| 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 (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**ОТЧЕТ**

| **по лабораторной работе №** | 5 |
| --- | --- |



Построение и программная реализация алгоритмов численного интегрирования

**Дисциплина:** Вычислительные алгоритмы

| Студент | ИУ7И - 46Б |  |  | Андрич К. |
| --- | --- | --- | --- | --- |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В.М. Градов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

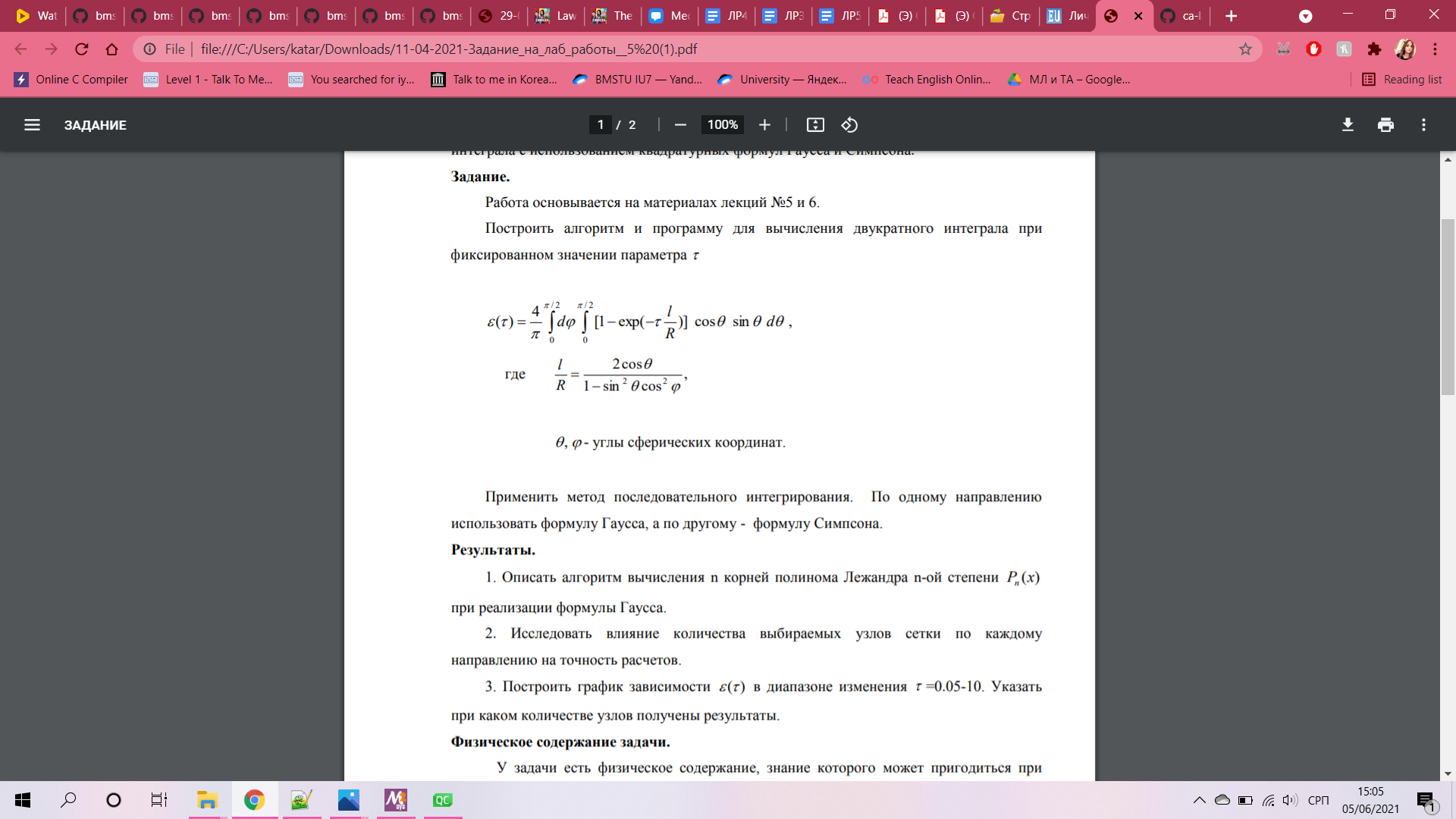
Москва, 2021

Цель работы

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

Задание

Построить алгоритм и программу для вычисления двукратного интеграла при фиксированном значении параметра.



