ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени М.В.ЛОМОНОСОВА»

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА ТЕОРИИ ВЕРОЯТНОСТЕЙ

КУРСОВАЯ РАБОТА

специалиста

МЕТОД МОНТЕ-КАРЛО В ТЕОРИИ РИСКА

Выполнил студент

309 группы

Власов Никита Андреевич

_____

подпись студента

Научный руководитель:

профессор Фалин Геннадий Иванович

_____

подпись научного руководителя

Москва

2020

Оглавление

Введение

Цель настоящей работы – систематически и подробно изложить основные принципы метода Монте-Карло. При этом я буду следовать разделам 2.1.1, 2.2.1, 2.4.1, 2.4.2, 3.1, 3.2 книги [1] и разделам 4, 5 статьи [3].

В 4 главе будут приведены примеры использования метода Монте-Карло для реальных задач, разработаны и реализованы соответствующие алгоритмы решения на языке программирования Python, построены таблицы, доказывающие эффективность метода Монте-Карло.

В своей работе я хочу разобрать следующие задачи:

1. Научиться генерировать равномерно распределенные случайные числа на интервале $[0,1]$ с помощью численных алгоритмов;
2. Научиться получать случайные числа в соответствии с заданным распределением;
3. Рассмотреть общую схему метода Монте-Карло;
4. Применить метод Монте-Карло для решения задач по теории риска;
5. Разработать программу, реализующую метод Монте-Карло.

Объектом исследования является сам метод Монте-Карло и его применения.

Методами исследования являются методы теории вероятностей для исследования случайных величин и методы теории программирования для построения программных средств.

Глава 1. Генерация случайных чисел

§1. Как мы получаем случайные числа?

Стохастическое моделирование и особенно метод Монте-Карло используют случайные величины. Таким образом, становится необходимой способность генерировать случайные числа в соответствии с заданным распределением. Главная проблема состоит в том, чтобы **найти числа, которые действительно случайны и непредсказуемы**. Конечно, бросать кости слишком медленно для большинства приложений, поскольку обычно **требуется много случайных чисел**.

Случайные числа для моделирования по методу Монте-Карло обычно генерируются численным алгоритмом. Это делает последовательность чисел детерминированной, поэтому эти случайные числа часто называют **псевдослучайными числами**. Но если мы смотрим на них, не зная алгоритма, они кажутся случайными. И во многих статистических тестах они ведут себя как настоящие случайные числа.

Прежде всего, нам нужно научиться генерировать равномерно распределенные случайные числа на интервале $[0,1]$. Теоретически, включены ли 1 или 0 или нет, не имеет значения, так как вероятность для одного числа равна нулю. Но мы должны знать, включены ли в выбранный диапазон нуль и единица, поскольку это может привести к сбоям программы, как, например, ln(0).

Затем эти равномерно распределенные случайные числа можно будет преобразовать в случайные числа из определённого распределения (например, из экспоненциального или нормального распределения) с помощью определённых методов (об этом будет рассказано далее).

§2. Линейный конгруэнтный генератор

Линейный конгруэнтный генератор (ЛКГ, LCG) был одним из первых ГСЧ. Линейные генераторы были впервые введены Лемером (1949) [5] и были очень популярны в течение многих лет.

Алгоритм линейного конгруэнтного генератора

$$s_{n+1} = (as_n + c) \ mod \ m, n \in N ,$$

где

- $m \in N \setminus \{0\}$ называется **модуль,**
- $a \in N$ называется **множитель,** $a < m$,
- $c \in N$ называется **приращение**, $c < m$,
- $s_0 \in N$ **начальное значение**, $s_0 < m$.

Числа в интервале $[0,1)$ получаются по формуле

$$u_n = \frac{s_n}{m}$$

Часто модуль m берут простым числом. Затем все вычисления производятся в конечном поле Z_m. Предпочтительными модулями являются простые числа Мерсенна, то есть простые числа вида $2^k - 1$. Например, $2^{31} - 1 = 2147483647$, самое большое 32-разрядное целое число со знаком. Иногда в качестве модуля выбирается степень 2, потому что вычисления можно производить быстрее, используя двоичную структуру компьютеров. Но тогда младшие биты случайных чисел сильно скоррелированы, особенно последний бит состояний s_n либо постоянен, либо строго чередуется между 0 и 1. Это означает, что все наши случайные числа s_n либо четны, либо нечетны, либо у нас есть постоянное чередование чисел: сначала четное, потом нечетное и т.д.

Если период такого генератора равен m или m-1 (в случае c = 0), то мы называем это свойство **полным периодом**, так как это максимально возможная длина периода. Для достижения длинного периода целесообразно выбрать очень большое число в качестве модуля. Затем нужно выбрать приращение и множитель, чтобы создать генератор с полным периодом. Если c = 0, m - простое число, a - простой корень по модулю m, то генератор имеет полный период. Для $c \neq 0$ критерии полного периода более сложны (см., например, [4]): m и c взаимно просты, каждое простое число, которое делит m, также должно делить $a-1$; если m делится на 4, то $a-1$также должен иметь делитель 4.

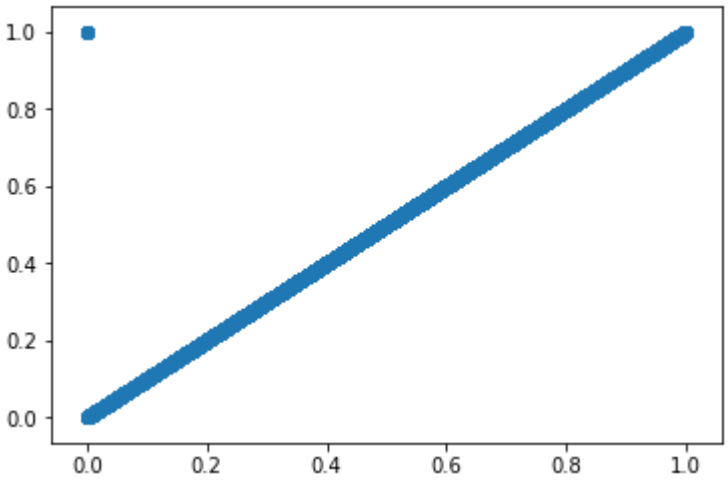
Недостатки линейного конгруэнтного генератора

Хотя ЛКГ просты в реализации и быстры, у них есть множество недостатков:

- существует высокая последовательная корреляция между случайными числами, поэтому они не очень хорошо имитируют случайность. Например, если множитель a мал по сравнению с модулем m, то за очень малым случайным числом всегда следует еще одно малое случайное число. Это означает, что редкие события располагаются слишком близко друг к другу.
- Рассмотрим множество всех возможных t-мерных векторов, заполненных последовательными случайными числами, $\Psi_t := \{(u_1, u_2, ..., u_t) \mid s_0 \in S\} \subseteq [0,1)^t$, где $S = \{0,1,...,m-1\}$ - это конечное множество состояний. Если бы наши псевдослучайные числа действительно были случайными и независимыми, то t-мерный гиперкуб был бы заполнен равномерно, без какой-либо прослеживаемой структуры. Но структура множества Ψ_t очень регулярна - все точки лежат на равноудаленных, параллельных гиперплоскостях. Это называется **решетчатой структурой** ЛКГ.

Пример: Рассмотрим ЛКГ $s_{n+1} = (1 \cdot s_n + 1) \ mod \ 3331$. Этот "ГСЧ" имеет полный период, то есть ρ = 3331. Но Ψ_2 состоит только из одной "гиперплоскости" и одной единственной точки (см. рисунок 1, код программы для получения этого рисунка приведён в Приложении 1).

Рисунок 1



Улучшение ЛКГ

Преимущества линейных конгруэнтных генераторов заключаются в том, что они очень быстры, не требуют много памяти, просты в реализации и легко понятны. Всё ещё существуют приложения, где достаточно хороших представителей этого типа генератора.

