О параллелизации метода Монте-Карло при решении некоторых метрологических задач

А.В. Степанов

ВНИИМ им. Д.И. Менделеева

Рассмотрены некоторые типовые метрологические задачи, для решения которых применим метод Монте-Карло. Показано, что его реализация зачастую допускает простую параллелизацию стандартными средствами используемого языка программирования (в статье – Java). Среди прочих, упомянуты такие задачи, как трансформирование распределений, получение оценок максимального правдоподобия, метрологическая аттестация алгоритмов обработки данных, получение критических значений статистических критериев.

Ключевые слова: метод Монте-Карло, параллелизм, многопоточность, неопределенность измерения, случайная величина, оценка параметров распределений, интервал охвата, коэффициент охвата.

1. Введение

Метод Монте-Карло, представляющий собой метод статистического моделирования, основанный на многократной генерации наборов случайных данных и последующем анализе результатов их обработки, широко используется при решении метрологических задач [1–4]. В качестве одного из применений можно упомянуть трансформирование распределений с целью оценивания неопределенности измерения, описываемого заданной математической моделью, рекомендованный к применению Приложением 1 к Руководству по выражению неопределенностей [5, 6]. Данный метод, в частности, особенно актуален, когда распределения входных величин обладают заметной асимметрией, или (и) модель измерения достаточно сложна (например, существенно нелинейна). Также можно упомянуть такие его преимущества, как отсутствие необходимости делать предположения о виде закона выходного распределения (при классическом подходе оно предполагается нормальным или t-распределением); возможность оценки разнообразных статистических характеристик распределения выходной величины (а не только среднеквадратического отклонения), и т.д. Другими примерами использования метода Монте-Карло являются получение критических значений при выводе критериев проверки статистических гипотез, а также оценка и сравнение мощностей и свойств статистических критериев при конечных объемах выборок [7]; метрологическая аттестация (т.е. исследование точностных свойств в рамках конкретной измерительной задачи) и сравнение характеристик алгоритмов обработки измерительных данных.

Развитие вычислительной техники и повышение ее доступности и производительности, широкое распространение программных библиотек обработки данных (разнообразных математических утилит, генераторов случайных чисел и т.д.), наличие подробной документации, специализированных статей, значительно увеличили доступность численного моделирования для исследователей. Становится возможным смещать фокус с классических алгоритмов обработки данных, постулирующих нормальность рассматриваемых распределений, теория применения которых хорошо проработана, на иные, иногда более релевантные, модели: например, рассматривать асимметричные распределения, учитывать естественные ограничения на множества допустимых значений, и т.д. При этом достаточно давно наблюдается тренд на параллелизацию проводимых численных экспериментов, связанный с распространением многоядерных и распределенных вычислительных систем [8, 9]. Отдельно следует упомянуть технологии проведения поточно-параллелельных вычислений на графических процессорах (GPU) [10], таких, как CUDA или ROCm (их использование предъявляет спе-

цифические требования как к аппаратному обеспечению, так и к подготовке исследователя, и в данной статье не они не рассматриваются).

Использование метода Монте-Карло допускает простую и естественную параллелизацию для решения многих вышеупомянутых задач. Ниже будут приведены примеры такой параллелизации с использованием механизмов поддержки многопоточности, когда каждый вычислительный эксперимент метода производится в отдельных потоках исполнения кода, после чего совместные результаты подвергаются постобработке. Указанный подход позволяет значительно ускорить проводящиеся вычисления при использовании многоядерных и многопроцессорных систем. Следующий пункт содержит описание данного подхода, с использованием конкретного языка программирования (Java); далее приведены примеры его применения для исследования свойств одного класса распределений. В конце статьи для некоторых примеров будут приведены цифры, демонстрирующие ускорение вычислений при использовании многопоточности.

2. Общий принцип организации кода

Схема моделирования, в общем случае, представляет собой многократно (порядка 10^6 раз и выше, в зависимости от требуемой точности результатов) повторяемые численные эксперименты (прогоны, trials) обработки сгенерированных данных, накопление и совместную постобработку результатов. Для многих рассматривавшихся приложений цикл обработки моделируемых данных, является относительно однородным, т.е. отдельные эксперименты имеют сходную вычислительную сложность.

Как уже было сказано выше, в данной работе предлагается рассматривать наиболее доступный подход, ориентированный на применение доступных многоядерных (многопроцессорных) вычислительных систем. При решении задач использовался использовался язык программирования Java (версии 8 и выше), тем не менее, данный выбор не принципиален (автор не ставил перед собой задачи сравнения быстродействия различных языков программирования и платформ). Дополнительным преимуществом, помимо личных предпочтений автора, был принцип WORA – «Write Once, Run Anywhere», так как реализованный функционал использовался на различных архитектурах (x86-64, Arm64), на машинах, имеющих от нескольких до нескольких десятков вычислительных ядер. Параллелизация достигалась путем использования механизмов многопоточности, поддерживаемых языком (распределенные вычисления рассматривались исключительно в виде эксперимента). Даже такой простейший подход давал заметный прирост производительности, используемый функционал показал достаточно хорошую масштабируемость при использовании на многоядерных платформах.