Применить метод последовательного интегрирования. По одному направлению использовать формулу Гаусса, а по другому - формулу Симпсона.

Код программы

В программе есть 2 файлов: main.py и functions.py

main.py

| from functions import \* import math  def main():  t, rc = get\_t()  if rc == 1:  return  else:  rc = 1  while not rc == 2:  n1, m1 = get\_method()  n2, m2 = get\_method()  t\_solve = lambda t: get\_res(t\_func(t), 0, math.pi / 2, 0, math.pi / 2, m1, m2, n1, n2)  result = t\_solve(t)  print("Result while τ = %.2f: %.6f" %(t, result))  label = make\_label(m1, m2, n1, n2)  prepare\_graph(t\_solve, label)  rc = menu()  show\_graph()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| --- |

functions.py

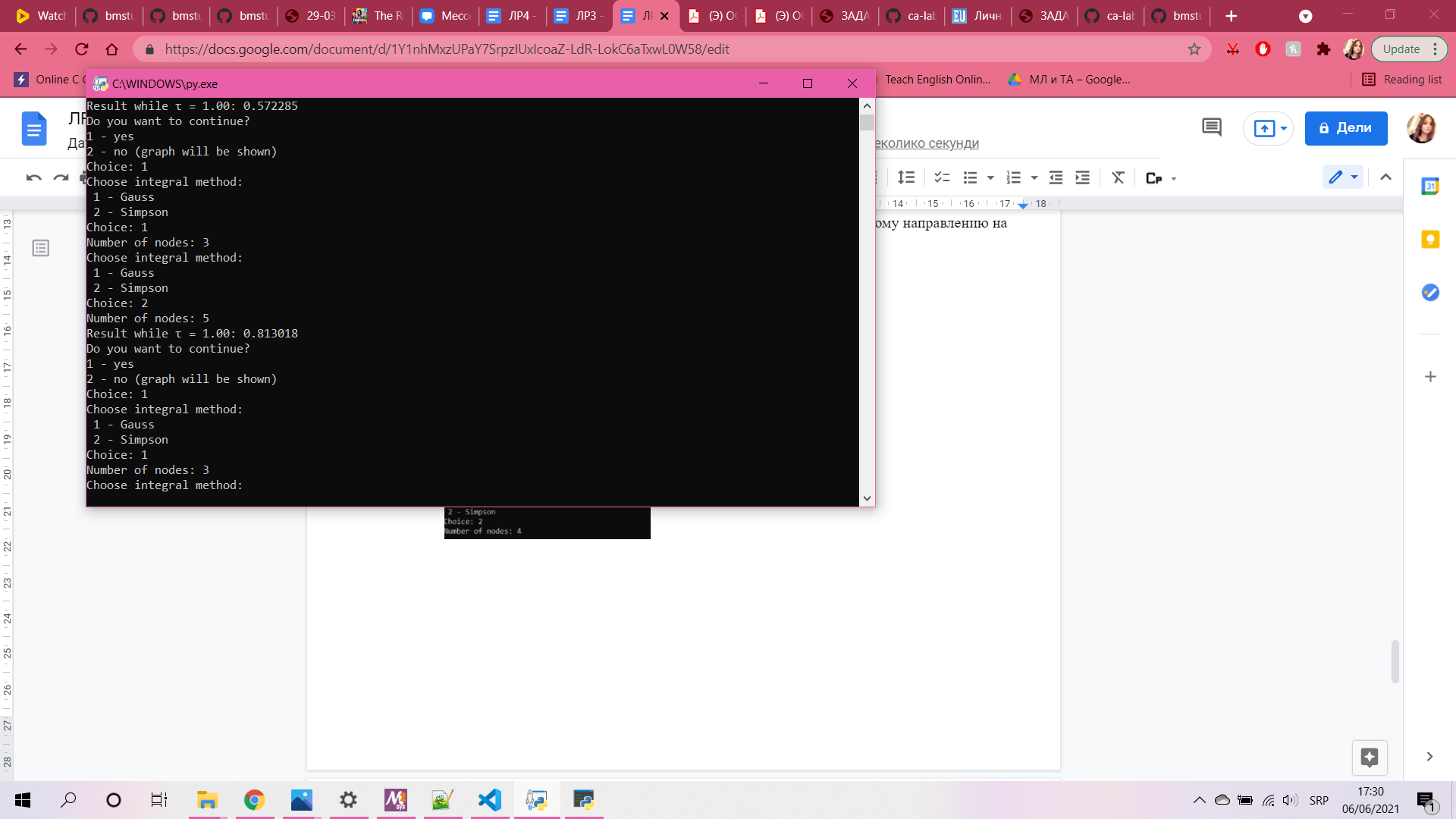
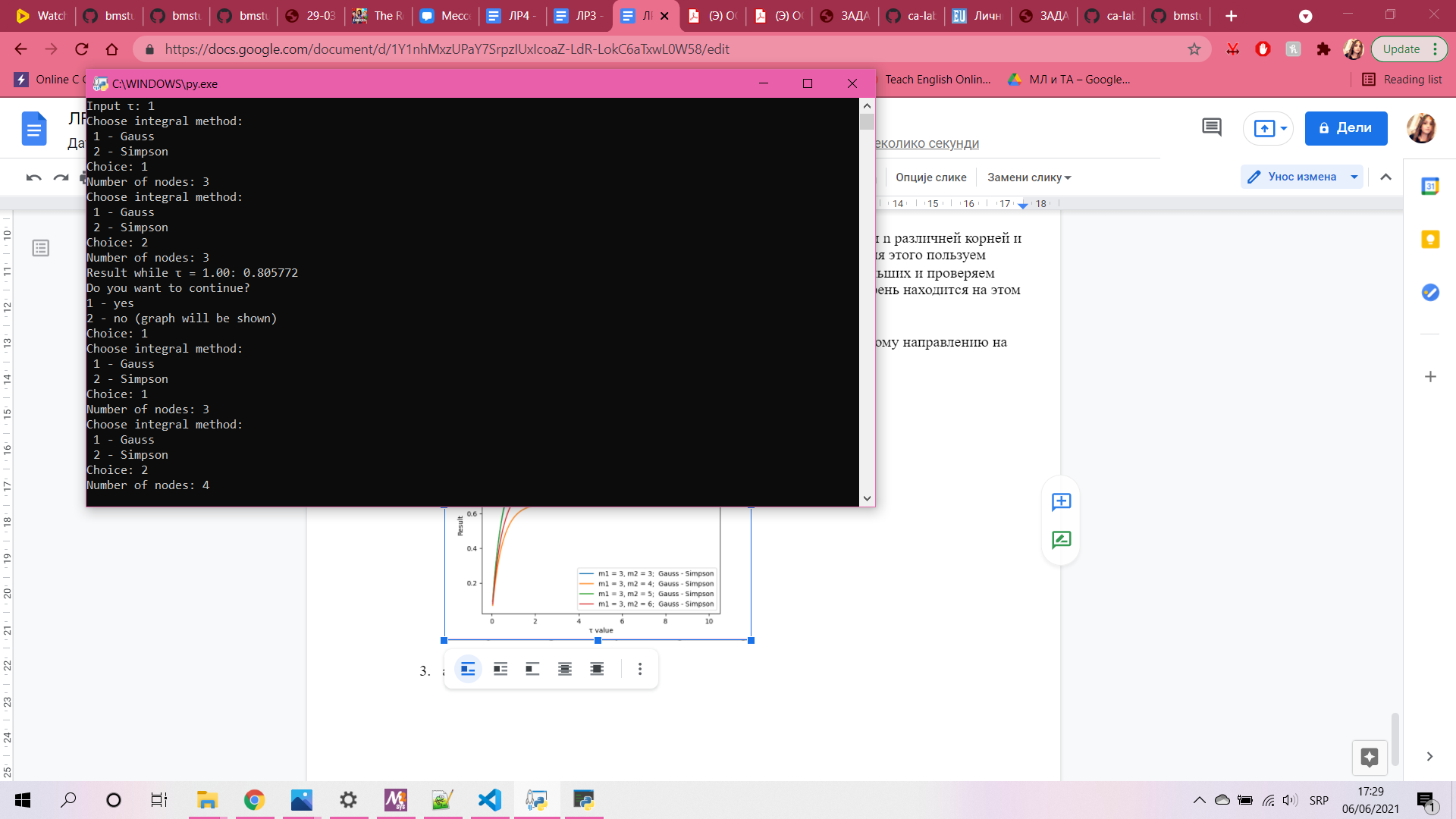