- Нельзя избежать решётчатой структуры, поэтому при работе с ЛКГ следует выбирать вариант со многими гиперплоскостями. Ключом к решению этой проблемы является спектральный тест (см. [4]). Он вычисляет наибольшее расстояние $1/l_t$ между двумя последовательными гиперплоскостями для семейства гиперплоскостей. Часто значение l_t вычисляется для нескольких Ψ_t, $t \in N$. Это дает нам приблизительное представление о количестве гиперплоскостей в Ψ_t. Чем больше гиперплоскостей в Ψ_t и чем меньше $1/l_t$, тем лучше генератор. Правило гласит, что $l_t \leq 1+a^2$ для случая $c = 0$, поэтому множитель a не должен быть слишком мал.
- Существует возможность перетасовать последовательность случайных чисел, в конечном итоге с помощью другого ЛКГ (или любого другого генератора случайных чисел) (см. [4]). *j*-е значение исходной последовательности не является *j*-м выходом, вместо этого оно удерживается в положении ожидания. *j*-е значение последовательности или значение другого ГСЧ решает, какая позиция ожидания освобождается, и эта позиция снова заполняется случайным числом. Этот метод удаляет части последовательной корреляции в последовательности случайных чисел.
- Вообще, мы можем совместить выход двух (или даже больше) различных ЛКГ. Затем мы имеем новый тип ГСЧ, называемый **комбинированным ГСЧ**. Существует два различных метода объединения генераторов:

  Метод 1: Если $u_n^{(1)} \in [0,1)$ выходное значение ЛКГ1, $u_n^{(2)}$ выходное значение ЛКГ2, то $u_n = (u_n^{(1)} + u_n^{(2)}) \ mod \ 1$ является выходным значением для комбинированного генератора.

  Метод 2: Если $s_n^{(1)} \in S$ *n*-тое целое значение в последовательности ЛКГ1, $s_n^{(2)}$ *n*-тое целое значение в последовательности ЛКГ2, то скомбинируем $s_n = (s_n^{(1)} + s_n^{(2)}) \ mod \ m_1$, где m_1 это модуль ЛКГ1 с условием $m_1 > m_2$.

Рекомендованные ЛКГ.

ЛКГ хорошего качества можно, например, использовать для перетасовки последовательностей или задания начального значения для других ГСЧ. Комбинированный генератор достаточно хорош для мелкомасштабного моделирования. В таблице приведены некоторые рекомендуемые варианты выбора параметров.

Авторы	ЛКГ(m,a,c)	Период генератора
Парк и Миллер	ЛКГ($2^{31}-1,16807,0$)	$2^{31}-2$
Фишман и Мур	ЛКГ($2^{31}-1,950706376,0$)	$2^{31}-2$
Фишман	ЛКГ($2^{31}-1,48271,0$)	$2^{31}-2$
Л’Экуаер	ЛКГ($2^{31}-249,40692,0$)	$2^{31}-250$
Ran2(комбинированный генератор)	ЛКГ1(2147483563,40014,0) ЛКГ2(2147483399,40692,0)	2.3×10^{18}

Глава 2. Моделирование случайных величин

Предположим, что мы нашли хороший ГСЧ, который генерирует случайные числа, которые равномерно распределены на интервале $[0,1]$. Наш следующий шаг заключается в преобразовании этих случайных чисел в неоднородные случайные числа, например, нормально распределенные, χ^2-распределенные или распределённые по закону Пуассона. Часто желаемое распределение может быть только аппроксимировано. Конечно, приближение должно быть максимально точным. Метод преобразования должен быть эффективным, быстрым и не должен использовать слишком много памяти. Одним из таких методов преобразования является метод инверсии.

§1. Метод инверсии

Лучший способ преобразования случайных чисел - это метод инверсии, поскольку он сохраняет их структуру случайных чисел. Но, если скорость имеет значение, этот метод может быть не лучшим, так как часто используются сложные функции или приходится работать с алгоритмами аппроксимации.

Метод инверсии основан на следующем простом факте. Предположим, что случайная величина X имеет функцию распределения F, где F строго возрастает и непрерывна. Тогда существует обратная функция F^{-1}. Для равномерно распределенной величины $U \sim U[0,1)$ случайная величина $F^{-1}(U)$ имеет то же распределение, что и X, т. е. она имеет функцию распределения F, так как

$$P(F^{-1}(U) \le x) = P(U \le F(x)) = F(x)$$

Если функция распределения F не является строго возрастающей или непрерывной, мы можем определить общую обратную функцию следующим образом

$$F^{\leftarrow}(u) := min\{x \mid F(x) \ge u\}$$

С помощью этой обобщенной обратной функции мы можем сформулировать метод инверсии.

Алгоритм (метод инверсии)

Пусть F это одномерная функция распределения.

1. Выбрать равномерно распределенное случайное число из интервала $(0,1)$
2. Получить случайное число с функцией распределения F через $x = F^{\leftarrow}(u)$

Пример. Экспоненциальное распределение

Экспоненциальное распределение подходит для измерения продолжительности жизни или моделирования радиоактивного распада. Если случайная величина X является экспоненциально распределенной с коэффициентом $\lambda > 0$, то её функция распределения равна

$$F(x) = 1 - e^{-\lambda x} \text{ для } x \ge 0,$$

а обратная функция для F следующая:

$$F^{-1}(u) = -\frac{\ln(1-u)}{\lambda} \text{ для } 0 \le u < 1$$

Тогда случайная величина $y = -\ln(1-u)/\lambda$, где $u \sim U[0,1)$, экспоненциально распределена с коэффициентом λ. На практике чаще используют $-\ln(u)/\lambda$, так как случайная величина $(1-U)$ имеет то же распределение, что и U.

§2. Метод принятия-отклонения (Acceptance-rejection method)

Бывает, что обратная функция F^{-1} слишком сложна или существуют только ее аппроксимации. Тогда генерация случайных чисел с помощью метода принятия-отклонения может быть намного быстрее и даже проще. Алгоритм принятия-отклонения может быть построен всякий раз, когда у нас есть функция плотности.

Предположим, что мы хотим смоделировать случайную величину *X* с плотностью *f(x)* и функцией распределения *F(x)*. Тогда мы ищем другую случайную величину *Y* с плотностью *g(y)*, значения которой легко могут быть получены путём преобразования равномерно распределённых случайных чисел и которая обладает свойством

$$f(x) \le Cg(x), \quad x \in R \ \ or \ \ x \in R^d,$$

где *C* – постоянная, $1 \le C < \infty$. Функцию *g* часто называют плотностью сравнения или мажорирующей функцией. Действительно, это всегда может быть достигнуто при некоторой простой, подходящей плотности и большой постоянной *C*. В случае ограниченной плотности с компактным носителем мажорирующая функция может быть построена путем выбора равномерной плотности на носителе.

Алгоритм. Метод принятия-отклонения.

Пусть даны плотности *f* и *g* с условием $f(x) \le Cg(x)$.

1. Сгенерировать равномерно распределенное случайное число *u* из интервала $[0,1]$.
2. Сгенерировать случайное число *y* из распределения, заданного плотностью g.
3. Если $u \le f(y)/(Cg(y))$, то принять *y* в качестве нового случайного числа *x* с плотностью распределения *f*. Иначе отклонить его и перейти к шагу 1.