В простейшем случае структура приложения может быть описана следующим образом (рис. 1). Заметим, что использование именно Java Parallel Stream API, демонстрируемое в приведенном листинге, не является обязательным (можно выбрать множество способов реализации, основанных на использовании классов пакета `java.util.concurrent`), но код в данном виде выглядит особенно лаконично.

В качестве генератора случайных чисел предлагается использовать класс `ThreadLocalRandom`, который обеспечивает корректную генерацию псевдослучайных чисел при использовании многопоточности.

Безусловно, приведенный код допускает различные модификации. Например, в случае, если выходные данные алгоритма моделирования одномерны (рис. 2), то отпадает необходимость в полях для хранения промежуточных результатов (вообще говоря, от них можно избавиться и в случае многомерных выходных данных, это незначимый технический вопрос организации кода).

```
public class MCExample {

    private final MCParams params;
    private final int nTrials;
    private final double[] out_1;
    private final double[] out_2;
    ...

    public MCExample(MCParams params, int nTrials) {
        this.params = params;  // use to generate input data
        this.nTrials = nTrials;
        out_1 = new double[nTrials];
        out_2 = new double[nTrials];
        ...
    }

    private double[] generateInputData() { /*...*/ }

    private double dataProcessing1(double[] data) {/*...*/}
    private double dataProcessing2(double[] data) {/*...*/}

    private void trial(int i) {
        double data[] = generateInputData();
        out_1[i] = dataProcessing1(data);
        out_2[i] = dataProcessing2(data);
        ...
    }

    private void runTrialsInParallel() {
        IntStream.range(0, nTrials).parallel().forEach(this::trial);
    }

    private void postProcessing() {
        // post-processing of out_1, out_2, ...
    }

    public static void main(String[] args) {
        MCParams params = new MCParams(...);
        MCExample mc = new MCExample(params, 5_000_000);
        mc.runTrialsInParallel();
        mc.postProcessing();
    }
}
```

Рис. 1. Общая схема параллелизации метода Монте-Карло с использованием Parallel Stream API

```
    ...
    private double trial(int i) { /*...*/ }

    private double[] runTrialsInParallel() {
        return IntStream.range(0, nTrials).parallel().
                mapToDouble(this::trial).toArray();
    }

    private void postProcessing(double results[]) { /*...*/ }

    public static void main(String[] args) {
        MCParams params = new MCParams(...);
        MCExample mc = new MCExample(params, 5_000_000);
        double[] results = mc.runTrialsInParallel();
        mc.postProcessing(results);
    }
    ...
```

Рис. 2. Модификация общей схемы при одномерных выходных данных

Заметим, что пользователь имеет возможность управлять числом потоков, выделяемых для параллельной обработки данных, например, используя опцию `-Djava.util.concurrent.ForkJoinPool.common.parallelism` при запуске программы.

3. Примеры использования

Ниже будут приведены примеры применения предлагаемого подхода. Численные эксперименты, выполнявшиеся при написании работ [7, 11–14] (и некоторых других), проводились с его использованием. Заметим, что приведенные ниже примеры не претендуют на полноту описания возможных применений, разнообразие которых крайне велико, а лишь демонстрируют общий для подобного рода задач подход.

3.1. TSP-распределение

В то время как сложившиеся метрологические практики, как правило, предполагают нормальность (иногда – равномерность) рассматриваемых законов распределения, широкое распространение программного обеспечения открывает возможности использования иных семейств распределений для моделирования погрешностей измерений и нечеткой информации. В качестве модельного в дальнейших примерах будет рассмотрено семейство двусторонних степенных (Two-Sided Power, TSP) распределений [15, 16]. Плотность данного распределения является четырехпараметрической функцией, определнной следующим образом:

$$f(x) = f(x|a,m,b,p) = \begin{cases} \frac{p}{b-a}\left(\frac{x-a}{m-a}\right)^{p-1}, & a < x \le m, \\ \frac{p}{b-a}\left(\frac{b-x}{b-m}\right)^{p-1}, & m \le x < b, \\ 0, & \text{иначе.} \end{cases} \quad (1)$$

Носителем распределения является интервал (a,b). Параметр p положителен, при $p = 1, 2$ имеем равномерное и треугольное распределение, соответственно. При $p < 1$ график f имеет U-образную форму (распределение является двухмодальным), при $p > 1$ параметр m представляет собой моду распределения. Факт принадлежности случайной величины x данному распределению будем обозначать следующим образом: $x \in TSP(a,m,b,p)$.

Семейство TSP-распределений, несмотря на крайнюю простоту математического описания, достаточно разнообразно. В работе [16] рассмотрена диаграмма моментов данного распределения, и показано, что разнообразие распределений семейства сопоставимо со случаем Бета-распределением. Также, опираясь на различные критерии, можно аппроксимировать им то или иное непрерывное распределение на заданном отрезке.

