6. Лабораторная работа №3

6.1. Вводные замечания

В третьей лабораторной работе рассматривается применение метода статистического, называемого также *методом Монте-Карло*, к симуляции дискретных случайных событий. Данный метод имеет ряд особенностей:

- сравнительно прост в реализации на компьютере;
- может быть эффективно распараллелен.

Для использования метода Монте-Карло необходимо научиться пользоваться генератором *псевдослучайных* чисел, поэтому начнем с рассмотрения примеров его использования в Fortran.

6.2. Генератор псевдослучайных чисел в Fortran

Числа, генерируемые с помощью компьютера, называются *псевдослучайными*, так как алгоритм их получения является детерминированным. В современной версии компилятора gfortran [1] используется алгоритм генерации псевдослучайных чисел под названием xoshiro [4, 2, 3]. Он дает очень «качественную» последовательность с большим периодом. Под «качеством» подразумевается выполнения статистических свойств, присущих равномерному распределению. В результате работы генератора мы получаем *последовательность псевдослучайных равномерно распределенных чисел*.

Кроме термина генератор, в отечественной математической литературе также часто можно встретить термин *счетчик* или *датчик* случайных чисел (см., например, [5, с. 58]).

Далее везде будем для краткости называть последовательность псевдослучайных равномерно распределенных чисел просто случайными числами.

Не будем углубляться в алгоритмы генерации, а будем использовать встроенную подпрограмму:

```
real :: x
call random_number(x)
```

Данная подпрограмма записывает переданный ей аргумент `x` случайное число из интервала $[0, 1)$. Аргументом `x` может быть как скаляр, так и массив. В последнем случае весь массив будет заполнен случайными числами.

Для работы генератора ему требуется передать начальное значение (seed). Прежде чем запускать генератор его необходимо инициализировать. При вызове подпрограмма `random_number` делает это неявно и автоматически. Но в зависимости от операционной системы, компилятора и библиотек, от запуска к запуску для инициализации может использоваться одно и то же число и последовательность случайных чисел также будет одинакова. Возможна противоположная ситуация, когда при каждом запуске seed меняется, а за ним меняется и последовательность чисел.

Для явной инициализации генератора используется специальная подпрограмма `random_seed`. Кроме инициализации генератора, подпрограмма `random_seed` позволяет узнать, сколько начальных значений требуется генератору (это число может зависть от версии и сборки компилятора) и какие значения используются в настоящий момент. Ее использование иллюстрирует следующий пример кода.

```
program random_example
  implicit none
  integer, parameter :: length = 10
  real, dimension(1:length) :: rand_array
  integer, dimension(:), allocatable :: seed
  integer :: seed_size
  integer :: i

  ! Узнаем размер массива, который надо передать для
  ! инициализации генератора случайных чисел
  call random_seed(size=seed_size)

  print "(G0, I5)", "Размер массива для seed: ", seed_size
```

```
14
15    ! Инициализируем этот массив
16    allocate(seed(1:seed_size))
17    ! Проверяем какие начальные значения установлены в генераторе в
18    ! настоящий момент
19    call random_seed(get=seed)
20    print "(G0,/, 5(I15,:,', '))",  "Текущий seed:", seed
21
22    ! Задаем свой seed массив
23    seed = [(i, i = 1,seed_size,1)]
24
25    ! И передаем его генератору
26    call random_seed(put=seed)
27    ! и тут же получаем обратно для проверки
28    call random_seed(get=seed)
29    print "(G0,/, 11(I4,:,', '))",  "Новый seed", seed
30
31    ! Заполняем массив случайными числами
32    call random_number(rand_array)
33    print "(G0,/, 5(G0,:,', '))",  "rand_array(1:10) = ", rand_array
34
35    deallocate(seed)
36
37  end program random_example
```