Доказательство работы алгоритма:

Воспользовавшись тем, что вероятность произведения двух событий A и B равна

$$P(A,B) = P(A \mid B) \cdot P(B)$$

вычислим вероятность принятия случайной величины *X* с функцией плотности распределения *g(x)* и функцией распределения *G(x)*, но уже при условии, что она меньше некого заданного параметра *x*:

$$P\left(U \le \frac{f(X)}{Cg(X)} \mid X \le x\right) = \frac{P\left(U \le \frac{f(X)}{Cg(X)}, X \le x\right)}{P\left(X \le x\right)} = \frac{1}{G(x)} \int_{-\infty}^{x} P\left(U \le \frac{f(t)}{Cg(t)}\right) g(t)dt =$$

$$= \frac{1}{CG(x)} \int_{-\infty}^{x} f(t)dt = \frac{F(x)}{CG(x)}$$

Заметим, что

$$P\left(U \le \frac{f(X)}{Cg(X)}\right) = E[P(U \le \frac{f(X)}{Cg(X)} \mid X)] = E[\frac{f(X)}{Cg(X)}] = \int \frac{f(t)}{Cg(t)} g(t)dy = \frac{1}{C}$$

И тогда по формуле Байеса:

$$P\left(X \le x \,|\, U \le \frac{f(X)}{Cg(X)} \right) = P\left(U \le \frac{f(X)}{Cg(X)} \,|\, X \le x \right) \cdot \frac{P(X \le x)}{P\left(U \le \frac{f(X)}{Cg(X)} \right)} = \frac{F(x)}{CG(x)} \cdot \frac{G(x)}{1/C} = F(x)$$

чтд

Для получения одного случайного числа с распределением *f* нам нужно более одного равномерно распределенного случайного числа. Скорость этого ГСЧ определяется временем генерации случайного числа *y*, временем вычисления *f (y)* и константой *C*, поскольку

$$P(U \le \frac{f(Y)}{Cg(Y)}) = \frac{1}{C}.$$

Таким образом, 1/*C* дает нам вероятность принятия. Для быстрого алгоритма константа *C* должна быть как можно ближе к 1. В этом случае принимаются почти все случайные числа, и не слишком много величин *u* и *y* генерируются впустую.

Если функция *f* слишком трудоемка для оценки, мы можем использовать функции сжатия q_1, q_2 с условием

$$q_1(x) \le f(x) \le q_2(x) < Cg(x),$$

которые могут быть вычислены намного быстрее. Если $u \le q_1(y) / Cg(y)$, то y может быть немедленно принято. Иначе, если $u > q_2(y) / Cg(y)$, то y может быть немедленно отклонено. Только в том случае, если оба случая неприменимы, следует оценить функцию *f*.

Пример. Стандартное нормальное распределение.

Рассмотрим сначала распределение случайной величины *X*, которая равна абсолютному значению стандартной нормальной величины, $X = |Z|, \;\; Z \sim N(0,1)$. Тогда *X* имеет плотность вероятности

$$f(x) = \sqrt{\frac{2}{\pi}} e^{-x^2/2}, \quad x \ge 0$$

Функция экспоненты в плотности напоминает нам об экспоненциальном распределении с плотностью $g(y) = e^{-y}$, из которого мы уже умеем получать случайные числа с помощью метода инверсии: $u \sim U[0,1) \Rightarrow y = -ln(u) \sim exp(1)$. Теперь попробуем найти константу *C* такую, что

$$\frac{f(x)}{g(x)} \cdot \frac{1}{C} = \sqrt{\frac{2}{\pi}} e^{x - x^2/2} \cdot \frac{1}{C} \le 1$$

Это функция имеет максимум в точке *x*=1, поэтому выберем $C = \sqrt{\frac{2e}{\pi}}$, вероятность принятия равна 0.76. На заключительном этапе мы можем преобразовать наше принятое случайное число в нормальное случайное число, присвоив ему случайный знак, определяемый другим независимым равномерно распределенным случайным числом.

Поскольку не менее трёх случайных чисел необходимо только для одного нормального случайного числа, вероятность принятия не очень близка к 1. Кроме того, необходимо использовать экспоненциальную функцию, поэтому этот метод очень медленный и поэтому не является стандартным методом для генерации нормальных случайных величин.

§3. Полярный метод

Основная идея этого метода заключается в моделировании не одной, а двух независимых стандартных гауссовских величин Z_1, Z_2 и рассмотрении пары $Z=(Z_1,Z_2)$ как точки на координатной плоскости.

Алгоритм. Полярный метод

1. Сгенерировать случайную равномерно распределённую точку $X=(X_1,X_2)$ на квадрате $-1\le x\le 1,-1\le y\le 1$ – её можно моделировать как пару $(2U_1-1,2U_2-1)$, где величины U_1,U_2 независимы и равномерно распределены на $(0,1)$.
2. Если эта точка не попала внутрь единичного круга, т.е. неравенство $X_1^2+X_2^2<1$ не выполнено, отбросить её и снова выполнить пункт 1.
3. Получить пару случайных чисел, имеющих стандартное нормальное распределение, по формулам

$$\begin{cases} Z_1 = X_1\sqrt{-2\dfrac{\ln(X_1^2+X_2^2)}{X_1^2+X_2^2}} \\ Z_1 = X_2\sqrt{-2\dfrac{\ln(X_1^2+X_2^2)}{X_1^2+X_2^2}} \end{cases}$$

Доказательство работы алгоритма.

Квадрат расстояния от точки $Z=(Z_1,Z_2)$ до начала координат O, $\rho^2=Z_1^2+Z_2^2$, имеет распределение хи-квадрат с двумя степенями свободы, т.е. экспоненциальное распределение со средним 2:

$$P(\rho^2\le x)=1-e^{-x/2}$$

Перейдём в полярные координаты. Для этого введём в рассмотрение угол $\varphi\in[0;2\pi)$, который вектор $\overrightarrow{OZ}$ образует с положительным направлением оси абсцисс.

Поскольку плотность распределения точки Z, $f(x_1,x_2)$, является произведением стандартных плотностей $\frac{1}{\sqrt{2\pi}}e^{-\frac{x_1^2}{2}}$ и $\frac{1}{\sqrt{2\pi}}e^{-\frac{x_2^2}{2}}$, она даётся формулой $f(x_1,x_2)=\frac{1}{2\pi}e^{-\frac{x_1^2+x_2^2}{2}}$. Поэтому $f(x_1,x_2)$ сохраняет постоянное значение, равное $\frac{1}{2\pi}e^{-\frac{r^2}{2}}$, на окружности $x_1^2+x_2^2=r^2$. Следовательно, случайный угол φ равномерно распределён на промежутке $[0;2\pi)$.

Итак, мы получили, что случайные величины $e^{-\frac{\rho^2}{2}}$, $\frac{\varphi}{2\pi}$ равномерно распределены на $(0;1)$. Покажем, что эти величины независимы.

Найдём двумерную функцию распределения $\mathrm{P}(\rho^2\le x,\varphi\le t)$ (при $x>0,0\le t<2\pi$). Неравенства $\rho^2\le x,\varphi\le t$ задают на координатной плоскости сектор D круга с центром в начале координат и радиусом $R=\sqrt{x}$, ограниченный положительным направлением оси

абсцисс и лучом $x_2 = tx_1$. Вероятность события $\rho^2 \le x, \varphi \le t$ равна интегралу от плотности $f(x_1, x_2) = \frac{1}{2\pi} e^{-\frac{x_1^2 + x_2^2}{2}}$ по этой области:

$$\mathrm{P}(\rho^2 \le x, \varphi \le t) = \frac{1}{2\pi} \iint_D e^{-\frac{x_1^2 + x_2^2}{2}} dx_1 dx_2$$

Переходя к полярным координатам, получим:

$$\mathrm{P}(\rho^2 \le x, \varphi \le t) = \frac{1}{2\pi} \int_0^t \left(\int_0^{\sqrt{x}} e^{-\frac{r^2}{2}} r dr \right) dt = (1 - e^{-x/2}) \cdot \frac{t}{2\pi}$$

Эта формула означает, что случайные величины ρ^2 и φ независимы, а, значит, независимы и случайные величины $e^{-\frac{\rho^2}{2}}$, $\frac{\varphi}{2\pi}$.

Получили систему:

$$\begin{cases} e^{-\frac{\rho^2}{2}} = U_1 \\ \frac{\varphi}{2\pi} = U_2 \end{cases}$$

где U_1 и U_2 независимые равномерно распределенные на (0,1) случайные величины.

