|  |  |
| --- | --- |
| Изображение выглядит как текст, керамические изделия, фарфор  Автоматически созданное описание | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Факультет «Фундаментальные науки»

Кафедра «Техническая физика»

**Лабораторная работа №6**

по курсу «Вычислительная физика»

Выполнили: Коберник Т. Н., Плетенёв Б. А.

Группа: ФН4-71Б

Преподаватели: Хасаншин Р.Х., Ивлиев П.А.

Москва, 2022 г.

Содержание

[1. Теоретическая часть 3](#_Toc117473200)

[1.1 Задачи, решаемые методом ММК 3](#_Toc117473201)

[1.2 Основные особенности ММК 3](#_Toc117473202)

[1.3 Общая схема ММК 4](#_Toc117473203)

[1.4 Вычисление определённых интегралов и снижение дисперсии 6](#_Toc117473204)

[2. Постановка задачи 10](#_Toc117473205)

[3. Программа 11](#_Toc117473208)

[4. Результаты 14](#_Toc117473209)

[5. Выводы 16](#_Toc117473215)

1. **Теоретическая часть**

## **. Задачи, решаемые методом ММК**

Методы Монте–Карло (ММК) – это численные методы [решения](http://matica.org.ua/sdelat-zakaz) математических [задач](http://matica.org.ua/sdelat-zakaz) с помощью моделирования случайных величин. ММК позволяют успешно решать математические задачи, обусловленные вероятностными процессами. Более того, при решении [задач](http://matica.org.ua/sdelat-zakaz), не связанных с какими-либо вероятностями, можно искусственно придумать вероятностную модель (и даже не одну), позволяющую решать эти задачи.

Чаще всего ММК используется для вычисления площадей, интегралов, приблизительных значений констант.

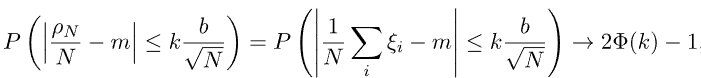
## **. Основные особенности ММК**

Метод имеет две основных особенности. Первая — простая структура вычислительного алгоритма. Вторая — ошибка вычислений, как правило, пропорциональна

, где  — некоторая постоянная (дисперсия), а  — число испытаний. Ясно, что добиться высокой точности на таком пути невозможно. Поэтому обычно говорят, что метод Монте-Карло особенно эффективен при решении тех задач, в которых результат нужен с небольшой точностью.  
  
Однако одну и ту же задачу можно решать различными вариантами метода Монте-Карло, которым отвечают различные значения . Во многих задачах удается значительно увеличить точность, выбрав способ расчета, которому соответствует значительно меньшее значение .

## **. Общая схема ММК**

Пускай требуется вычислить какую-то неизвестную величину m. Смоделируем такую случайную величину , чтобы . Пусть при этом .  
Рассмотрим N независимых случайных величин  (реализаций), распределения которых совпадают с распределением  . Если N достаточно велико, то, согласно центральной предельной теореме, распределение суммы  будет приблизительно нормальным с параметрами .   
  
На основе Центральной предельной теоремы не трудно получить соотношение:

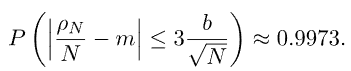


Где функция распределения стандартного нормального распределения.

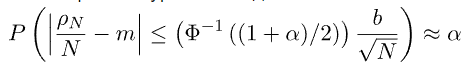
Это соотношение дает и метод расчета m, и оценку погрешности.  
  
Найдем N значений случайной величины . Из указанного соотношения видно, что среднее арифметическое этих значений будет приближенно равно m. С вероятностью близкой к , ошибка такого приближения не превосходит величины . Очевидно, эта ошибка стремится к нулю с ростом N.

В зависимости от целей последнее соотношение используется по-разному.

1. Если взять k=3, то получим так называемое «правило »:



1. Если требуется конкретный уровень надежности вычислений :



Точность вычислений

Как видно из приведенных выше соотношений, точность вычислений зависит от параметра N и величины b – среднеквадратичного отклонения случайной величины   
Рассмотрим вычисление определенного интеграла.

Выберем произвольную случайную величину  с плотностью распределения , определенной на интервале . И рассмотрим случайную величину .  
  
Математическое ожидание последней случайной величины равно:

Таким образом, получаем:

Изображение выглядит как текст, часы

Автоматически созданное описание

Последнее соотношение означает, что если выбрать N значений , то при достаточно большом N

Изображение выглядит как текст

Автоматически созданное описание

Таким образом, для вычисления интеграла, можно использовать практически любую случайную величину . Но дисперсия , а вместе с ней и оценка точности, зависит от того какую случайную величину  взять для проведения расчетов.  
Можно показать, что  будет иметь минимальное значение, когда  пропорционально Выбрать такое значение  в общем случае очень сложно (сложность эквивалентна сложности решаемой задачи), но руководствоваться этим соображением стоит, т.е. выбирать распределение вероятностей по форме схожее с модулем интегрируемой функции.