Заметим, что функция распределения для (1) и обратная к ней имеют простое математическое выражение [16]:

$$F(y) = \begin{cases} \frac{m-a}{b-a}\left(\frac{x-a}{m-a}\right)^{p}, & a \le x \le m, \\ 1 - \frac{b-m}{b-a}\left(\frac{b-x}{b-m}\right)^{p}, & m \le x \le b, \end{cases}$$

$$F^{-1}(y) = \begin{cases} a + \sqrt[n]{y(m-a)^{p-1}(b-a)}, & 0 \le y \le \frac{m-a}{b-a}, \\ b - \sqrt[n]{(1-y)(b-m)^{p-1}(b-a)}, & \frac{m-a}{b-a} \le y \le 1, \end{cases}$$

что позволяет легко реализовать генератор случайных чисел на основе метода обратного преобразования (рис. 3).

```
...
private final double q = (m - a) / (b - a);
...
private double[] generateInputData() {

    double res[] = new double[sampleLength];
    for (int i = 0; i < sampleLength; ++i) {
        double u = ThreadLocalRandom.current().nextDouble();
        if (u < q) {
            res[i] = a + Math.pow(u *
                     Math.pow(m - a, p - 1) * (b - a), 1 / p);
        } else {
            res[i] = b - Math.pow((1 - u) *
                     Math.pow(b - m, p - 1) * (b - a), 1 / p);
        }
    }
    return res;
}
...
```

Рис. 3. Генератор выборок из TPS-распределения

Заметим также, что крайняя простота описания (1) позволяет, например, аналитически выразить границы интервалов охвата (которые в метрологической практике, как правило, служат мерой расширенной неопределенности [17]). Напомним, что отрезок (можно также рассматривать открытый интервал) $I_{P_0} = [c_1, c_2]$ называют интервалом охвата для некоторого непрерывного распределения, отвечающего вероятности (уровню) охвата P_0, если $P\{c_1 \le x \le c_2\} = P_0$, или, что то же самое, $F(c_2) - F(c_1) = P_0$, где F – функция распределения. Очевидно, что данное определение не является однозначным, рассматриваемое распределение для заданного уровня P_0 имеет множество различных интервалов охвата, включая кратчайший из них. Если рассматриваемый интервал охвата является двусторонним и вероятностно-симметричным, т.е.

$$F(c_1) = 1 - F(c_2) = \frac{1 - P_0}{2},$$

то из выражения для $F(x)$ следует, что

$$c_1 = a + (m - a)\sqrt[p]{\frac{1}{2}\frac{b-a}{m-a}(1 - P_0)}, \quad c_2 = b - (b - m)\sqrt[p]{\frac{1}{2}\frac{b-a}{b-m}(1 - P_0)} \tag{2}$$

(считаем, что $m \in [c_1, c_2]$). В случае симметричного распределения, положив $m = 0$, получим: $c_1 = -c_2$, и можно также вывести выражение для коэффициента охвата

$$K = \frac{c_2}{\sqrt{Var(x)}} = \left(1 - \sqrt[p]{1 - P_0}\right)\sqrt{\frac{1}{2}(p+1)(p+2)}.$$

Построение кратчайшего интервала охвата для TSP-распределения также является несложной вычислительной задачей.

С точки зрения метрологических приложений, иногда более естественным может выглядеть случай, когда пользователь вместо параметров m, p задает среднее значение случайной величины x_0 (которое ассоциируется с измеренным значением) и среднеквадратическое отклонение (СКО) u (которое ассоциируется с неопределенностью измерения), считая при этом диапазон значений известным. В этом случае параметры m, p могут быть выражены через x_0 (для симметричного распределения совпадает с m), u.

3.2. Пример 1. Трансформирование распределений

Рассмотрим частный случай задачи трансформирования распределений [5,6], когда все входные величины принадлежат семейству (1), симметричны, а выходная величина явля-

ется их линейной комбинацией:

$$y = \sum_{i=1}^{n} \alpha_i x_i, \quad x_i \in TSP(m_i - r_i, m_i, m_i + r_i, p_i).$$

Требуется выяснить, можно ли также аппроксимировать распределение выходной величины y симметричным TSP-распределением, с приемлемым качеством (в общем случае, очевидно, распределение y не принадлежит семейству TSP). В качестве критерия «приемлемости», с метрологической точки зрения, можно выбрать следующий: вероятность охвата $\hat{P}_0$, вычисленная для интервала охвата TSP-аппроксимации y, отвечающего заданному уровню вероятности P_0, не будет значимо (более чем на 1–2%) отличаться от P_0 в меньшую сторону для фактического распределения y (отличие в большую сторону не так критично, т.к. оно означает переоценку интервала охвата, а не его недооценку).

