|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **Информатика и системы управления**

КАФЕДРА **Программное обеспечение ЭВМ и информационные технологии**

**Лабораторная работа №5.**

**«Построение и программная реализация алгоритмов численного интегрирования»**

Студент **Леонов Владислав Вячеславович**

Группа **ИУ7-46Б**

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Леонов В.В.

*подпись, дата фамилия, и.о.*

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Градов В.М.

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2021 г.*

**Оглавление**

[Цель работы 3](#_Toc69572445)

[Исходные данные 3](#_Toc69572446)

[Описание алгоритма 3](#_Toc69572447)

[Код программы 6](#_Toc69572448)

[Результаты работы 10](#_Toc69572449)

[Ответы на контрольные вопросы 12](#_Toc69572450)

# Цель работы

Получение навыков построения алгоритма вычисления двукратного интеграла с использованием квадратурных формул Гаусса и Симпсона.

# Исходные данные

*1. Двукратный интеграл при фиксированном значении параметра .*

где

*2. Количество узлов N1 и N2 для последовательного интегрирования по двум направлениям.*

По одному направлению использовать формулу Гаусса, а по-другому – формулу Симпсона.

# Описание алгоритма

***1. Квадратурная формула Гаусса.***

Пусть интеграл вычисляется на стандартном интервале Задача состоит в том, чтобы подобрать точки и коэффициенты , так чтобы квадратурная формула была точной для всех полиномов наивысшей возможной степени:

Получаем систему нелинейных уравнений:

Для решения данной систему будем использовать полиномы Лежандра:

Можно показать, что узлами формулы Гаусса являются нули многочлена Лежандра степени *n.* Таким образом, нужно найти все корни многочлена Лежандра, после чего решить систему линейных алгебраических уравнений (решаем методом Гаусса).

Для вычисления интеграла на произвольном интеграле , для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной:

***2. Формула Симпсона.***

Суть метода заключается в приближении подынтегральной функции на отрезке интерполяционным многочленом второй степени , то есть приближение графика функции на отрезке параболой.

Следует отметить, что если подынтегральная функция не имеет соответствующих производных, то указанный порядок точности не достигается. Так, если на отрезке интегрирования не существуют 3-я и 4-я производные, то порядок точности формулы Симпсона будет только второй, .

# Код программы

Код программы представлен ниже.

|  |
| --- |
| Файл ***main.py*** |
| **from** io\_funcs **import** get\_nodes, get\_parameter, get\_integrators  **from** function **import** function  **from** statistics **import** cmp\_integ\_value\_by\_param, graph\_label, graph\_show  **from** config **import** PARAM\_LIMITS, INTEGRATE\_LIMITS  **from** integrators **import** integrate, SimpsonException      **def** main():    **while** True:  nodes = get\_nodes()  parameter = get\_parameter()  integrators = get\_integrators()    **try**:  parameter\_integral = **lambda** parameter: integrate(  function(parameter), INTEGRATE\_LIMITS, nodes, integrators)    **print**("Integral value:", parameter\_integral(parameter))    cmp\_integ\_value\_by\_param(parameter\_integral, PARAM\_LIMITS,  graph\_label(nodes, integrators))    continue\_condition = int(  input("Enter 0 to exit or 1 to continue:"))  **if** continue\_condition == 0:  **break**    **except** SimpsonException:  **print**("Simpson method cannot be applied.")    graph\_show()      **if** \_\_name\_\_ == "\_\_main\_\_":  main() |

|  |
| --- |
| Файл ***config.py*** |
| **import** math  **from** data\_classes **import** ParamLimits, IntegrateLimits    *# Parameter limits*  PSTART, PEND, PSTEP = 0.05, 10.0, 0.05    *# Inner integrate limits*  I\_LEFT, I\_RIGHT = 0, math.pi / 2    *# Outer integrate limits*  O\_LEFT, O\_RIGHT = 0, math.pi / 2      PARAM\_LIMITS = ParamLimits(PSTART, PEND, PSTEP)    INTEGRATE\_LIMITS = [  IntegrateLimits(I\_LEFT, I\_RIGHT),  IntegrateLimits(O\_LEFT, O\_RIGHT)  ] |