Отсюда,

$$\begin{cases} \rho^2 = -2 \ln U_1 \\ \varphi = 2\pi U_2 \end{cases}$$

Перейдём обратно к декартовым координатам:

$$\begin{cases} Z_1 = \rho \cos\varphi = -\sqrt{2 \ln U_1} \cos(2\pi U_2) \\ Z_1 = \rho \sin\varphi = -\sqrt{2 \ln U_1} \sin(2\pi U_2) \end{cases} \qquad (*)$$

Избавимся от тригонометрических функций в формулах.

Возьмём случайную (равномерно распределённую) точку $X = (X_1, X_2)$ на квадрате $-1 \le x \le 1, -1 \le y \le 1$. Если эта точка не попала внутрь единичного круга, т.е. неравенство $X_1^2 + X_2^2 < 1$ не выполнено, отбросим её и выберем ещё одну такую точку (независимо от первой). Первая точка, попавшая внутрь единичного круга, равномерно распределена на нём.

Покажем это. Если случайная точка *X* равномерно распределена на области *D*, то для любой подобласти $A \subset D$ мы имеем: $P(X \in A) = \frac{\text{площадь } A}{\text{площадь } D}$

Выделим в *D* подобласть *D** и рассмотрим условное распределение точки *X*, при условии, что она попала в эту подобласть: $P(X \in A \mid X \in D^*)$ (здесь $A \subset D^*$ – подобласть выделенной области *D**). Непосредственно по определению условной вероятности мы имеем:

$$P(X \in A \mid X \in D^*) = \frac{P(X \in A, X \in D^*)}{P(X \in D^*)} = \frac{P(X \in A)}{P(X \in D^*)} = \frac{\text{площадь } A / \text{площадь } D}{\text{площадь } D^* / \text{площадь } D} = \frac{\text{площадь } A}{\text{площадь } D^*}$$

что и означает равномерность условного распределения.

Итак, точка $X=(X_1,X_2)$ равномерно распределена на единичном круге. Поэтому и угол ψ, образуемый радиус-вектором $\overrightarrow{OX}$ с положительным направлением оси абсцисс, имеет равномерное распределение (на промежутке $[0;2\pi)$). Действительно, для $t\in(0;2\pi)$ неравенство $\psi \le t$ задаёт на координатной плоскости сектор единичного круга, ограниченный положительным направлением оси абсцисс и лучом $x_2=tx_1$. Поскольку пара (X_1,X_2) равномерно распределена на единичном круге, вероятность события $\psi \le t$ равна отношению площади этого сектора к площади единичного круга:

$$P(\psi \le t)=\frac{t/2}{\pi}=\frac{t}{2\pi},$$

что и означает равномерную распределённость ψ на промежутке $[0;2\pi)$.

Теперь случайную (равномерно распределённую) точку $(\cos\psi,\sin\psi)$ на единичной окружности можно получить с помощью формул: $\cos\psi=\frac{X_1}{\sqrt{X_1^2+X_2^2}}$, $\sin\psi=\frac{X_2}{\sqrt{X_1^2+X_2^2}}$.

Рассмотрим расстояние $s=\sqrt{X_1^2+X_2^2}$ от начала координат случайной точки (X_1,X_2) на единичном круге. Покажем, что случайная величина s^2 не зависит от ψ и равномерно распределена на (0,1).

Для доказательства этого факта найдём двумерную функцию распределения $P(s^2\le x,\psi\le t)$ (при $0<x<1, 0\le t<2\pi$). Неравенства $s^2\le x,\psi\le t$ задают на координатной плоскости сектор $D_{\sqrt{x};t}$ круга с центром в начале координат и радиусом $R=\sqrt{x}$, ограниченный положительным направлением оси абсцисс и лучом $x_2=tx_1$. Поскольку пара (X_1,X_2) равномерно распределена на единичном круге, вероятность события $s^2\le x,\psi\le t$ равна отношению площади сектора $D_{\sqrt{x};t}$ к площади единичного круга:

$$\mathrm{P}(s^2\le x,\psi\le t)=\frac{tx/2}{\pi}=x\cdot\frac{t}{2\pi}$$

Эта формула означает, что случайные величины s^2 и ψ независимы, причём s^2 имеет равномерное распределение на (0,1), а ψ – равномерное распределение на промежутке $[0;2\pi)$. Поэтому положим в формулах (*) $U_1=s^2$ и $U_2=\frac{\psi}{2\pi}$

В результате получим формулы

$$\begin{cases} Z_1=-\sqrt{2\ln U_1}\cos(2\pi U_2)=-\sqrt{2\ln s^2}\cos(\psi)=X_1\sqrt{-2\frac{\ln(X_1^2+X_2^2)}{X_1^2+X_2^2}} \\ Z_1=-\sqrt{2\ln U_1}\sin(2\pi U_2)=-\sqrt{2\ln s^2}\sin(\psi)=X_2\sqrt{-2\frac{\ln(X_1^2+X_2^2)}{X_1^2+X_2^2}} \end{cases}$$

чтд

Замечание. Этот метод требует проведения процедуры отбрасывания точек, не попадающих в единичный круг, что, в свою очередь, требует на каждом шаге пары величин, равномерно распределённых на (0,1). Вероятность попадания внутрь единичного круга равна $\frac{\pi}{4}$, так что до получения точки, которая нас устроит, потребуется в среднем $\frac{4}{\pi}$

проб. Это число близко к единице и потому этот этап не намного увеличивает время вычислений.

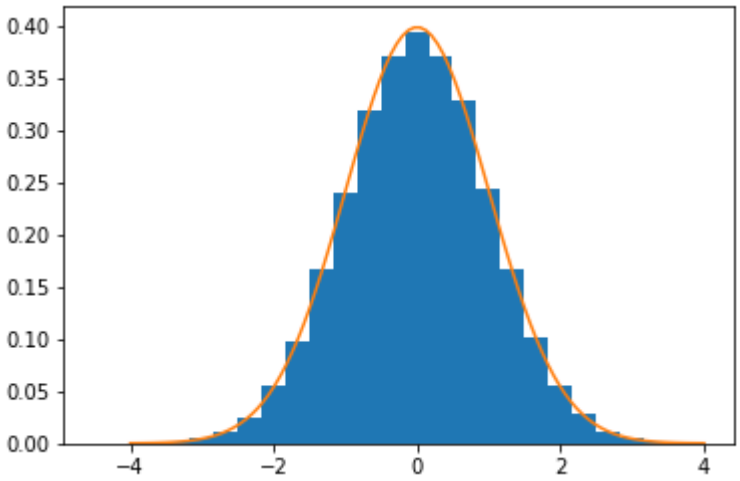
Сравнение метода принятия-отклонения и полярного метода

Были рассмотрены два метода моделирования стандартных нормальных случайных величин: метод принятия-отклонения и полярный метод.

Какой из них лучше?

Попробуем сгенерировать 100000 случайных чисел из стандартного нормального распределения тем и другим способом. Коды программ для генерации этих случайных чисел приведены в Приложении 2.

Эти программы дают возможность убедиться на практике в том, что методы, описанные выше, действительно работают, так как гистограммы, получающиеся в результате их работы, имеют форму графика плотности стандартного нормального распределения:



Кроме того, в результатах работы этих программ мы можем посмотреть время, затраченное на генерацию 100000 случайных чисел. Так, генерация методом принятия-отклонения длится 1.9 секунд, в то время как генерация с помощью полярного метода требует всего 1.4 секунды. Поэтому мы на практике убедились, что полярный метод работает быстрее, чем метод принятия-отклонения.

Глава 3. Метод Монте-Карло

§1. Идея метода Монте-Карло

Основная идея метода Монте-Карло состоит в том, чтобы аппроксимировать математическое ожидание *E(X)* средним арифметическим результатов большого числа независимых экспериментов, которые имеют то же распределение, что и *X*. В основе этого метода лежит один из самых известных результатов теории вероятностей - усиленный закон больших чисел.