Легко проверить, что для симметричного распределения $TSP(m - r, m, m + r, p)$ дисперсия величины x имеет вид:

$$u^2 = Var(x) = \frac{2r^2}{(p+1)(p+2)}$$

(здесь u – обозначение для неопределенности, отождествляемой с СКО величины x). И обратно, рассматривая последнее равенство, как уравнение относительно p, можем выразить p через u (или, что то же самое, через $Var(x)$):

$$p = \frac{1}{2}\left(\sqrt{1 + \frac{8r^2}{u^2}} - 3\right) = \frac{1}{2}\left(\sqrt{1 + \frac{8r^2}{Var(x)}} - 3\right)$$

(отбросили отрицательный корень квадратного уравнения, т.к. $p > 0$). Математическое ожидание и дисперсия величины y имеют вид:

$$m_y = E(y) = \sum_{i=1}^{n} \alpha_i m_i, \quad u_y^2 = Var(y) = \sum_{1}^{n} \alpha_i^2 u_i^2, \quad u_i^2 = \frac{2r_i^2}{(p_i+1)(p_i+2)}.$$

Далее, очевидно, $|y - m_y| < r_y = \sum_{i=1}^{n} |\alpha_i| r_i$. Таким образом, предполагая принадлежность величины y семейству TSP ($y \in TSP(m_y - r_y, m_y, m_y + r_y, p_y)$), для величины p_y получим оценку:

$$\hat{p}_y = \frac{1}{2}\left(\sqrt{1 + \frac{8r_y^2}{u_y^2}} - 3\right), \quad \Longrightarrow \quad \hat{K}_y = \left(1 - \sqrt[\hat{p}_y]{1 - P_0}\right)\sqrt{\frac{1}{2}(\hat{p}_y + 1)(\hat{p}_y + 2)},$$

и оценка интервала охвата величины y, полученная в предположении о приемлемости аппроксимации величины y TSP-распрелением, а также искомое значение $\hat{P}_0$, имеют вид:

$$\hat{I}_{P_0} = [m_y - \hat{K}_y u_y,\, m_y + \hat{K}_y u_y], \quad \hat{P}_0 = P\left\{y \in \hat{I}_{P_0}\right\} = P\left\{|y - m_y| \le \hat{K}_y u_y\right\}$$

(подчеркнем, что вероятность $\hat{P}_0$ является эмпирической, получаемой в результате моделирования).

Рассмотрим конкретный пример. Пусть $m_0 = m_1 = 0,\ r_1 = r_2 = 1,\ p_1 = 0.8,\ p_2 = 2.7,$ $\alpha_1 = 0.3,\ \alpha_2 = 0.7$, тогда $m_y = 0,\ r_y = 1,\ p_y \approx 3.19$, и $K_y \approx 2.01$. Проведенное моделирование показывает, что $\hat{P}_0 \approx 0.957 > P_0$, аппроксимацию можно считать приемлемой. Аналогично, изменим значение параметра p_2, все остальные оставив прежними: пусть $p_2 = 1$. Тогда $p_y \approx 1.71,\ K_y \approx 1.85$, и $\hat{P}_0 \approx 0.956 > P_0$, построенная TSP-аппроксимация снова является приемлемой.

С точки зрения программной реализации, получение трансформированного распределения величины y является задачей, подлежащей параллелизации; метод (3) генерации входных данных при этом придется очевидным образом доработать, чтобы различные значения выборки принадлежали разным TSP-распределениям.

Заметим, что решение данной задачи в случае, когда распределения входных величин x_i асимметричны, решение данной задачи представляет еще больший интерес. Тем не менее, опустим этот вопрос здесь, во избежание громоздких математических выкладок.

3.3. Пример 2. Получение оценок максимального правдоподобия измеряемого значения при многократных неравноточных измерениях

Обработка неравноточных измерительных данных является одной из распространенных метрологических задач. Предположим, что имеется n результатов измерений $x_i, i = \overline{1,n}$ некоторой величины, каждому из которых приписано свое распределение $TSP(a_i, m_i, b_i, p_i)$. Предположим, что неизвестное измеряемое значение x_0 ассоциируется с математическим ожиданием для каждого распределения, и известны максимальные отклонения $r_{1,i}$, $r_{2,i}$ от x_0, т.е. $a_i = x_0 - r_{1,i}$, $b_i = x_0 + r_{2,i}$ $(r_{1,i}, r_{2,i} > 0)$. Выражение для m_i в этом случае имеет вид:

$$m_i = x_0 + \frac{r_{1,i} - r_{2,i}}{p_i - 1}, \quad p_i \neq 1$$

(положим $m_i = x_0 + \frac{r_{2,i} - r_{1,i}}{2}$ при $p_i = 1$). В итоге, $x_i \in TSP(x_0 - r_{1,i}, x_0 + \frac{r_{1,i} - r_{2,i}}{p_i - 1}, x_0 + r_{2,i}, p_i)$ (в случае, когда измеряемое значение ассоциируется с наиболее часто встречаемым значением, т.е. модой распределения, считаем, что $x_i \in TSP(x_0 - r_{1,i}, x_0, x_0 + r_{2,i}, p_i)$).

Для получения оценки $\hat{x}_0$ измеряемого значения x_0 предлагается использовать метод максимального правдоподобия [13], т.е.

$$\hat{x}_0 = \arg\max_{x_0} L(x_0), \quad L(x_0) = \prod_{i=1}^{n} f\left(x_i \middle| x_0 - r_{1,i}, x_0 + \frac{r_{1,i} - r_{2,i}}{p_i - 1}, x_0 + r_{2,i}, p_i\right),$$

здесь $L(x_0)$ – функция правдоподобия, а f задается формулой (1). Получение оценок $\hat{x}_0$ является задачей, подлежащей параллелизации; метод (3) снова придется очевидным образом изменить, чтобы получать выборку из разных распределений. Итоговая выборка $\{\hat{x}_{0,j}\}_1^N$ (`results`, в обозначениях листинга (2)) подвергается метрологической пост-обработке: в качестве окончательной оценки значения измеряемой величины x_0 вычисляется среднее значение выборки, в качестве мер неопределенности и расширенной неопределенности x_0 вычисляются выборочное СКО и интервал охвата для заданного уровня вероятности. Полученные оценки могут отличаться от таковых для стандартных процедур, предполагающих нормальность распределений (для достаточно больших n они, как правило, близки).