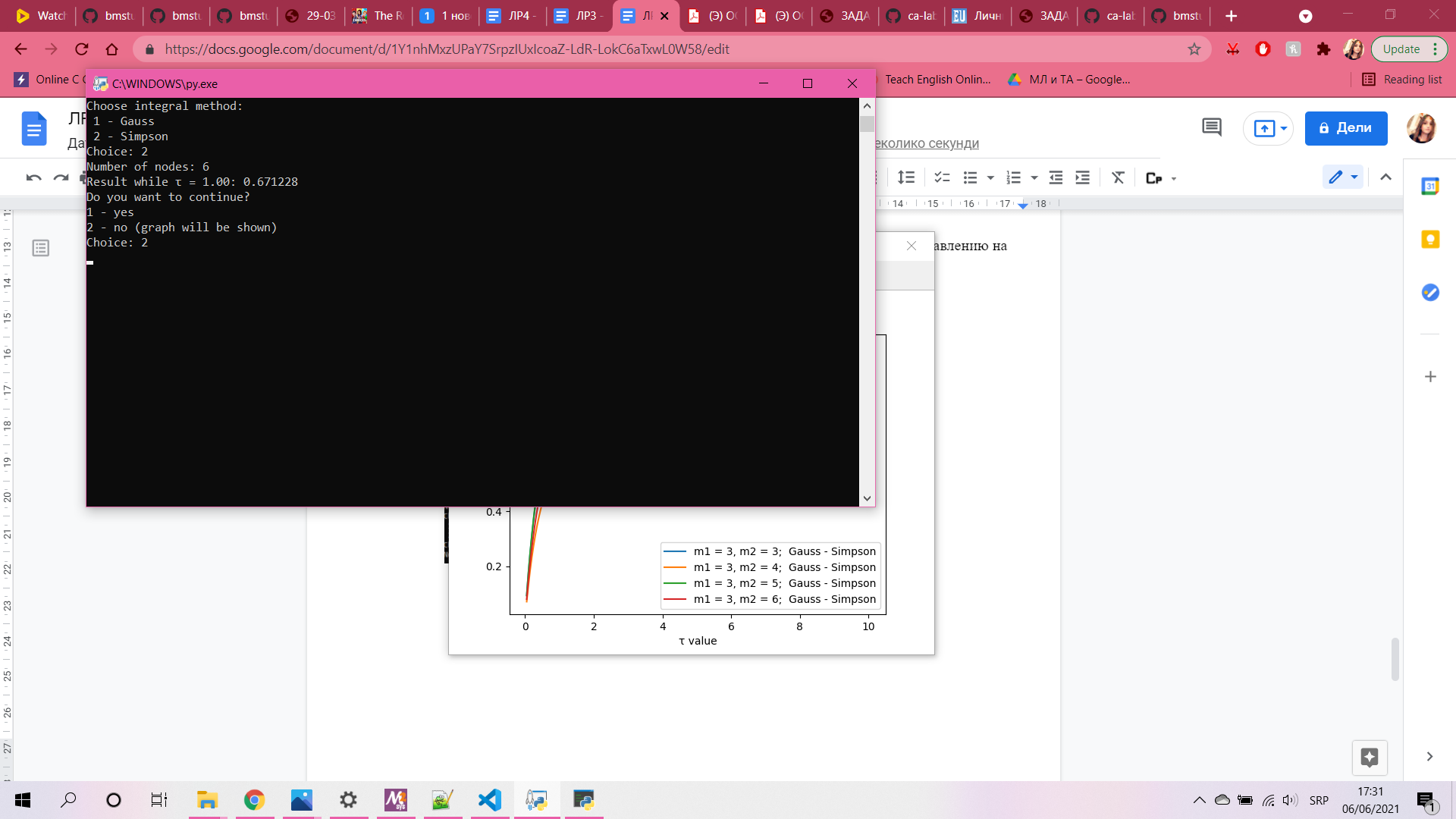
| from numpy.linalg.linalg import \_multi\_dot\_matrix\_chain\_order from numpy.polynomial.legendre import leggauss from numpy import arange import matplotlib.pyplot as plt import math  def menu():  print("Do you want to continue?")  print("1 - yes")  print("2 - no (graph will be shown)")  try:  rc = float(input("Choice: "))  except:  print("Input error")  return rc  def get\_t():  t, rc = 0, 0  try:  t = float(input("Input τ: "))  except:  print("Input error")  rc = 1  return t, rc  def get\_method():  print("Choose integral method: ")  print(" 1 - Gauss")  print(" 2 - Simpson")  n = 0  try:  n = int(input("Choice: "))  except:  print("Input error")  m = 0  if n == 1:  min\_value = 0  else:  min\_value = 2  while (m <= min\_value):  try:  m = int(input("Number