Для операционных систем семейства Unix есть возможность использовать псевдоустройства `/dev/urandom` и `/dev/random`. Как это сделать показано в примере ниже.

```
1   module devrandom
2     implicit none
3     contains
4
5     ! Функция считывает целое число одинарной точности
6     function intrand()
7       integer(kind=4) :: intrand
8
9       open(unit=10, file="/dev/urandom", access='stream', form='unformatted', action="read")
10      read(unit=10) intrand
11      close(unit=10)
12    end function intrand
13
14    ! Функция считывает целое число двойной точности
15    function longrand()
16      integer(kind=8) :: longrand
17
18      open(unit=10, file="/dev/urandom", access='stream', form='unformatted', action="read")
19      read(unit=10) longrand
20      close(unit=10)
21    end function longrand
22
23    ! Функция считывает и возвращает массив целых чисел двойной точности
24    function array_intrand(N)
25      integer, intent(in) :: N
26      integer(kind=8), dimension(1:N) :: array_intrand
27
```

```
    open(unit=10, file="/dev/urandom", access='stream', form='unformatted', action="read")
    read(10) array_intrand
    close(unit=10)
  end function array_intrand

  ! Функция считывает и возвращает массив целых чисел двойной точности
  function array_longrand(N)
    integer, intent(in) :: N
    integer(kind=8), dimension(1:N) :: array_longrand

    open(unit=10, file="/dev/urandom", access='stream', form='unformatted', action="read")
    read(10) array_longrand
    close(unit=10)
  end function array_longrand
end module devrandom

program main
  use devrandom
  implicit none

  ! Считываем одно целое число одинарной точности
  ! и второе число двойной точности
  print *, "seed4 = ", intrand()
  print *, "seed8 = ", longrand()

  ! Считываем целый массив
  print "(a10,/,1(i20, 1x))",  "random: ", array_longrand(10)

  close(unit=10)
end program main
```

6.3. Задачи теории вероятностей

Рассмотрим ряд задач теории вероятностей, где используется дискретное пространство случайных событий и частоту этих событий можно приближенно вычислить методом Монте–Карло.

6.3.1. Классическая задача о разорении игрока

Рассмотрим *задачу о разорении* [8, с. 336]. Два игрока играют в азартную игру. Игрок №1 выигрывает рубль при каждом успехе и проигрывает рубль при каждой неудаче. Вероятность успеха равна p, а вероятность неудачи $1 - p$ (распределение Бернулли). Будем предполагать, что игрок №1 и его противник (игрок №2) имеют в общей сложности N рублей, причем в начале игры игрок №1 имеет n, а игрок №2 $N - n$ рублей. Игра продолжается до тех пор, пока капитал первого игрока либо опустится до нуля, либо возрастет до N рублей то есть, пока один из игроков не разорится. Нас интересует вероятность разорения или выигрыша игрока, а также среднее число раундов, которое нужно провести, чтобы определить разорился ли игрок или выиграл.

Данная задача равносильна задаче о *случайном блуждании* [9, с. 112] точки по отрезку $[0, N]$ с целочисленными координатами. В начальный момент времени точка находится в координате n. Через равные моменты времени точка может перепрыгнуть или в точку $n-1$ или $n+1$. Процесс продолжается, пока точка не окажется в координате 0 или N. Часто говорят о поглощающих экранах, находящихся в этих точках, которые поглощают точку и процесс блуждания заканчивается. Эта примитивная модель броуновского процесса.

Данная задача очень подробно изучена аналитически, поэтому удобна в качестве учебной, так как результаты компьютерных вычислений могут быть проверены по аналитическим формулам. Дополнительно приводится таблица 1 из [8, с. 343] с некоторыми расчетами для разных p, q, n и N.