3.4. Пример 3. Метрологическая аттестация алгоритма получения оценок максимального правдоподобия параметров TSP-распределения

Применение метода Монте-Карло позволяет оценить смещения оценок максимального правдоподобия для параметров TSP-распределения, получаемых по выборкам конечной длины. Обозначим $\{x_i\}_1^n$ выборку длиной n из этого распределения, а $x_{(1)} \leq \cdots \leq x_{(n)}$ – соответствующий вариационный ряд. В этом данные оценки имеют вид [15, 16]:

$$\hat{m} = x_{(i_0)}, \quad \hat{p} = -\frac{n}{\log M(i_0)}, \quad i_0 = \arg\max_i M(i), \quad M(i) = \prod_{j=1}^{i-1} \frac{x_{(j)} - a}{x_{(i)} - a} \prod_{j=i+1}^{n} \frac{b - x_{(j)}}{b - x_{(i)}}.$$

Таким образом, в обозначениях кода (1), массивы `out_1`, `out_2` предназначены для сбора оценок $\hat{m}, \hat{p}$ (а получение этих оценок подлежит параллелизации). Пост-обработка включает

в себя вычисление смещения оценок от истинных значений, заданных при моделировании, и соответствующих СКО.

Предположим, что $p > 1$. Моделирование показывает, что в случае асимметричного TPS-распределения оценки параметров m, p являются смещенными. Например, при $a = 0$, $b = 1$ (случай так называемого стандартного TPS-распределения, STSP) и $m > 0.5$ математическое ожидание $E\hat{m}$ смещено, по сравнению со значением m, использовавшимся при моделировании, влево (в сторону нуля) тем сильнее, чем ближе исходный параметр m к единице; величина $E\hat{p}$ смещена относительно p вверх. В табл. 1 ниже приведены относительные смещения $\delta\hat{m}$, $\delta\hat{p}$ (в процентах) оценок параметров m, p, для выборок длиной $n = 20, 50$. Применение смещенных оценок максимального правдоподобия $\hat{m}$, $\hat{p}$ для оценивания интервала охвата по формулам (2) приводят к его недооценке при малых n, особенно в случае значений p, близких к единице, что влечет необходимость коррекции полученных оценок $\hat{m}$, $\hat{p}$. В то же время, при длинах выборки более 50 (и особенно ближе к 100) недооценка интервала охвата не является практически значимой (пренебрежимо мала).

Таблица 1. $\delta\hat{m}$, $\delta\hat{p}$, %

n	20						50					
m	0.75		0.90		0.95		0.75		0.90		0.95	
p	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$
1.5	–7.4	15.8	–10.0	15.3	–10.6	14.9	–2.6	5.9	–3.9	5.9	–4.4	5.8
2.0	–2.0	12.3	–3.2	12.6	–3.7	12.5	–0.7	4.5	–1.1	4.6	–1.3	4.7
2.5	–1.0	11.0	–1.6	11.4	–2.0	11.6	–0.4	4.0	–0.6	4.2	–0.7	4.3
3.0	–0.7	10.4	–1.0	10.8	–1.3	11.1	–0.3	3.8	–0.4	3.9	–0.4	4.0
3.5	–0.5	10.0	–0.7	10.4	–0.9	10.8	–0.2	3.7	–0.3	3.8	–0.3	3.9
4.0	–0.4	9.7	–0.6	10.1	–0.7	10.5	–0.2	3.6	–0.2	3.7	–0.2	3.8
4.5	–0.3	9.5	–0.5	9.9	–0.6	10.3	–0.1	3.5	–0.2	3.6	–0.2	3.7
5.0	–0.3	9.4	–0.4	9.8	–0.5	10.2	–0.1	3.5	–0.2	3.6	–0.2	3.7

3.5. Пример 4. Подбор наиболее подходящего распределения из семейства по выборочным данным

Метод выбора распределения из конечного набора непрерывных распределений, наилучшим образом согласующегося с имеющимися экспериментальными данными, рассмотрен в работе [18]. Применительно к семейству (1), он выглядит следующим образом: для имеющейся выборки длины n экспериментальных данных $\{x_i\}_1^n$ из конечного множества $S = \{(a, b, m, p)\}$ наборов параметров распределений данного семейства выбирается набор $\hat{s} \in S$, доставляющий минимум функционалу следующего вида:

$$\hat{s} = \left(\hat{a}, \hat{m}, \hat{b}, \hat{p}\right) = \arg\min_{s \in S} \sum_{1}^{n} \left(F_s^{-1}\left(\frac{i}{n}\right) - x_{(i)} \right)^2 .$$

Здесь $\{x_{(i)}\}_1^n$ – вариационный ряд для рассматриваемой выборки, а F_s – функция распределения, отвечающая конкретному набору параметров s.