of nodes: "))  except:  print("Input error")  if min\_value == 2 and m <= 2:  print("For Simpson's method number of nodes must be greater than 2")  return n, m  def Gauss(function, a, b, n):  args, coeffs = leggauss(n)  result = 0  for i in range(n):  result += (b - a) / 2 \* coeffs[i] \* function((b + a) / 2 + (b - a) \* args[i] / 2)  return result  def Simpson(function, a, b, n):  h = (b - a) / (n - 1)  x = a  result = 0  for i in range((n - 1) // 2):  result += function(x) + 4 \* function(x + h) + function(x + 2 \* h)  x += 2 \* h  result \*= (h / 3)  return result  def t\_func(t):  coeff = lambda x, y: 2 \* math.cos(x) / (1 - (math.sin(x) \*\* 2) \* (math.cos(y) \*\* 2))  func = lambda x, y: (4 / math.pi) \* (1 - math.exp(-t \* coeff(x, y))) \* math.cos(x) \* math.sin(x)  return func  def get\_res(function, lim1, lim2, lim3, lim4, m1, m2, n1, n2):  if n1 == 1:  method1 = Gauss  else:  method1 = Simpson  if n2 == 1:  method2 = Gauss  else:  method2 = Simpson  internal = lambda x: method1(lambda y: function(x, y), lim3, lim4, m2)  result = method2(internal, lim1, lim2, m1)  return result  def make\_label(m1, m2, n1, n2):  label = "m1 = " + str(m1) + ", m2 = " + str(m2) + "; "  if n1 == 1:  label += "Gauss "   else:  label += "Simpson "  if n2 == 1:  label += "- Gauss"   else:  label += "- Simpson"  return label  def prepare\_graph(function, label):  X, Y = [], []  for t in arange(0.05, 10 + 0.05, 0.05):  X.append(t)  Y.append(function(t))  plt.plot(X, Y, label = label)  def show\_graph():  plt.legend()  plt.ylabel("Result")  plt.xlabel("τ value")  plt.show() |
| --- |