Усиленный закон больших чисел (вместе с его различными вариантами) является одной из самых мощных теорем теории вероятностей и был центральным объектом исследования на протяжении всей истории теории вероятностей. Он утверждает, что среднее арифметическое последовательности независимых, одинаково распределенных случайных величин $(X_n)_{n\in\mathbb{N}}$ почти наверное сходится к математическому ожиданию $\mu = E(X_1)$, которое, конечно же, одинаково для всех X_n.

Теорема (усиленный закон больших чисел в форме Колмогорова)

Пусть $(X_n)_{n\in\mathbb{N}}$ последовательность независимых одинаково распределенных вещественных случайных величин, определенных на вероятностном пространстве (Ω, F, P). Пусть $\mu = E(X_1)$.

Тогда для P-п.в. $\omega\in\Omega$

$$\frac{1}{n}\sum_{i=1}^{n} X_i(\omega) \xrightarrow{n\to\infty} \mu$$

§2. Грубый метод Монте-Карло

Пусть X - вещественная случайная величина с конечным математическим ожиданием $E(X)$. Один из самых известных методов вычисления этого математического ожидания приведен в следующем алгоритме.

Алгоритм (грубый метод Монте-Карло). Приблизим $E(X)$ средним арифметическим $\frac{1}{N}\sum_{i=1}^{N} X_i(\omega)$ для некоторого *N* натурального. Здесь $X_i(\omega)$ - это результаты *N* независимых экспериментов, которые имеют такое же распределение вероятностей, как и X.

Метод в этом чистом виде называется **грубым методом Монте-Карло**, чтобы отличить его от всех его остальных вариантов. Для рассмотрения точности метода Монте-Карло мы должны отметить, что различные запуски метода Монте-Карло обычно приводят к различным результатам (хотя они могут быть довольно близки друг к другу!) при аппроксимации определенного выражения. Поэтому нам приходится иметь дело со стохастической ошибкой. Сначала покажем, что метод Монте-Карло правильно аппроксимирует соответствующее математическое ожидание в среднем.

Теорема (несмещенность оценки Монте-Карло). Пусть $(X_n)_{n\in\mathbb{N}}$ последовательность независимых одинаково распределенных вещественных случайных величин, определенных на вероятностном пространстве (Ω, F, P).

Тогда оценка Монте-Карло

$$\bar{X}_N := \frac{1}{N}\sum_{i=1}^{N} X_i, N \in \mathbb{N}$$

это несмещенная оценка для $\mu = E(X)$, то есть $E(\bar{X}_N) = \mu$

Доказательство.

В самом деле, в силу независимости и одинаковой распределенности случайных величин X_n имеем

$$E(\bar{X}_N) = E(\frac{1}{N}\sum_{i=1}^{N} X_i) = \frac{1}{N}\sum_{i=1}^{N} E(X_i) = \frac{1}{N} \cdot N \cdot E(X_1) = \mu$$

чтд

Хотя это уже гарантирует правильность оценки по методу Монте-Карло в среднем, это не помогает нам получить представление об абсолютном значении ошибки. Поэтому мы смотрим на стандартное отклонение разности между $\bar{X}_N$ и μ .Так как

$$Var(\bar{X}_N - \mu) = Var(\bar{X}_N) = \frac{1}{N^2}\sum_{i=1}^{N} Var(X_i) = \frac{\sigma^2}{N},$$

стандартное отклонение ошибки имеет порядок $O(1/\sqrt{N})$. Поскольку стандартное отклонение является мерой (средней) точности грубого метода Монте-Карло, этот расчет имеет следующее важное следствие:

Повышение точности оценки по методу Монте-Карло

Увеличение (средней) точности грубой оценки Монте-Карло на одну цифру (т. е. уменьшение ее стандартного отклонения в 10 раз) требует увеличения числа запусков метода Монте-Карло в 100 раз.

Можно обосновать использование стандартного отклонения в качестве меры точности оценки по методу Монте-Карло с помощью центральной предельной теоремы.

Центральная предельная теорема

Пусть $(X_n)_{n\in\mathbb{N}}$ последовательность независимых одинаково распределенных вещественных случайных величин, определенных на вероятностном пространстве (Ω, F, P). Предполагаем также, что они имеют конечную дисперсию $\sigma^2 = Var(X)$. Тогда нормированная и центрированная сумма этих случайных величин сходится по распределению к стандартному нормальному распределению, т. е.

$$\frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma} \xrightarrow{D} N(0,1) \text{ при } n \to \infty$$

Из центральной предельной теоремы можно сделать вывод, что для больших значений *n* оценка Монте-Карло приблизительно $N(\mu, \sigma^2/n)$-распределена. Поскольку стандартное отклонение σ однозначно характеризует разброс значений нормального распределения вокруг его среднего значения μ, использование стандартного отклонения в качестве меры точности оценки методом Монте-Карло оправдано.

§3. Примеры применения метода Монте-Карло

Пример 1. Оценка вероятности события

Оценка вероятности события является важным применением метода Монте-Карло. Пусть *A* будет определенным событием. Мы хотим оценить вероятность его возникновения *P(A)*. Используем соотношение между математическим ожиданием индикаторной функции 1_A,

$$1_A(\omega) = \begin{cases} 1, & \omega \in \mathrm{A} \\ 0, & \omega \notin \mathrm{A} \end{cases}$$

и вероятностью события *A*,

$$E(1_A) = P(A).$$

Тогда оценка Монте-Карло для *P(A)* просто является относительной частотой появления *A* в *N* независимых экспериментах.

Формально, пусть A_i обозначает появление события *A* в эксперименте *i*. Тогда мы определяем оценку Монте-Карло для *P(A)* как

$$\hat{p}_N(A) = \frac{1}{N}\sum_{i=1}^{N} 1_{A_i}$$

Поскольку $Var(1_A) = P(A)(1-P(A))$, то положим $\hat{\sigma}_N^2 = \hat{p}_N(A)(1-\hat{p}_N(A))$ и получим доверительный интервал уровня $1-\alpha$ для $P(A)$

$$\left[\hat{p}_N(A) - \frac{\xi_{1-\alpha/2}}{\sqrt{N}}\hat{\sigma}_N, \hat{p}_N(A) + \frac{\xi_{1-\alpha/2}}{\sqrt{N}}\hat{\sigma}_N \right],$$

где $\xi_{1-\alpha/2}$ - $(1-\alpha/2)$-квантиль стандартного нормального распределения.

Пример 2. Интегрирование с помощью Монте-Карло

Простым, но очень эффективным применением метода Монте-Карло является нахождение значения определенных интегралов вида

$$\int_{[0,1]^d} g(x)dx,$$

где *g* вещественная ограниченная функция. Вводя функцию плотности $f(x)$ *d*-мерного равномерного распределения на $[0,1]^d$ следующим образом

$$f(x) = 1_{[0,1]^d}(x), x \in R^d,$$

мы можем переписать интеграл выше как ожидаемое значение случайной величины $g(X)$, где *X* это случайная величина, равномерно распределенная на $[0,1]^d$, то есть

$$I = \int_{[0,1]^d} g(x)dx = \int g(x)f(x)dx = E(g(X))$$

Это позволяет нам вычислить оценку по методу Монте-Карло $\hat{I}$ для интеграла выше путём генерации *N* случайных величин $X_1, \ldots, X_N$, которые независимы и одинаково распределены на $[0,1]^d$. То есть оценка Монте-Карло следующая:

$$\hat{I}_N(\omega) = \frac{1}{N}\sum_{i=1}^{N} g(X_i(\omega))$$

Глава 4. Метод Монте-Карло в теории риска

В этой главе я буду ссылаться на статью [3]. В этой статье проведён анализ моделей индивидуального и коллективного риска, хорошо рассказана теоретическая часть проблемы, однако же нет практических примеров (для модели индивидуального риска). Я подробно рассмотрю некоторые задачи и приведу соответствующие алгоритмы решения на языке Python.