В качестве критерия метрологической аттестации, в частности, можно рассмотреть вероятность выбора верного распределения из множества предложенных вариантов. Код, ответственнй за выбор распределения, наилучшим образом согласующегося со сгенерированной выборкой, при этом является телом метода `trial` (в обозначениях рис. 1), а эмпирические вероятности выбора того или иного распределения основаны на счетчиках выбора предложенных альтернатив (т.е. массив `out` содержит столько счетчиков, сколько рассматривается вариантов выбора). Пусть, например, $n = 200$, $a = -1$, $b = 1$,

$S = \{m = -0.1, 0, 0.1\} \times \{p = 1.5, 2, 2.5\}$, и входные (моделируемые) данные принадлежат распределению $TSP(-1, 0, 1, 2)$ (т.е. $s_0 = (0, 2)$). Вероятности выбора наборов $s \in S$ приведены в табл. 2.

Таблица 2. Значения $P(s)$

$s = (m, p)$	$P(s)$
$(-0.1, 1.5)$	0.001
$(-0.1, 2.0)$	0.053
$(-0.1, 2.5)$	0.005
$(0.0, 1.5)$	0.017
$(\mathbf{0.0}, \mathbf{2.0})$	**0.823**
$(0.0, 2.5)$	0.072
$(0.1, 1.5)$	0.001
$(0.1, 2.0)$	0.026
$(0.1, 2.5)$	0.002

Естественно, что вероятность $P\{s = s_0\}$ (выделена жирным шрифтом) существенным образом зависит от длины выборки n (увеличивается с ростом n). Для $n = 250, 300$, например, она составляет 0.882 и 0.920, соответственно. Заметим также, что преимуществом рассмотренного метода является возможность расширять множество альтернатив распределениями из других семейств (например, проводившиеся численные эксперименты показали, что распределения семейства TSP достаточно уверенно отделяются от усеченных нормальных распределений, при достаточной длине выборки). Также можно рассматривать его в качестве критерия подбора аппроксимирующего распределения из семейства (1) для произвольного непрерывного распределения.

В работе [11] задача подбора TSP-распределения по выборочным данным рассмотрена более подробно (в частности, показано, что при подборе единственного параметра степени p для симметричного распределения данный метод дает схожие оценки, по сравнению с методом максимального правдоподобия).

3.6. Пример 5. Оценивание интервалов охвата по выборочным размаху и СКО

Рассмотрим вопрос об оценивании интервалов охвата симметричного TSP-распределения по выборочным данным. В качестве оценки интервала охвата, отвечающего уровню вероятности P_0, рассмотрим отрезки вида $I = [c_1, c_2] = [\bar{x} - C_d d, \bar{x} + C_d d]$, где $\bar{x}$ – выборочное среднее, d – выборочная статистика, характеризующая меру разброса данных, например, СКО S или размах R. Воспользуемся методом Монте-Карло для выбора коэффициента C_d, исходя из условия $P\{F(c_2) - F(c_1) \geq P_0\} = P_1$, т.е. предполагается, что рассматриваемый отрезок содержит с заданной вероятностью P_1 некоторый интервал охвата для данного уровня вероятности P_0. Вычисление значений $F(c_2) - F(c_1)$ для пробных значений константы C_d подлежит параллелизации. Ниже в табл. 3, 4 в качестве примера приведены некоторые значения оценок коэффициентов C_R, C_S при $P_0 = 0.95$ и $P_1 = 0.95, 0.99$ (заметим, что величина C_d возрастает с ростом P_1, а также убывает с ростом длины выборки n, для каждого рассматриваемого распределения).

В рассмотренных примерах обе оценки обеспечивают, в среднем, сопоставимую длину отрезка $[c_1, c_2]$.

Таблица 3. Значения C_R, C_S ($P_0 = 0.95$, $P_1 = 0.95$)

	n	p						
		0.75	1	1.5	2	3	5	7
C_r	5	1.82	1.84	1.90	1.99	2.14	2.34	2.44
	7	1.20	1.25	1.33	1.39	1.49	1.63	1.70
	10	0.90	0.95	1.01	1.05	1.13	1.22	1.26
	15	0.74	0.76	0.80	0.83	0.88	0.94	0.97
C_s	7	3.23	3.40	3.63	3.82	4.14	4.51	4.71
	10	2.65	2.82	3.04	3.21	3.48	3.77	3.94
	15	2.32	2.45	2.65	2.80	3.02	3.27	3.40
	20	2.16	2.28	2.46	2.61	2.81	3.02	3.13

Таблица 4. Значения C_R, C_S ($P_0 = 0.95$, $P_1 = 0.99$)

	n	p						
		0.75	1	1.5	2	3	5	7
C_r	5	3.23	3.00	2.96	3.07	3.33	3.68	3.87
	7	1.80	1.78	1.82	1.91	2.07	2.29	2.40
	10	1.19	1.22	1.27	1.34	1.45	1.59	1.66
	15	0.88	0.91	0.95	0.99	1.07	1.16	1.20
C_s	7	4.88	4.87	5.00	5.23	5.70	6.31	6.65
	10	3.50	3.64	3.85	4.06	4.43	4.88	5.13
	15	2.78	2.93	3.15	3.33	3.63	3.97	4.16
	20	2.50	2.63	2.83	2.99	3.26	3.55	3.71

3.7. Пример 6. Получение критических значений для статистических критериев

Примеры применения метода Монте-Карло в данном случае крайне разнообразны. Можно еще раз упомянуть получение критических значений для критериев однородности дисперсий выборок из TSP-распределений [7].