## **. Вычисление определённых интегралов и снижение дисперсии**

Рассмотрим 2 простейших способа вычисления интегралов с помощью ММК.

1. Метод, связанный с вычислением среднего значения подынтегральной функции

Пускай необходимо вычислить

Выберем произвольную случайную величину   с плотностью распределения , определенной на интервале . При чем

Мат. ожидание

Откуда

Возьмём N значений случайной величины . Она находится из уравнения

Таким образом при большом N мы можем оценить значение исходного интеграла из выражения

Полученное выражение является аналогом суммы Дарбу со случайными узловыми точками.

1. Метод, связанный с интерпретацией интеграла как площади

Снова необходимо вычислить

Рас-м случайную точку с координатами на прямоугольнике

Изображение выглядит как текст, карта, вычерчивание линий

Автоматически созданное описание

Пускай внутри прямоугольника ABEF разбросано N случайных точек, и N’ из них попало под кривую f(x). Тогда из геометрии очевидно, что при больших N

Тогда

Теперь рассмотрим способы уменьшения дисперсии

1. Выделение главной части

Метод основан на расчете основной части интеграла аналитически (если это возможно). И только оставшуюся поправку считают с помощью ММК.

Пускаю удалось подобрать такую функцию, что При чем известен интеграл

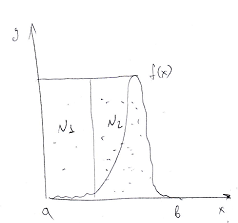
Тогда можно представить

Выберем произвольную случайную величину   с плотностью распределения , определенной на интервале . При чем

Тогда

1. Существенная выборка

Метод основан на предположении, что в более значимых областях (где подынтегральная функция принимает максимальные значения) можно проводить большее количество испытаний.



Из пункта 1.3 знаем. Что погрешность зависит от . Идеальной плотностью является

На практике рекомендуется использовать

1. Симметризация

Пускай подынтегральная функция f(x) монотонная на отрезке интегрирования (a, b). Тогда постоим симметричную ей на этом отрезке – f(b - x). Далее построим полу сумму этих двух функций:

Эта операция уменьшит разброс значений подынтегральной функции по оси Y и увеличит точность получаемого значения интеграла.

1. **Постановка задачи**
2. Вычислить ММК интегралы

Использовать простейшие методы и способы понижения дисперсии, оценить соответствующие значения дисперсии

1. Вычислить ММК определенный интеграл с лабораторной работы №5



Использовать простейшие методы и, если получится, способы понижения дисперсии и сравнить полученный результат с результатами вычислений по формулам Симпсона, прямоугольников и трапеций.