§1. Анализ модели индивидуального риска

Ссылаясь на статью [3], покажем, как метод Монте-Карло может быть применён для анализа простейшей модели страхового портфеля – модели индивидуального риска. Эта модель базируется на следующих упрощающих предположениях:

- анализируется фиксированный относительно короткий промежуток времени (так что можно пренебречь инфляцией и не учитывать доход от инвестирования активов) – обычно это один год;
- число договоров страхования N фиксировано и неслучайно;
- премии полностью вносятся в начале анализируемого периода; никаких поступлений в течение этого периода нет;
- мы наблюдаем каждый отдельный договор страхования и знаем статистические свойства связанных с ним индивидуальных потерь X.

Поскольку не все договоры приводят к страховому случаю, большинство из случайных величин $X_1,\ldots,X_N$, где X_i – потери по i-му договору, равны нулю. Обычно предполагается, что случайные величины $X_1,\ldots,X_N$ – независимы (в частности, исключаются катастрофы, когда одновременно по нескольким договорам наступают страховые случаи).

Основной характеристикой этой модели является *вероятность разорения*, которая определяется как вероятность R того, что суммарные потери по портфелю $S = X_1 + \ldots + X_N$ превысят активы компании u: $R = P(X_1 + \ldots + X_N > u)$. Если нам заданы число договоров страхования N и распределения случайных величин $X_1,\ldots,X_N$, описывающих размеры индивидуальных выплат по договорам, то мы можем смоделировать эти величины и тем самым смоделировать величину суммарных выплат страховщика за анализируемый период времени $S = X_1 + \ldots + X_N$.

При заданной величине резервного фонда u с каждым отдельным циклом моделирования совокупности выплат связана случайная величина ρ, равная 1 или 0 в соответствии с тем, $S > u$ или $S \le u$ (т.е. в соответствии с тем, «разорилась» компания или нет). Если $R = \mathrm{P}(S > u)$ – вероятность разорения страховщика за рассматриваемый промежуток времени, то

$$P(\rho = 1) = R, \qquad P(\rho = 0) = 1 - R$$

Иными словами, нам нужно оценить вероятность события $A = \{S > u\}$. Индикаторная функция этого события – это ρ, а вероятность этого события является вероятностью разорения $P(A) = R$.

Произведём большое число K независимых циклов моделирования. Они приведут к определённым значениям $\rho_1,\ldots,\rho_K$ индикатора события «компания разорилась». Тогда, как это было показано ранее (см. Глава 3, §3, Пример 1), грубая оценка Монте-Карло для вероятности разорения такая:

$$\bar{R} = \frac{1}{K}\sum_{i=1}^{K}\rho_i$$

Для каждой из случайных величин ρ_i мы имеем:

$$E\rho_i = P(\rho_i = 1) = R,$$

$$Var\,\rho_i = P(\rho_i = 1)\cdot P(\rho_i = 0) = R(1-R)$$

Следовательно, с вероятностью 95% неизвестное значение R лежит в интервале

$$\left[\bar{R} - 1.96\cdot\sqrt{\frac{R(1-R)}{K}}, \bar{R} + 1.96\cdot\sqrt{\frac{R(1-R)}{K}}\right],$$

поскольку 97,5%-квантиль стандартного нормального распределения составляет около 1,96.

Границы этого интервала выражены через неизвестную вероятность R, но на практике здесь можно заменить величину R её оценкой $\bar{R}$.

§2. Применение метода Монте-Карло для расчета вероятности разорения в модели индивидуального риска

Рассмотрим следующую задачу.

Задача. Предположим, что портфель состоит из 4-х одинаковых договоров страхования, учитывающих смерть от несчастного случая: если смерть застрахованного наступила от несчастного случая, то его наследникам выплачивается 500000 руб.; в случае смерти от «естественных» причин страховая выплата равна 250000 руб. Для каждого из застрахованных вероятность смерти от несчастного случая равна 0.1, вероятность смерти от естественных причин равна 0.1 и, следовательно, вероятность дожития равна 0.8. Определить вероятность разорения R в зависимости от величины капитала компании.

Решение.

Примем сумму 250000 руб. в качестве единицы измерения денежных сумм. Тогда каждая из случайных величин $X_1,\ldots,X_4$ имеет распределение, задаваемое таблицей

n	0	1	2
$P(n)$	0.8	0.1	0.1

Необходимо вычислить вероятность $P(X_1 + X_2 + X_3 + X_4 = n)$. Для этого воспользуемся формулой свёртки для дискретных величин

$$P(X_1 + X_2 = n) = \sum_{k=0}^{n} p_1(k)\,p_2(n-k),$$

где $p_1(k) = P(X_1 = k), p_2(k) = P(X_2 = k)$

Для подсчёта распределения суммы $X_1 + X_2$ образуем матрицу, элементами которой являются $p_1(i)\,p_2(j)$ и сложим её элементы вдоль побочной диагонали. Получим

n	0	1	2	3	4
$q(n) = P(X_1 + X_2 = n)$	0.64	0.16	0.17	0.02	0.01

Далее, для подсчета вероятности $r(n) = P(X_1 + X_2 + X_3 = n) = P((X_1 + X_2) + X_3 = n)$ образуем матрицу, элементами которой являются $p_3(i)q(j)$:

0.512	0.128	0.136	0.016	0.008
0.064	0.016	0.017	0.002	0.001
0.064	0.016	0.017	0.002	0.001

Поэтому для распределения $X_1 + X_2 + X_3$ имеем таблицу:

n	0	1	2	3	4	5	6
r(n)	0.512	0.192	0.216	0.049	0.027	0.003	0.001

Аналогично вычисляем таблицу для четырёх случайных величин, то есть для распределения суммарного иска:

0	1	2	3	4	5	6	7	8
0.4096	0.2048	0.2432	0.0800	0.0481	0.0100	0.0038	0.0004	0.0001

Соответственно для функции распределения $P(X_1 + X_2 + X_3 + X_4 \le n)$ получим таблицу:

0	1	2	3	4	5	6	7	8
0.4096	0.6144	0.8576	0.9376	0.9857	0.9957	0.9995	0.9999	1.0000

Таким образом, зависимость вероятности разорения $R = P(X_1 + X_2 + X_3 + X_4 > u)$ от величины имеющегося капитала задаётся таблицей

Капитал, u	0	1	2	3	4	5	6	7	8
Вероятность разорения, R	0.5905	0.3856	0.1424	0.0624	0.0143	0.0043	0.0005	0.0001	0

Теперь попробуем смоделировать вероятность разорения с помощью метода Монте-Карло. Алгоритмы моделирования я буду записывать на языке программирования Python.

В первую очередь для метода Монте-Карло нужен генератор случайных чисел. Я буду использовать простейший линейный конгруэнтный генератор. Вот функция, которая его задаёт:

```
import numpy as np
def rng(m=2147483647, a=16807, c=0):
    rng.current = (a * rng.current + c) % m
    return rng.current / m
rng.current = 1
```

Далее необходимо смоделировать индивидуальный иск X.

В программе зададим параметры:

```
p0=0.8
p1=0.1
p2=0.1
N=4
```

Здесь N – это количество страховых договоров.

Функция, которая моделирует случайную величину, имеющую нужное распределение.

```
def X(p0,p1):
    r=rng()
    if r<p0:
            return 0
    elif r>=p0 and r<p0+p1:
            return 1
    else:
            return 2
```

Далее можно смоделировать суммарный иск, то есть $S = X_1 + X_2 + X_3 + X_4$. Функция, которая это делает:

```
def S(p0,p1,N):
    s=0
    for i in range(N):
            s+=X(p0,p1)
    return s
```

И, наконец, запустим алгоритм Монте-Карло с помощью следующей функции:

```
def prob_razor(u,K,p0,p1):
    R=0
    for i in range(K):
            s=S(p0,p1,N)
            if s>u:
                    R+=1
    return R/K
```

Эта функция принимает на вход u – величину капитала страховой компании, K – количество циклов метода Монте-Карло, p0 и p1, и возвращает оценку Монте-Карло для вероятности разорения компании R.