Здесь же рассмотрим простейший критерий согласованности данных. Предположим, что у исследователя имеется массив накопленных данных из симметричного TPS-распределения (выборка длины n). Предположим, что имеется новый массив данных из того же самого распределения длины $k < n$. Рассмотрим статистику $C_{k,n} = \frac{R_k}{S_n}$, где R_k – размах второй выборки, а S_n – выборочное СКО первой. Соответствующие критические значения для некоторых значений параметров p, k, n (параметр масштаба не имеет значения) и уровня вероятности 0.95 приведены в табл. 5.

Таблица 5. Критические значения $C_{k,n}$

n	k	p				
		1.0	1.5	2	3	5
50	7	3.39	3.75	3.99	4.29	4.60
	10	3.49	3.91	4.20	4.59	4.99
100	7	3.33	3.68	3.91	4.20	4.49
	10	3.42	3.82	4.11	4.48	4.85

В качестве другого тривиального примера можно рассмотреть использование выборочного коэффициента асимметрии $G_1 = m_3 \cdot m_2^{-3/2}$, где m_i – i-й центральный выборочный момент, для проверки гипотезы о симметричности распределения (ожидаем, что его абсолютное значение будет ниже некоторого порога, с вероятностью P_0). Например, критические значения величины $|G_1|$ при $n = 100$ и $P_0 = 0.95$ составляют 0.303, 0.385, 0.549 для $p = 2, 3, 5$, соответственно.

В обоих приведенных выше примерах вычисления статистик параллелизовались (при этом в первом примере требовалось генерация двух выборок различных длин на эксперимент).

3.8. Ускорение при использовании многопоточности

Репозиторий [19] содержит несколько примеров кода, соответствующих описанным задачам. Все они носят демонстрационный характер, на практике вычисления, как правило, более объемны. В таблице ниже приведены данные, позволяющие оценить ускорение при использовании многопоточности, по сравнению с последовательным (для оценивания последнего комментировался метод `parallel`). Вычисления проводились на ноутбуке Huawei MateBook D16, с процессором Intel Core i5-12450H (8 ядер / 12 потоков, Hyper-Threading), под управлением OS Windows 11.

Таблица 6. Ускорение, полученное для некоторых примеров

Пример	Время выполнения кода, сек.		Ускорение
	последовательное	многопоточное	
2	1376.11	264.13	5.21
3	486.04	45.07	10.78
4	1296.71	154.65	8.38
6	85.80	8.38	10.24

Заметим, что ускорение для примеров 3, 6 близко к идеальному, т.е. эти задачи очень хорошо масштабируемы. Пример 4 также демонстрируют очень хороший результат, пример 2 – достаточно средний. Естественно, что при использовании большего числа ядер можно добиться более впечатляющих результатов (к сожалению, конкретные значения коэффициента ускорения, полученные при работе на более производительных многоядерных системах, не были сохранены автором).

4. Заключение

Современные версии популярных языков программирования (C++, Java, Go, Python, ...) содержат мощную и, в то же время, синтаксически лаконичную поддержку механизмов параллелизации исполнения кода (в частности, поддержку многопоточности), использование которой зачастую не требует экспертных навыков работы с таким кодом от разработчиков прикладного (научного) программного обеспечения. Данный инструментарий хорошо документирован, снабжен многочисленными примерами использования и, в случае необходимости, легко может быть освоен в сжатые сроки, т.е. порог вхождения при его использовании на описанном (относительно тривиальном) уровне перестает быть традиционно высоким. С другой стороны, широкое распространение получают персональные компьютеры с достаточно большим количеством ядер (на данный момент – 12, 14, 16 и выше); в случае необходимости, не слишком сложно получить доступ к машинам и с бо́льшим количеством (48, 64, ...) вычислительных ядер. Существует достаточно обширный класс алгоритмически не слишком сложных и вычислительно однородных (хорошо масштабируемых) задач, для которых сочетание указанных факторов позволяет значительно (в разы, иногда – на порядок и более, в зависимости от используемого аппаратного обеспечения) ускорить процесс математического моделирования и обработки результатов, даже не прибегая к использованию специализированной вычислительной техники и соответствующих библиотек, что повышает скорость и досупность проведения вычислительных экспериментов, заметно ускоряет поиск ошибок и валидацию полученных результатов, а также позволяет добиться повышения точности результатов за счет увеличения числа испытаний метода Монте-Карло. Безусловно, использование распределенных вычислений позволило бы добиться дальнейшего ускорения экспериментов, но оно уже накладывает на исследователя и его работодателя некоторые издержки как по организации и поддержке соответствующей вычислительной инфраструктуры, так и финансовые расходы на ее приобретение (или аренду) и обслуживание. Использование же встроенных механизмов поддержки многопоточности позволяет быстро получить заметный эффект уже на этапе прототипирования или выполнения исследовательских задач в инициативном порядке (например, при подготовке научных публикаций) в случае, когда у работника отсутствуют как доступ к высокопроизводительным вычислительным мощностям в своей организации, так и достаточно сложные задачи, для решения которых требуются такие мощности (далеко не каждый численный эксперимент требует наличия суперкомпьютера, многие из рассмотренных задач вполне могут быть решены на достаточно производительном персональном компьютере с многоядерным процессором в сжатые сроки). При этом, безусловно, стоит подчеркнуть, что существует множество вычислительно сложных исследовательских задач, когда описанного подхода совершенно недостаточно, и требуется использование совсем иной вычислительной техники (суперкомпьютеров и вычислительных кластеров).