Еще раз перечислим все введенные обозначения в терминах задачи о разорении:

p	q	n	N	**Вероятность**		**Мат. ожидание**	
				разорения	выигрыша	выигрыша	кол. раундов
0,5	0,5	9	10	0,1	0,9	0	9
0,5	0,5	90	100	0,1	0,9	0	900
0,5	0,5	900	1000	0,1	0,9	0	90000
0,5	0,5	950	1000	0,05	0,95	0	47500
0,5	0,5	8000	10000	0,2	0,8	0	16000000
0,45	0,55	9	10	0,210	0,790	-1,1	11
0,45	0,55	90	100	0,866	0,134	-76,6	765,6
0,45	0,55	99	100	0,182	0,818	-17,2	171,8
0,4	0,6	90	100	0,983	0,017	-88,3	441,3
0,4	0,6	99	100	0,333	0,667	-32,3	161,7

Таблица 1: Вероятностные характеристики задачи о разорение для разных шансов на успех p и неудачу q и для разных начальных и конечных капиталах n и N. В столбце «кол. раундов» приведено среднее количество раундов игры. Таблица взята из [8, с. 343]

- n — начальный капитал игрока №1;
- N — общий капитал, то есть у игрока №2 будет $N - n$ рублей;
- p — вероятность успеха на каждом ходу для игрока №1;
- $q = 1 - p$ — вероятность неудачи для игрока №1 (она же вероятность успеха для игрока №2).

Вероятность проигрыша $q(n)$ при начале игры с суммой n дается формулой:

$$q(n) = \begin{cases} \dfrac{\left(\frac{q}{p}\right)^n - \left(\frac{q}{p}\right)^N}{1 - \left(\frac{q}{p}\right)^N}, & p \neq q, \\ 1 - \dfrac{n}{N}, & p = q. \end{cases}$$

Математическое ожидание результата игры то есть теоретическая средняя сумма, которую выигрывает игрок за бесконечное количество игр. Более формально, это математическое ожидание следующей случайной величины X:

$$X = \begin{cases} N - n, & \text{выигрыш}, \\ -n, & \text{проигрыш}. \end{cases} \quad \mathbb{E}[X] = \begin{cases} (1 - q(n))N - n, & p \neq q, \\ 0, & p = q. \end{cases}$$

Интересно, что дисперсия для случая $p = q$ равна $\mathbb{D}[X] = n(N - n) = Nn - n^2$ и ее максимум достигается при $n = N/2$ то есть в том случае, когда ирок №1 на старте обладает половиной всего капитала и, фактически, играют равные игроки.

Можно получить аналитические формулы и для средней продолжительности игры при стартовой сумме n у игрока №1:

$$\mu(n) = \begin{cases} \dfrac{n}{q-p} - \dfrac{N}{q-p}\dfrac{1-(q/p)^n}{1-(q/p)^N}, & p \neq q, \\ n(N-n), & p = q = 1/2. \end{cases}$$

6.3.2. Игра с жетонами [6]

Три игрока (с номерами 1,2 и 3), имеющие изначально x, y и z жетонов, соответственно, играют в следующую игру. В каждом раунде каждый игрок ставит на кон один жетон. Затем бросают кубик, на котором цифры 4, 5 и 6 заменены на 1, 2 и 3. Иными словами, при броске кубика с равной вероятностью выпадают цифры 1, 2 и 3. При выпадении числа i игрок с номером i забирает с кона все три жетона. Игра заканчивается, когда кто-нибудь из игроков проигрывает все своим жетоны. Задача определить среднюю длительность игры.

Точная формула для средней продолжительности игры для начального расклада x, y и z известна и имеет вид:

$$\mu(x, y, z) = \frac{xyz}{x + y + z - 2}$$

Это более сложный вариант задачи о разорении игрока, так как тут участников трое.

6.3.3. Парадокс Монти Холла

Данный парадокс получил свое название в честь Монти Холла — ведущего популярной в США телевизионной игры Let's Make a Deal. Первый выпуск вышел еще в 1963 году и новые выпуски продолжают выходить в настоящее время, хотя ведущий, разумеется уже сменился.