Напишем также функцию, которая печатает доверительный интервал уровня percent для R

```
import scipy.stats
def conf_int(percent,R,K):
    quantile=scipy.stats.norm.ppf(1-(1-percent/100)/2)
    print('Доверительный интервал уровня ‘+str(percent)+’%:\n[',R-quantile*np.sqrt(R*(1-
R)/K),',',R+quantile*np.sqrt(R*(1-R)/K),'], длина интервала:',2*quantile*np.sqrt(R*(1-
R)/K) )
```

И последняя функция, которая выведет на экран всю интересующую нас информацию:

```
def print_prob(K,u,p0,p1):
   R=prob_razor(u,K,p0,p1)
   print('Вероятность разорения при начальном капитале', u,':',R)
   conf_int(95,R,K)
```

Запустим поочерёдно эту функцию для величины начального капитала u=0, 1, 2, 3, 4, 5, 6, 7 и 8 при количестве циклов метода Монте-Карло K=1000000

Вот что получим в результате:

```
In [37]: print_prob(1000000,0,p0,p1)
```

```
Вероятность разорения при начальном капитале 0 : 0.590522
Доверительный интервал уровня 95%:
[ 0.5895582122129228 , 0.5914857877870772 ], длина интервала: 0.0019275755741542805
```

```
In [38]: print_prob(1000000,1,p0,p1)
```

```
Вероятность разорения при начальном капитале 1 : 0.385693
Доверительный интервал уровня 95%:
[ 0.3847389707250118 , 0.38664702927498823 ], длина интервала: 0.001908058549976415
```

```
In [39]: print_prob(1000000,2,p0,p1)
```

```
Вероятность разорения при начальном капитале 2 : 0.14248
Доверительный интервал уровня 95%:
[ 0.1417949107179546 , 0.14316508928204538 ], длина интервала: 0.0013701785640907879
```

```
In [40]: print_prob(1000000,3,p0,p1)
```

```
Вероятность разорения при начальном капитале 3 : 0.062501
Доверительный интервал уровня 95%:
[ 0.06202656471563798 , 0.06297543528436203 ], длина интервала: 0.0009488705687240447
```

```
In [42]: print_prob(1000000,4,p0,p1)
```

```
Вероятность разорения при начальном капитале 4 : 0.014288
Доверительный интервал уровня 95%:
[ 0.014055400469580589 , 0.014520599530419412 ], длина интервала: 0.0004651990608388246
```

```
In [43]: print_prob(1000000,5,p0,p1)
```

```
Вероятность разорения при начальном капитале 5 : 0.004266
Доверительный интервал уровня 95%:
[ 0.0041382590372773115 , 0.004393740962722688 ], длина интервала: 0.0002554819254453769
```

```
In [44]: print_prob(1000000,6,p0,p1)
```

```
Вероятность разорения при начальном капитале 6 : 0.000473
Доверительный интервал уровня 95%:
[ 0.00043038368180558264 , 0.0005156163181944173 ], длина интервала: 8.523263638883468e-05
```

```
In [45]: print_prob(1000000,7,p0,p1)
```

```
Вероятность разорения при начальном капитале 7 : 0.000105
Доверительный интервал уровня 95%:
[ 8.491739996690207e-05 , 0.0001250826000330979 4 ], длина интервала: 4.016520006619586e-05
```

```
In [46]: print_prob(1000000,8,p0,p1)
```

```
Вероятность разорения при начальном капитале 8 : 0.0
Доверительный интервал уровня 95%:
[ 0.0 , 0.0 ], длина интервала: 0.0
```

Капитал, u	0	1	2	3	4	5	6	7	8
Вероятность разорения, R	0.5905	0.3856	0.1424	0.0624	0.0143	0.0043	0.0005	0.0001	0
Оценка Монте-Карло, $\bar{R}$	0.590522	0.385693	0.14248	0.062501	0.014288	0.004266	0.000473	0.000105	0

Как видно, отличие оценки от истинного значения очень мало, то есть метод Монте-Карло оказался эффективным в этом случае.

§3. Анализ модели коллективного риска

Так же как и в модели индивидуального риска, в *модели коллективного риска* анализируется относительно короткий промежуток времени и предполагается, что премии полностью вносятся в начале анализируемого периода. Однако, в отличие от модели индивидуального риска, в модели коллективного риска весь портфель заключённых договоров страхования рассматривается как единое целое, без различения отдельных составляющих его договоров. Соответственно, наступающие страховые случаи не связываются с конкретными договорами, а рассматриваются как результат суммарного риска страховщика. Отсюда следует, что основной характеристикой портфеля является не число заключённых договоров *N*, а общее число страховых случаев ν за анализируемый период. Ясно, что ν является случайной величиной.

Второе важное отличие модели коллективного риска от модели индивидуального риска заключается в том, что предполагается одинаковая распределённость случайных величин $Y_1, Y_2, \ldots,$ описывающих величины потерь вследствие последовательных страховых случаев. Это предположение означает определённую равноценность страховых случаев, связанную с тем, что они рассматриваются как следствие общего риска компании, а не индивидуальных договоров с их специфическими особенностями. Кроме того, важно подчеркнуть, что случайные величины Y_i описывают только реальный ущерб и поэтому в отличие от величин X_j, фигурирующих в модели индивидуального риска, строго положительны.

В теории коллективного риска также обычно предполагается, что число страховых случаев и потери после наступления страховых случаев независимы в совокупности. Это довольно естественное предположение, т.к. частота наступления страховых случаев и их тяжесть в большинстве реальных ситуаций зависят от разных факторов.

Так же как и в модели индивидуального риска, в модели коллективного риска «разорение» определяется суммарными выплатами *S* страховой компании. Однако теперь *S* записывается в виде $S = Y_1 + \ldots + Y_\nu$, т.е. является суммой случайного числа случайных слагаемых, и поэтому в модели коллективного риска вероятность разорения страховщика определяется как

$$R = P(Y_1 + \ldots + Y_\nu > u),$$

где, как и в модели индивидуального риска, u – активы страховщика.

Если нам заданы:

1. распределение π_i числа страховых случаев ν за анализируемый промежуток времени;
2. (общее) распределение $F(x)$ случайных величин $Y_1, Y_2, \ldots,$ описывающих размер ущерба после наступления 1,2,… страхового случая, то мы можем смоделировать эти величины и тем самым смоделировать величину суммарных выплат страховщика $S = Y_1 + \ldots + Y_\nu$.

Пусть u – размер резервов страховщика, предназначенных для выплат по анализируемому портфелю договоров. Как и при имитационном моделировании модели индивидуального риска, с каждым отдельным циклом моделирования модели коллективного риска связана случайная величина ρ, равная 1 или 0 в соответствии с тем, $S > u$ или $S \le u$ (т.е. в соответствии с тем, «разорилась» компания или нет).

Как и при имитационном моделировании модели индивидуального риска, проведём большое число K циклов моделирования. Они дадут определённые значения $\rho_1, \ldots, \rho_K$ индикатора события «компания разорилась». Оценкой Монте-Карло, как и прежде, является $\bar{R} = \frac{1}{K}\sum_{i=1}^{K} \rho_i$.

§4. Применение метода Монте-Карло для оценки вероятности разорения в модели коллективного риска

В качестве примера рассмотрим следующую ситуацию. Пусть число страховых случаев в течение года имеет геометрическое распределение с параметром $p = 0.2$, то есть

$$P(\nu = n) = 0.8^n \cdot 0.2, \; n = 0,1,2,\ldots$$

а потери после наступления страхового случая имеют экспоненциальное распределение с параметром 2, $Y_i \sim Exp(2)$. Необходимо найти вероятность разорения при величине начального капитала $u = 5$.