Литература

1. Harris P., Cox M. On a Monte Carlo method for measurement uncertainty evaluation and its implementation // Metrologia. 2014. Vol. 51, no. 4. P. S176–S182. DOI: 10.1088/0026-1394/51/4/S176.

2. van der Veen A., Cox M. Getting started with uncertainty evaluation using the Monte Carlo method in R // Accreditation and Quality Assurance. 2021. Vol. 26. P. 129–141. DOI: 10.1007/s00769-021-01469-5.

3. Crowder S., Delker C., Forrest E., Martin N. Monte Carlo Methods for the Propagation of Uncertainties // Introduction to Statistics in Metrology. Springer, 2020. P. 153–180. DOI: 10.1007/978-3-030-53329-8_8.

4. Zhang J. Modern Monte Carlo methods for efficient uncertainty quantification and propagation: A survey // WIREs Computational Statistics. 2020. Vol. 15, no. 5. DOI: 10.1002/wics.1539.

5. Evaluation of measurement data – Supplement 1 to the «Guide to the expression of uncertainty in measurement» – Propagation of distributions using a Monte Carlo method / Joint Committee for Guides in Metrology, JCGM 101:2008.

6. ГОСТ 34100.3.1 Неопределенность измерения. Часть 3. Руководство по выражению неопределенности измерения. Дополнение 1: Трансформирование распределений с использованием метода Монте-Карло. Стандартинформ, 2018.

7. Stepanov A.V., Chunovkina A.G. On testing of the homogeneity of variances for two-sided power distribution family // Accreditation and Quality Assurance. 2023. Vol. 28. P. 129–137. DOI: 10.1007/s00769-022-01525-8.

8. J. Rosenthal. Parallel computing and Monte Carlo algorithms // Far East Journal of Theoretical Statistics. 2000. Vol. 4, no. 2. P. 207–236.

9. Maigne L., Hill D., Calvat P. et al. Parallelization of monte Carlo simulations and submission to a grid environment // Parallel Processing Letters. 2004. Vol. 14, no. 2. P. 177–196. DOI: 10.1142/S0129626404001829.

10. Некрасов К.А., Поташников С.И., Боярченков А.С., Купряжкин А.Я. Метод Монте-Карло на графических процессорах: учебное пособие. Екатеринбург: Изд-во Урал. ун-та, 2016. 60 с.

11. Stepanov A.V., Chunovkina A.G. On choosing two-sided power distribution for measurement data // Advanced Mathematical and Computational Tools in Metrology and Testing XIII Series on Advances in Mathematics for Applied Sciences, 2024. Vol. 94. P. 268–277. DOI: 10.1142/9789819800674_0025.

12. Степанов А.В., Чуновкина А.Г. Об одном методе подбора закона распределения из семейства TSP и его свойствах // Материалы международной Воронежской зимней математической школы. Воронеж, ВГУ, 2024. С. 238–241.

13. Степанов А.В., Чуновкина А.Г. О применении TSP-распределения при анализе многократных неравноточных результатов измерений // Уфимская осенняя математическая школа - 2024: материалы международной научной конференции, 2024. Т. 2. С. 308–309.

14. Степанов А.В., Чуновкина А.Г. Об оценке параметров асимметричного TSP распределения и его применении // Вероятностные методы в дискретной математике. Тезисы докладов XI международной Петрозаводской конференции, 2024. Петрозаводск: КарНЦ РАН, 2024. С. 106–108.

15. Van Dorp J.R., Kotz S. The standard two-sided power distribution and its properties // The American Statistician. 2002. Vol. 56, no. 2. P. 90–99.

16. Kotz S., van Dorp J.R. Beyond Beta: Other Continuous Families of Distributions with Bounded Support and Applications. World Scientific Publishing, 2004.

17. Stoudt S., Pintar A., Possolo A. Coverage Intervals // Journal of Research of the National Institute of Standards and Technology. 2021. Vol. 126. DOI: 10.6028/jres.126.004.

18. Тырсин А.Н. Метод подбора наилучшего закона распределения непрерывной случайной величины на основе обратного отображения // Вестник ЮУрГУ. Серия: Математика. Механика. Физика. 2017. Т. 9, № 1. С. 31–38.

19. `https://github.com/stepanov17/pavt_examples`