В конце каждого выпуска участник, добравшийся до финала, становился вместе с Монти Холлом перед тремя дверьми: №1, №2 и №3. Монти Холл объяснял финалисту, что за одной из этих дверей скрывается очень ценный — например новый автомобиль, а за двумя другими — козел. Финалист должен был выбрать одну из трех дверей. После того, как участник выбирал дверь, Монти Холл открывал одну из двух оставшихся дверей, за которой всегда находился козел. Затем ведущий спрашивал финалиста, не желает ли он изменить свое решение, то есть отказаться от ранее выбранной им закрытой двери в пользу другой закрытой двери.

Парадокс заключается в том, что при смене выбора вероятность выиграть автомобиль равна $2/3$, а при отказе сменить выбор — $1/3$. Такой результат противоречит интуиции, но математически получается простым перебором вариантов. Есть также подход к решению, основанный на формуле Байеса. Подробнее см. в Википедии или в книге [7, с. 127].

6.3.4. Парадокс дней рождения

Задача звучит следующим образом: найти вероятность совпадения дней рождения хотя бы у пары людей в группе из n человек. Парадокс заключается в том, что интуитивно кажется, что данная вероятность крайне мала, однако расчет показывает, что уже при $n = 23$ вероятность совпадения превышает $0,5$.

В книге [9, с. 34] данная задача рассмотренная как частный случай задачи о совпадениях. Рассматривается урна, которая содержит M пронумерованных шаров (начиная с 1). Из урны последовательно вынимаются с шары с последующим возвращением обратно. Какова вероятность, что за n испытаний ни один шар не будет вынут повторно? В такой формулировке эта задача является обратной к парадоксу дней рождения.

Аналитические решения имеет следующий вид:

$$P(n) = 1 - \frac{365!}{365^n (365 - n)!}$$

В связи с 365! формула сложно применима в общем случае. Есть альтернативные подходы, смотрите, например, в Википедии или в [9, с. 35].

6.4. Задания лабораторной работы

При правильной реализации параллельных вычислений в данных задачах, вы получите очень заметный выигрыш во времени выполнения. Постройте графики времени выполнения, эффективности и ускорения в зависимости от количества потоков.

6.4.1. Задание №1

- Используйте подпрограмму `random_number` и напишите функцию, которая возвращает случайное целое число i, такое что $1 \leqslant i \leqslant N$.
- Проверьте на примере $n = 6$, что вероятность выпадения чисел $1, 2, 3, 4, 5, 6$ одинакова, то есть распределение равномерное. Это соответствует задаче по подбрасыванию шестигранного игрального кубика.
- Проверку осуществите проведя минимум $N = 10^6$ испытаний методом Монте-Карло.
- Программу ускорьте с помощью параллельных потоков OpenMP.

6.4.2. Задание №2

Смоделируйте задачу про парадокс Монти Холла методом Монте-Карло для двух стратегий игрока:

1. игрок никогда не меняет свое первое решение и открывает ту дверь, которую выбрал первой;
2. игрок всегда меняет свое решение и открывает вторую дверь, предложенную Монти Холлом.

Программа должна работать в параллельном режиме. Вычислите частоту побед в для обеих стратегий. Теоретическое решение дает вероятность 2/3 выиграть при смене решения и при 1/3 если решение не менять.

6.4.3. Задание №3

Смоделируйте задачу про парадокс дней рожденья методом Монте-Карло. Вычислите частоту совпадений дней рождения для 10^6 испытаний. Постройте график изменения этой частоты в зависимости от количества людей в группы. Начните с двух человек ($n = 2$) и до 100 человек ($n = 100$). Программа должна работать в параллельном режиме.