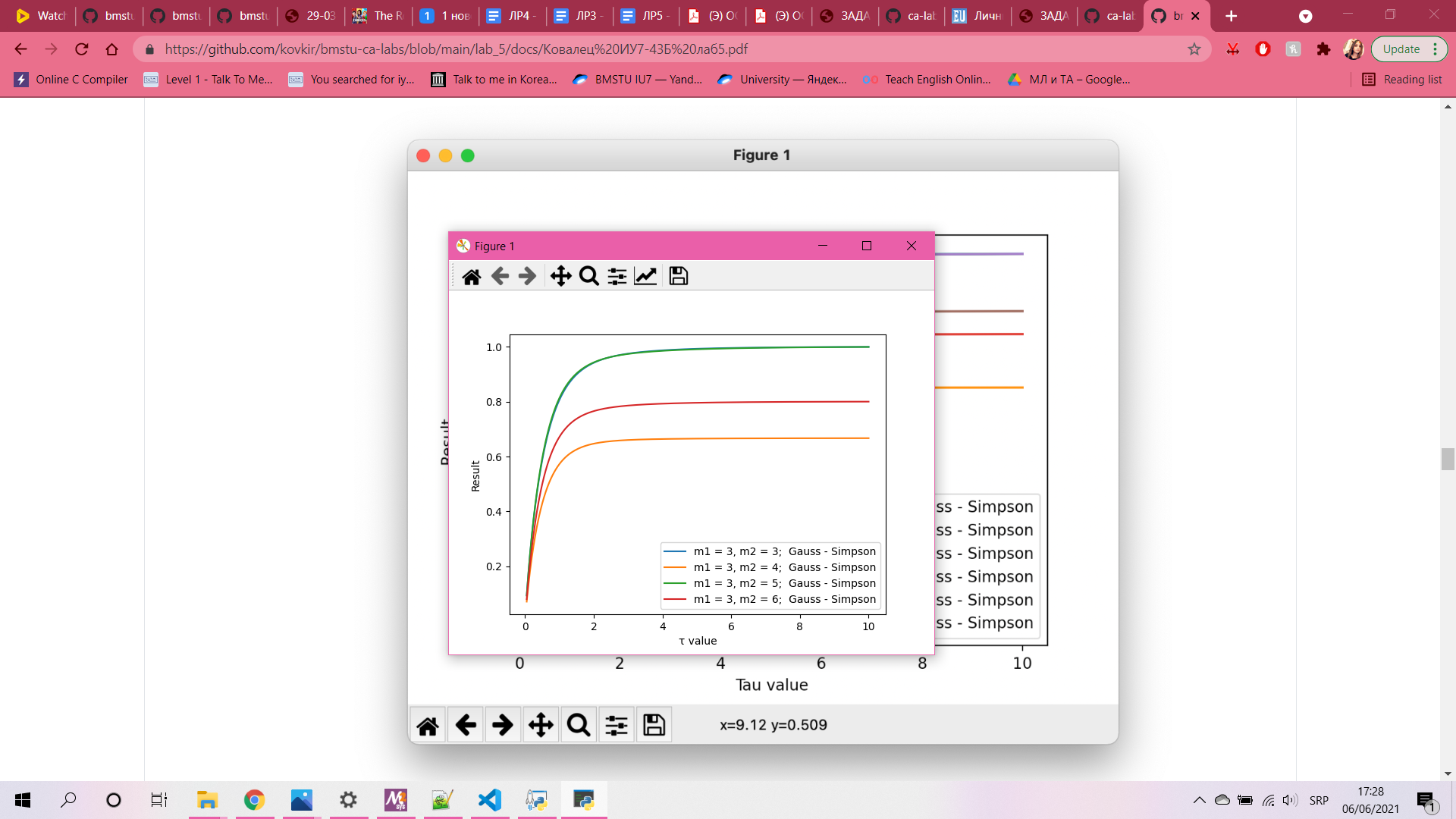
Результаты работы

1. Описать алгоритм вычисления n корней полинома Лежандра n-ой степени P (x) n при реализации формулы Гаусса.

