca N., Schepke C., Tremarin G. Parallel directives evaluation in porous media application: a case study. *31st PDP*, 2023, pp. 275–282.
DOI: https://doi.org/10.1109/PDP59025.2023.00050

[13] Wei S., Yang Z., Lou G., et al. Transmission line decoupling based parallel simulation method of an active distribution network. *IEEE SPEC*, 2021, pp. 226–230.
DOI: https://doi.org/10.1109/iSPEC53008.2021.9735969

[14] Carreño M.A., Rodríguez M.D.F., Gers J.M. A simple application for executing dynamic simulations in parallel on cluster machines. *IEEE ISGT*, 2022.
DOI: https://doi.org/10.1109/ISGT50606.2022.9817560

[15] Kolesnikov A.V., Basarab M.A., Klyucharev P.G. Analysis of the data transfer protocols impact on the NB-IoT device power consumption. *Herald of the Bauman Moscow*

*State Technical University. Series Instrument Engineering*, 2024, no. 4 (149), pp. 56–75 (in Russ.). EDN: WHJCKD

[16] Ezhova N.A., Sokolinskiy L.B. Survey of parallel computation models. *Vestnik Yuzhno-Uralskogo gosudarstvennogo universiteta. Ser. Vychislitelnaya matematika i informatika* [Bulletin of the South Ural State University. Ser. Computational Mathematics and Software Engineering], 2019, vol. 8, no. 3, pp. 58–91 (in Russ.).
DOI: https://doi.org/10.14529/cmse190304

[17] Grigor'ev Yu.A., Pluzhnikov V.L. Model of transaction processing in database parallel system. *Herald of the Bauman Moscow State Technical University. Series Instrument Engineering*, 2010, no. 4 (81), pp. 78–90 (in Russ.). EDN: NCTAHD

[18] Babkin A.A., Makhov N.M., Ponomarev K.I. Multithreading and parallel computing in Qt C++: QThreadPool and Qt Concurrent. *Nauchnyy lider* [Scientific Leader], 2025, no. 36 (237), pp. 16–17 (in Russ.). EDN: EFOAHM

[19] Borovkov V.E., Klyucharev P.G., Denisenko D.I. Technique for assessing the effectiveness of the functioning of web backdoor detection systems. *Informatika i avtomatizatsiya* [Informatics and Automation], 2025, vol. 24, no. 1, pp. 125–162 (in Russ.).
DOI: https://doi.org/10.15622/ia.24.1.6

[20] Basarab M.A., Sheluhin O.I., Konovalov I.A. Assessment of the thresholding impact on reliability of anomaly detection in network traffic using statistical approach. *Herald of the Bauman Moscow State Technical University. Series Instrument Engineering*, 2018, no. 5 (122), pp. 56–67 (in Russ.). EDN: VMJYWC

[21] Amdal D. Computer architecture and Amdal's law. *Otkrytye sistemy. SUBD* [Open Systems. DBMS], 2014, no. 2, pp. 46–50 (in Russ.).

**Bykov A.Yu.** — Cand. Sc. (Eng.), Assoc. Professor, Department of Information Security, BMSTU (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

**Poremskiy A.R.** — Post-Graduate Student, Department of Information Security, BMSTU (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

**Please cite this article in English as:**

Bykov A.Yu., Poremskiy A.R. Estimation of confidence intervals in simulation modeling of queueing systems using parallel computing and a library in the C++ programming language. *Herald of the Bauman Moscow State Technical University, Series Instrument Engineering*, 2026, no. 3 (156), pp. 85–101 (in Russ.). EDN: INOXYB