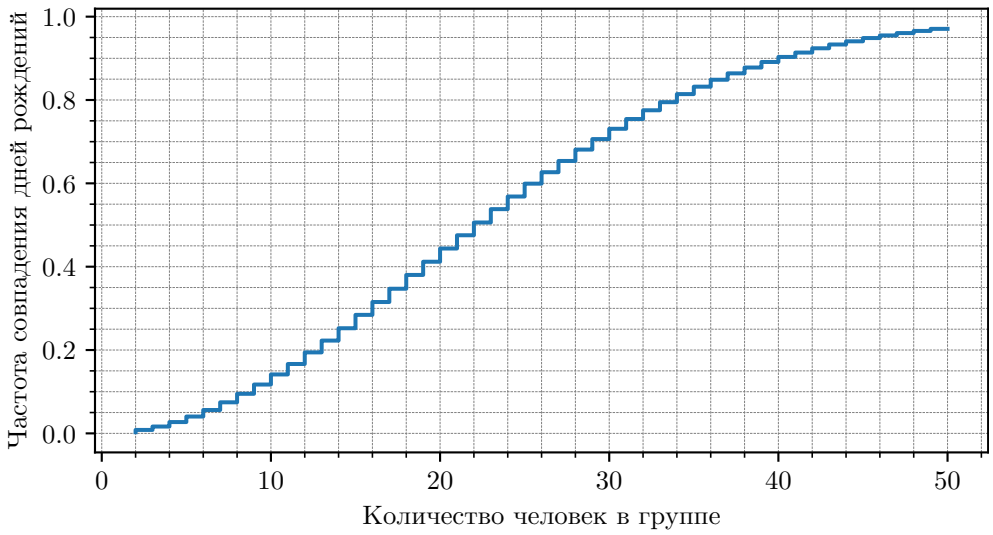

Рис. 4: Иллюстрация к задаче о парадоксе дней рождения

6.4.4. Задание №4

Смоделируйте задачу о разорении игрока методом Монте-Карло. Вычислите частоту побед и проигрышей. Вычислите также среднее число раундов в каждой игре.

6.4.5. Задание №5

Смоделируйте игру в жетоны тремя игроками методом Монте-Карло. Вычислите частоту побед и проигрышей. Вычислите среднюю продолжительность раундов в каждой игре и сравните ее с теоретическим значением.

Список литературы

1. Gfortran — the GNU Fortran compiler, part of GCC / Free Software Foundation. — 2020. — URL: https://gcc.gnu.org/wiki/GFortran.
2. *Steele G.*, *Vigna S.* Computationally easy, spectrally good multipliers for congruential pseudorandom number generators. — 2020.

3. *Vigna S.* It is high time we let go of the Mersenne Twister. — 2019.
4. *Vigna S.* On the probability of overlap of random subsequences of pseudorandom number generators // Information Processing Letters. — 2020. — Т. 158. — С. 105939. — ISSN 0020-0190. — DOI: 10.1016/j.ipl.2020.105939. — URL: http://www.sciencedirect.com/science/article/pii/S0020019020300260.
5. *Лагутин М. Б.* Наглядная математическая статистика. — 6-е изд. — Москва : БИНОМ. Лаборатория знаний, 2018. — 472 с. — ISBN 9785001010654.
6. *Паньгина Н. Н.*, *Паньгин А. А.* Статистическое моделирование: метод Монте-Карло // Компьютерные инструменты в образовании. — 2002. — № 5.
7. *Уилан Ч.* Голая статистика. Самая интересная книга о самой скучной науке / под ред. А. Степанов ; пер. И. Веригин. — 2-е изд. — Манн, Иванов и Фербер, 2017. — 352 с. — ISBN 9785001008231.
8. *Феллер У.* Введение в теорию вероятностей и ее приложения : в 2 т. Т. 1. — 3-е изд. — Москва : Мир, 1984. — 498 с.
9. *Ширяев А. Н.* Вероятность : в 2 т. Т. 1. — 4-е изд. — Москва : МЦНМО, 2007. — 552 с. — ISBN 9785940571056.