|  |
| --- |
| Файл ***data\_classes.py*** |
| **from** dataclasses **import** dataclass      @dataclass  **class** ParamLimits:  start: float  end: float  step: float    **def** \_\_init\_\_(self, start: float, end: float, step: float):  self.start = start  self.end = end  self.step = step      @dataclass  **class** IntegrateLimits:  left: float  right: float    **def** \_\_init\_\_(self, left: float, right: float):  self.left = left  self.right = right |

|  |
| --- |
| Файл ***function.py*** |
| **from** math **import** cos, sin, exp, pi      **def** function(param):  sub\_func = **lambda** theta, phi: 2 \* cos(theta) / (1 - (sin(theta)\*\*2) \* (cos(phi)\*\*2))  main\_func = **lambda** theta, phi: (4 / pi) \* (1 - exp(-param \* sub\_func(theta, phi))) \* cos(theta) \* sin(theta)  **return** main\_func      **def** function\_2\_to\_1(function, theta):  **return** **lambda** phi: function(theta, phi) |

|  |
| --- |
| Файл ***integrators.py*** |
| **from** data\_classes **import** IntegrateLimits  **from** function **import** function\_2\_to\_1      **class** SimpsonException(Exception):  **pass**      **def** integrate(func, int\_limits: list[IntegrateLimits], nodes: tuple[int], int\_funcs: list):  inner\_int\_func = **lambda** theta: int\_funcs[1](function\_2\_to\_1(func, theta), int\_limits[1], nodes[1])    **return** int\_funcs[0](inner\_int\_func, int\_limits[0], nodes[0])      **def** simpson(func, int\_limits: IntegrateLimits, nodes\_amount: int):    **if** (nodes\_amount < 3 **or** nodes\_amount % 2 == 0):  **raise** SimpsonException    h = (int\_limits.right - int\_limits.left) / (nodes\_amount - 1)  x = int\_limits.left  int\_value = 0    **for** iteration **in** range((nodes\_amount - 1) // 2):  int\_value += func(x) + 4 \* func(x + h) + func(x + 2 \* h)  x += 2 \* h    **return** int\_value \* h / 3      **def** gauss(func, int\_limits: IntegrateLimits, nodes\_amount: int):  t\_list, a\_list = leggauss(nodes\_amount)    int\_value = 0    **for** i **in** range(nodes\_amount):  int\_value += (int\_limits.right -  int\_limits.left) / 2 \* a\_list[i] \* func(  arg\_format(t\_list[i], int\_limits))    **return** int\_value      **def** arg\_format(t: float, int\_limits: IntegrateLimits):  **return** (int\_limits.left + int\_limits.right) / 2 + (int\_limits.right - int\_limits.left) \* t / 2 |

|  |
| --- |
| Файл ***io\_funcs.py*** |
| **from** integrators **import** gauss, simpson  **def** get\_nodes() -> tuple[int, int]:  inner\_nodes = int(input("Inner nodes:"))  outer\_nodes = int(input("Outer nodes:"))  **return** inner\_nodes, outer\_nodes  **def** get\_parameter() -> float:  param = float(input("Parameter value:"))  **return** param    **def** get\_integrators():  SIMPSON = 0  GAUSS = 1  outer\_mode = int(input("Outer mode: "))  inner\_mode = int(input("Inner mode: "))  inner\_func = gauss **if** inner\_mode == GAUSS **else** simpson  outer\_func = gauss **if** inner\_mode == GAUSS **else** simpson  **return** outer\_func, inner\_func |