1. **Программа**
2. **import** numpy as np
3. **import** math
4. **import** random
6. **def** our\_func(x):
7. **return** x **\*** math.atan(x)
9. **def** monte\_carlo\_v1(function, a, b, n):
10. sum\_of\_unif\_distr **=** [function(random.uniform(a, b)) **for** kk **in** range(n)]
11. **return** (b **-** a) **/** n **\*** sum(sum\_of\_unif\_distr)
13. **def** monte\_carlo\_v2(function, a, b, n):
14. x **=** np.linspace(a, b, n)
15. max\_f **=** max([function(i) **for** i **in** x])
16. min\_f **=** min([function(i) **for** i **in** x])
17. uppers, downers **=** 0, 0
18. **for** j **in** range(n):
19. point **=** [random.uniform(a, b), random.uniform(0, max\_f)]
20. **if** point[1] < function(point[0]):
21. downers **+=** 1
22. whole\_area **=** (max\_f) **\*** (b **-** a)
23. **return** whole\_area **\*** downers **/** n
25. **def** monte\_carlo\_v1\_advanced\_symm(function, a, b, n):
26. sum\_of\_unif\_distr **=** 0
27. **for** kk **in** range(n):
28. rand\_num **=** random.uniform(a, b)
29. sum\_of\_unif\_distr **+=** 1 **/** 2 **\*** (function(rand\_num) **+** function(b **-** rand\_num))
30. **return** (b **-** a) **\*** sum\_of\_unif\_distr **/** n
32. **def** monte\_carlo\_v1\_advanced\_mainp(function, a, b, n, helper\_function):
33. sum\_of\_unif\_distr **=** 0
34. **for** kk **in** range(n):
35. rand\_num **=** random.uniform(a, b)
36. sum\_of\_unif\_distr **+=** (function(rand\_num) **-** helper\_function(rand\_num))
37. **return** (b **-** a) **/** n **\*** sum\_of\_unif\_distr
39. **def** monte\_carlo\_v1\_advanced\_distr(function, a, b, n, distribution):
40. sum\_of\_unif\_distr **=** 0
41. **for** kk **in** range(n):
42. rand\_num **=** random.uniform(a, b)
43. sum\_of\_unif\_distr **+=** function(rand\_num) **/** distribution(rand\_num)
44. **return** sum\_of\_unif\_distr **/** n
46. print('Интегрирование с помощью метода ММК с вычислением среднего')
47. print(f'Значение интеграла exp(x) = {monte\_carlo\_v1(math.exp, 0, 1, 50000)}')
48. print(f'Значение интеграла sin(x) = {monte\_carlo\_v1(math.sin, 0, math.pi/2, 50000)}')
49. print(f'Значение интеграла x\*arctg(x) = {monte\_carlo\_v1(our\_func, 0, math.sqrt(3), 50000)}')
50. print(' ')
51. print('Интегрирование с помощью метода ММК с интерпретацией интеграла как площади')
52. print(f'Значение интеграла exp(x) = {monte\_carlo\_v2(math.exp, 0, 1, 50000)}')
53. print(f'Значение интеграла sin(x) = {monte\_carlo\_v2(math.sin, 0, math.pi/2, 50000)}')
54. print(f'Значение интеграла x\*arctg(x) = {monte\_carlo\_v2(our\_func, 0, math.sqrt(3), 50000)}')
55. print(' ')
56. print('Интегрирование с помощью метода ММК с вычислением среднего и со снижением дисперсии методом симметризации')
57. print(f'Значение интеграла exp(x) = {monte\_carlo\_v1\_advanced\_symm(math.exp, 0, 1, 50000)}')
58. print(f'Значение интеграла sin(x) = {monte\_carlo\_v1\_advanced\_symm(math.sin, 0, math.pi/2, 50000)}')
59. print(f'Значение интеграла x\*arctg(x) = {monte\_carlo\_v1\_advanced\_symm(our\_func, 0, math.sqrt(3), 50000)}')
60. print(' ')
61. print('Интегрирование с помощью метода ММК с вычислением среднего и со снижением дисперсии методом главной части')
62. print(f'Значение интеграла exp(x) = {monte\_carlo\_v1\_advanced\_mainp(math.exp, 0, 1, 50000, lambda x: 1 + x) + 3 / 2}')
63. print(f'Значение интеграла sin(x) = {monte\_carlo\_v1\_advanced\_mainp(math.sin, 0, math.pi/2, 50000, lambda x: x) + (math.pi \*\* 2) / 8}')
64. print(f'Значение интеграла x\*arctg(x) = {monte\_carlo\_v1\_advanced\_mainp(our\_func, 0, math.sqrt(3), 50000, lambda x: x \* x) + math.sqrt(3)}')
65. print(' ')
66. print('Интегрирование с помощью метода ММК с вычислением среднего и со снижением дисперсии методом существенной выборки')
67. print(f'Значение интеграла exp(x) = {monte\_carlo\_v1\_advanced\_distr(math.exp, 0, 1, 50000, lambda x: 2 \* (1 + x) / 3)}')
68. print(f'Значение интеграла sin(x) = {monte\_carlo\_v1\_advanced\_distr(math.sin, 0, math.pi/2, 50000, lambda x: 8 \* x/(math.pi \*\* 2))}')
69. print(f'Значение интеграла x\*arctg(x) = {monte\_carlo\_v1\_advanced\_distr(our\_func, 0, math.sqrt(3), 50000, lambda x: x \* x/math.sqrt(3))}')
70. **Результаты**

Точные значения интегралов

Результаты работы программы

**Интегрирование с помощью метода ММК с вычислением среднего**

**Интегрирование с помощью метода ММК с интерпретацией интеграла как площади**

**Интегрирование с помощью метода ММК с вычислением среднего и со снижением дисперсии методом симметризации**

**Интегрирование с помощью метода ММК с вычислением среднего и со снижением дисперсии методом главной части**

**Интегрирование с помощью метода ММК с вычислением среднего и со снижением дисперсии методом существенной выборки**

Для сравнения приведем результаты вычисления последнего интеграла методом прямоугольников, трапеций и Симпсона

1. **Выводы**

В ходе проведения работы было выяснено, что методы Монте-Карло значительно уступаю в точности методам прямоугольников, трапеция и парабол. Если сравнивать между собой два простейших метода вычисления интеграла с помощью ММК, то можно заметить, что их точность зависит от рода подынтегральной функции. Для экспоненты и синуса оказался более точным метод, связанный с интерпретацией интеграла как площади, для арктангенса же – метод, связанный с нахождением среднего значения функции. Эффективность методов уменьшения дисперсии так же зависит от характера подынтегральной функции. Лучше всего на представленных функциях себя проявил метод симметризации. Он значительно увеличил точность всех полученных результатов. Отдельно хотелось бы отметить, что для синуса немного точнее сработал метод выделения главной части. Вероятно, это связано с тем, что ряд Тейлора для синуса наиболее приближен к реальному значению функции, нежели аналогичные ряды для экспоненты и арктангенса. Метод существенной выборки оказался несостоятельным, он не только не увеличил точность рассчитанных значений, но и в некоторых случаях даже ухудшил их. Вероятно, это связано с недостаточно точным подбором оптимальной плотности вероятности случайной величины.