Решение.

Найдём плотность суммарных потерь S. Для этого заметим, что поскольку величина ущерба в результате страхового случая Y_i имеет экспоненциальное распределение, сумма n таких величин $Y_1 + \ldots + Y_n$ имеет гамма-распределение с параметрами $\lambda = 2$ и $\alpha = n$, так что её плотность следующая

$$f_{Y_1+\ldots+Y_n}(x) = \frac{2^n x^{n-1}}{(n-1)!} e^{-2x}$$

Плотность суммарных потерь $S = Y_1 + \ldots + Y_\nu$ может быть найдена следующим образом:

$$f_S(x) = \sum_{n=1}^{\infty} P(\nu = n) f_{Y_1+\ldots+Y_n}(x) = \sum_{n=1}^{\infty} 0.8^n \cdot 0.1 \cdot \frac{2^n x^{n-1}}{(n-1)!} e^{-2x} = 0.32 e^{-2x} \sum_{n=1}^{\infty} \frac{(1.6x)^{n-1}}{(n-1)!} = 0.32 e^{-0.4x}$$

Теперь можно легко найти вероятность разорения:

$$R = P(S > 5) = \int_5^{\infty} f_S(x)dx = 0.8e^{-2} = 0.108268$$

Ниже приведён код программы для решения этой задачи.

Генератор случайных чисел:

```
import matplotlib.pyplot as plt
import numpy as np
def rng(m=2**31-1, a=16807, c=0):
    rng.current = (a * rng.current + c) % m
    return rng.current / m
rng.current = 1
```

Функция для моделирования случайных величин, имеющих геометрическое распределение с параметром p:

```
def geom(p):
    u=rng()
    pi=p
    s=pi
    nu=0
    while u>=s:
        nu=nu+1
        pi=pi*(1-p)
        s=s+pi
    return nu
```

Функция для моделирования случайных величин, имеющих экспоненциальное распределение:

```
def exp(l):
     return -np.log(rng())/l
```

Функция моделирования величины суммарных выплат страховщика:

```
def S(p,l):
    nu=geom(p)
    s=0
    for i in range(nu):
        s+=exp(l)
    return s
```

Функция для получения оценки для вероятности разорения по методу Монте-Карло:

```
def prob_razor(u,K,p,l):
    R=0
    for i in range(K):
        s=S(p,l)
        if s>u:
            R+=1
```

```
    return R/K
```

Функция, печатающая доверительный интервал для вероятности разорения:

```
import scipy.stats
def conf_int(percent,R,K):
    quantile=scipy.stats.norm.ppf(1-(1-percent/100)/2)
    print('Доверительный интервал уровня'+str(percent)+':\n[',R-quantile*np.sqrt(R*(1-
R)/K),',',R+quantile*np.sqrt(R*(1-R)/K),'], длина интервала:',2*quantile*np.sqrt(R*(1-
R)/K) )
```

Запускаем процедуру с необходимыми параметрами:

```
for i in range(10000,100001,10000):
    print('K=',i,'R=',prob_razor(5,i,0.2,2))
```

Результаты работы этой программы приведены в следующей таблице:

10000	0.1090
20000	0.1063
30000	0.1085
40000	0.1093
50000	0.1076
60000	0.1094
70000	0.1087
80000	0.1075
90000	0.1082
100000	0.1083

При этом запуск алгоритма на 10000000 циклов дал результат 0.10820 при доверительном 95%-ном интервале [0.10801;0.10839].

Как видно, результат имитационного моделирования практически не отличается от истинного значения вероятности разорения (абсолютная погрешность не больше, чем $1.96 \cdot \sqrt{\frac{\bar{R}(1-\bar{R})}{K}} \approx 1.9 \cdot 10^{-4}$)

Заключение и выводы

В этой работе был рассмотрен метод Монте-Карло в своей грубой форме и приведены примеры его использования в задачах по теории риска.

Метод Монте-Карло имеет свои преимущества и недостатки.

Из преимуществ можно выделить:

- процедура моделирования является простой и естественной
- при большом количестве итераций метод обеспечивает хорошую точность
- его легко применять без предварительного анализа задачи

Однако у метода Монте-Карло есть свои недостатки:

- без дополнительных способов уменьшения дисперсии оценки сходимость метода является весьма медленной
- в любом случае этот метод является оценочным, то есть позволяет получить лишь доверительный интервал для интересующей нас величины (а не точное её значение)

Поэтому метод Монте-Карло стоит применять для решения задач, в которых не требуется высокая точность ответа.

Приложение 1

Код программы написан на языке программирования Python.

```
import matplotlib.pyplot as plt
import numpy as np
def rng(m=3331, a=1, c=1):
        rng.current = (a * rng.current + c) % m
        return rng.current / m
rng.current = 1
random = [rng() for i in range(5000)]
plt.scatter(random[1:], random[:-1])
plt.show()
```

Приложение 2

Код программы написан на языке программирования Python.

Программа для генерации 100000 случайных чисел из стандартного нормального распределения методом принятия-отклонения:

```
import matplotlib.pyplot as plt
import numpy as np
def rng(m=2**31-1, a= 16807, c=0):
   rng.current = (a * rng.current + c) % m
   return rng.current / m
rng.current = 1
def exp(l):
   return -np.log(rng())/l
def f(x):
   return np.sqrt(2/np.pi)*np.exp(-x**2/2)
def g(y):
   return np.exp(-y)
def sign():
   u=rng()
   if u<0.5:
      return -1
   else:
      return 1
import time
from scipy import stats
C=np.sqrt(2*np.e/np.pi)
c1=[]
start_time = time.time()
i=0
while i<100000:
   u=rng()
   y=exp(1)
```

```
    if u<=f(y)/(C*g(y)):
        c1.append(sign()*y)
        i+=1
print("--- %s seconds ---" % (time.time() - start_time))
dist = stats.norm()
x = np.linspace(-4, 4, 100)
plt.hist(c1,normed=True,bins=25)
plt.plot(x,dist.pdf(x))
plt.show()
```

Программа для генерации 100000 случайных чисел из стандартного нормального распределения с помощью полярного метода:

```
from scipy import stats
import time
import matplotlib.pyplot as plt
import numpy as np
def rng(m=2**31-1, a= 16807, c=0):
    rng.current = (a * rng.current + c) % m
    return rng.current / m
rng.current = 1
c2=[]
start_time = time.time()
i=0
while i<100000:
    x1=2*rng()-1
    x2=2*rng()-1
    s=x1**2+x2**2
    if s<1:
        z1=x1*np.sqrt(-2*np.log(s)/s)
        z2=x2*np.sqrt(-2*np.log(s)/s)
        c2.append(z1)
        i+=1
print("--- %s seconds ---" % (time.time() - start_time))
dist = stats.norm()
x = np.linspace(-4, 4, 100)
plt.hist(c2,normed=True,bins=25)
plt.plot(x,dist.pdf(x))
plt.show()
```

Литература

[1] R.Korn, E.Korn, G.Kroisandt. *Monte Carlo Methods and Models in Finance and Insurance*. Chapman & Hall/CRC Press Financial Mathematics Series, 2010. ISBN: 978-1-4200-7618-9

[2] Г.И.Фалин. *Математический анализ рисков в страховании*. Российский юридический издательский дом. Москва, 1994. ISBN 5-88635-003-0

[3] Г.И.Фалин. Анализ рисков с помощью метода Монте-Карло. *Управление Риском*, 2017, №1.

[4] D. E. Knuth. The Art of Computer Programming, Volume 2 (Seminumerical Algorithms). Addison-Wesley, Reading, Massachusetts, USA, 3rd edition, 1998.

[5] D. Lehmer. Mathematical methods in large-scale computing units. In *Proceedings of the 2nd Symposium on Large-Scale Digital Calculating Machinery*,141–146, Harvard University Press, Cambridge, Massachusetts, USA, 1949.