|  |
| --- |
| Файл ***statistics.py*** |
| **import** matplotlib.pyplot **as** plt  **from** integrators **import** simpson  **from** data\_classes **import** ParamLimits  **from** numpy **import** arange      **def** cmp\_integ\_value\_by\_param(parameter\_integral, param\_limits: ParamLimits,  graph\_label: str):  params = arange(param\_limits.start, param\_limits.end + param\_limits.step,  param\_limits.step)  integ\_values = list()  **for** param **in** params:  integ\_values.append(parameter\_integral(param))  plt.plot(params, integ\_values, label=graph\_label)      **def** graph\_label(nodes, funcs):  label = "Nodes ({0},{1});".format(\*nodes)  label += "Simpson" **if** funcs[0] == simpson **else** "Gauss"  label += "-" + ("Simpson" **if** funcs[1] == simpson **else** "Gauss")  **return** label      **def** graph\_show():  plt.legend()  plt.ylabel("Integral value")  plt.xlabel("Parameter value")  plt.show() |

# Результаты работы

***1. Описать алгоритм вычисления n корней полинома Лежандра n-ой степени при реализации формулы Гаусса.***

Отрезок следует разбить на 2*n* равных частей, и рассмотреть каждую часть разбиения на наличие смены знаков на концах. Необходимо, чтобы нашлось таких отрезков, в противном случае нужно увеличить значение числа разбиения отрезка в два раза и повторять данную процедуру, пока не будет найдено нужное количество описанных выше типов отрезков.

После этого находим корни полинома Лежандра на отрезках с разными знаками значения функции на концах методом половинного деления. Используем данный метод, ввиду простоты реализации и ввиду того, что он всегда сходится и работает достаточно быстро.

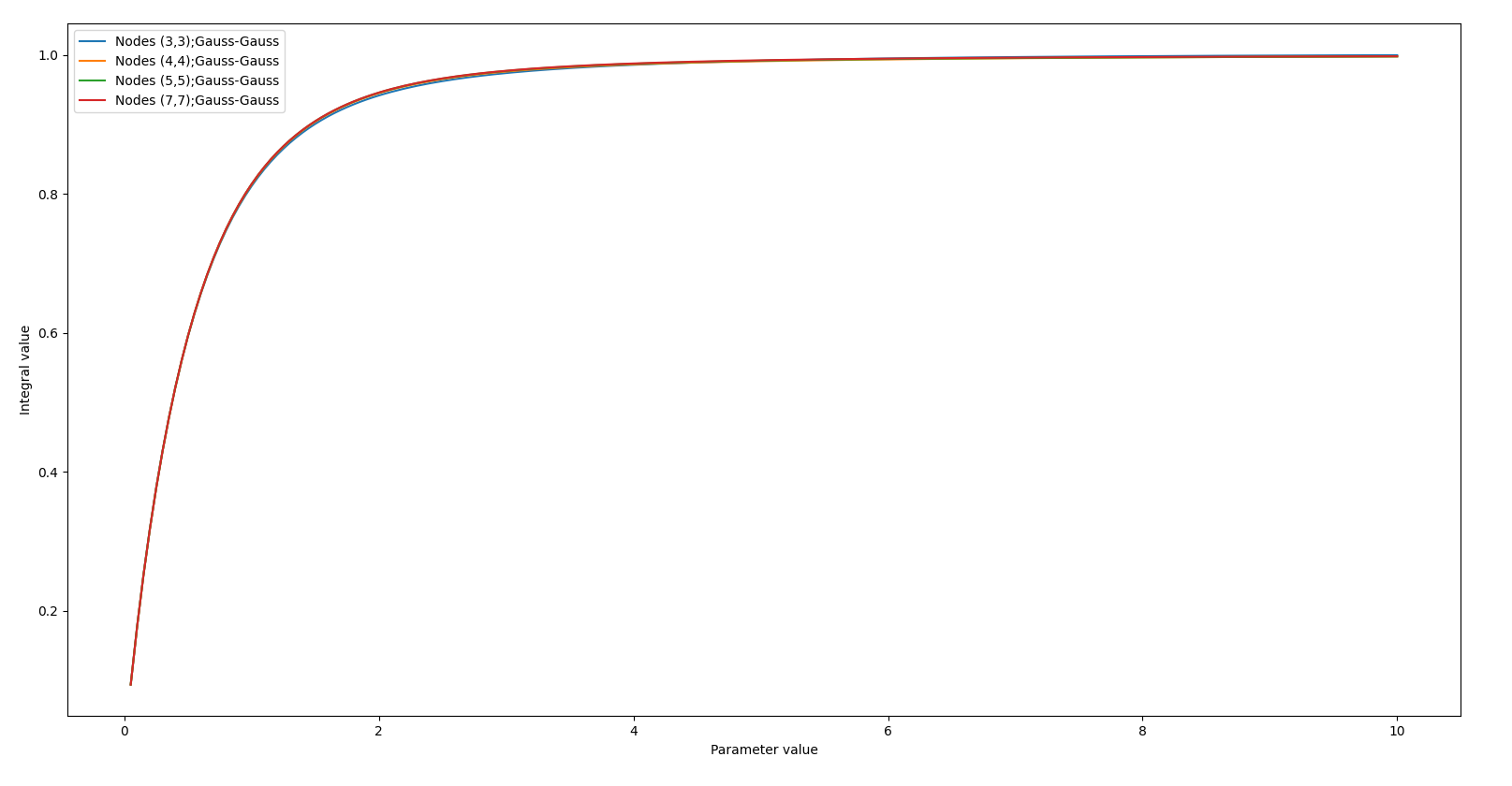
***2. Исследовать влияние количества выбираемых узлов сетки по каждому направлению на точность расчётов.***

*Алгоритмы интегрирования: Гаусс-Симпсон, .*

*Значение интеграла, полученное при помощи MatLab:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | *Количество узлов для метода Гаусса* | | | | | |
| *3* | *5* | *7* | *9* | *11* | *13* |
| *Узлы для метода Симпсона* | *3* | *0.727855* | *0.728062* | *0.727977* | *0.727992* | *0.727990* | *0.727990* |
| *5* | *0.729585* | *0.729697* | *0.729824* | *0.729778* | *0.729787* | *0.729787* |
| *7* | *0.729476* | *0.729932* | *0.729918* | *0.729914* | *0.729904* | *0.729908* |
| *9* | *0.729505* | *0.729948* | *0.729928* | *0.729932* | *0.729931* | *0.729929* |
| *11* | *0.729535* | *0.729945* | *0.729942* | *0.729932* | *0.729938* | *0.729936* |
| *13* | *0.729547* | *0.729944* | *0.729946* | *0.729935* | *0.729939* | *0.729939* |

***3. Построить график зависимости в диапазоне изменения Указать при каком количестве узлов получены результаты.***



Из приведённого выше графика можно сделать вывод о том, что использование метода Гаусса для интегрирования обеспечивает очень высокую точность и даже при небольшом количестве узлов можно получать значение двукратных интегралов с маленькой погрешностью. Заметим, что при количестве узлов получаем очень близкие значения, причем данный интеграл имеет предел равный 1, при .

# Ответы на контрольные вопросы

***1. В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается.***

Следует отметить, что если подынтегральная функция не имеет соответствующих производных, то указанный порядок точности не достигается. Так, если на отрезке интегрирования не существуют 3-я и 4-я производные, то порядок точности формулы Симпсона будет только второй, .

***2. Построить формулу Гаусса численного интегрирования при одном узле.***

***3. Построить формулу Гаусса численного интегрирования при двух узлах.***

***4. Получить обобщенную кубатурную формулу, аналогичную (6.6) из лекции №6, для вычисления двойного интеграла методом последовательного интегрирования на основе формулы трапеций с тремя узлами по каждому направлению.***