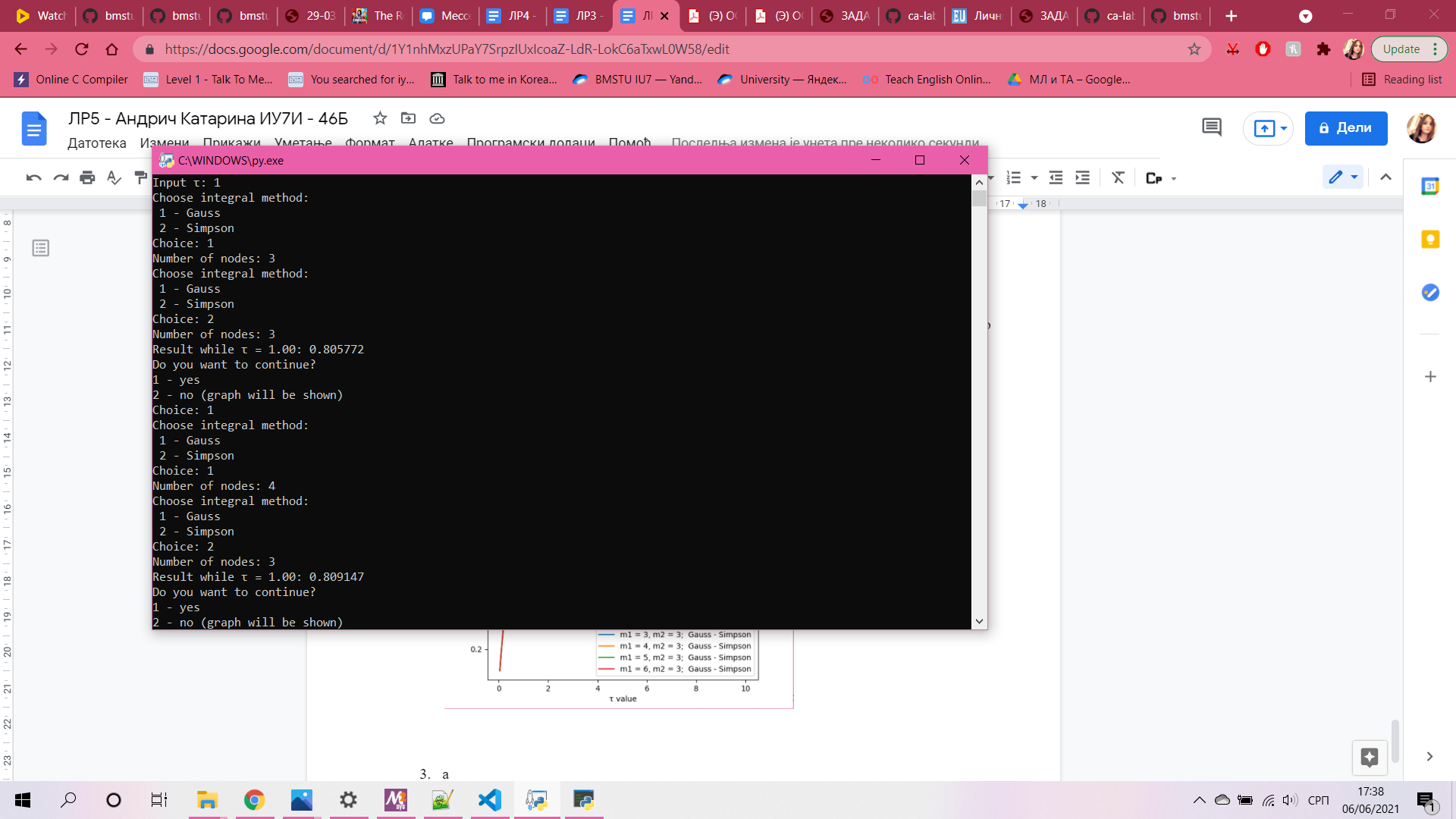
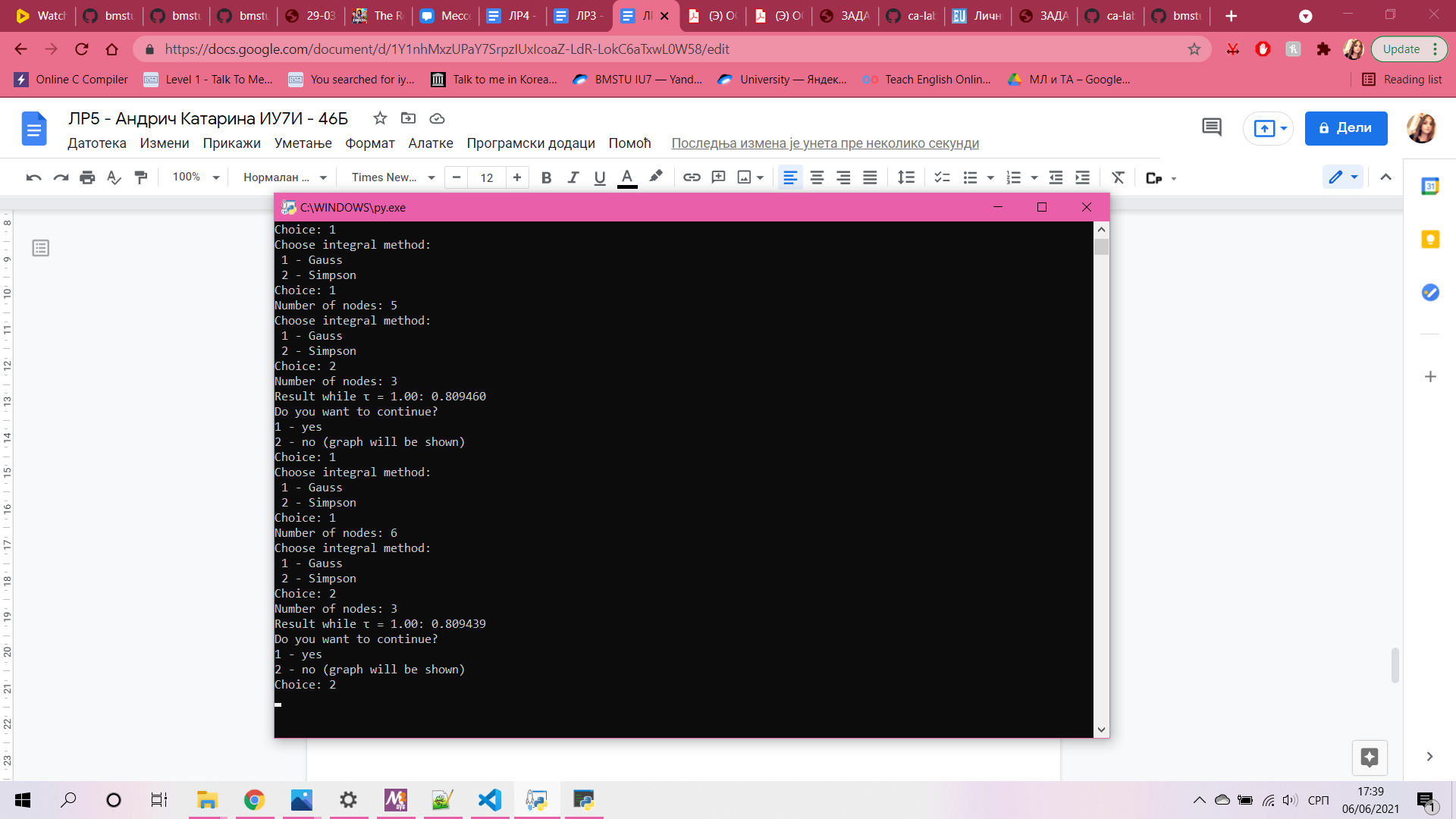
Из-за того что на отрезке [-1, 1] у полинома Лежандра n-ой степени n различней корней и из каждого корня x следует -x можем искать их только на (0, 1]. Для этого пользуем метод половинного деления. Интервал разбиваем на несколько меньших и проверяем каждый. Если у отрезка разные знаки на концах, это значит что корень находится на этом отрезке, а если один из них равен 0 это значит что корень найден.

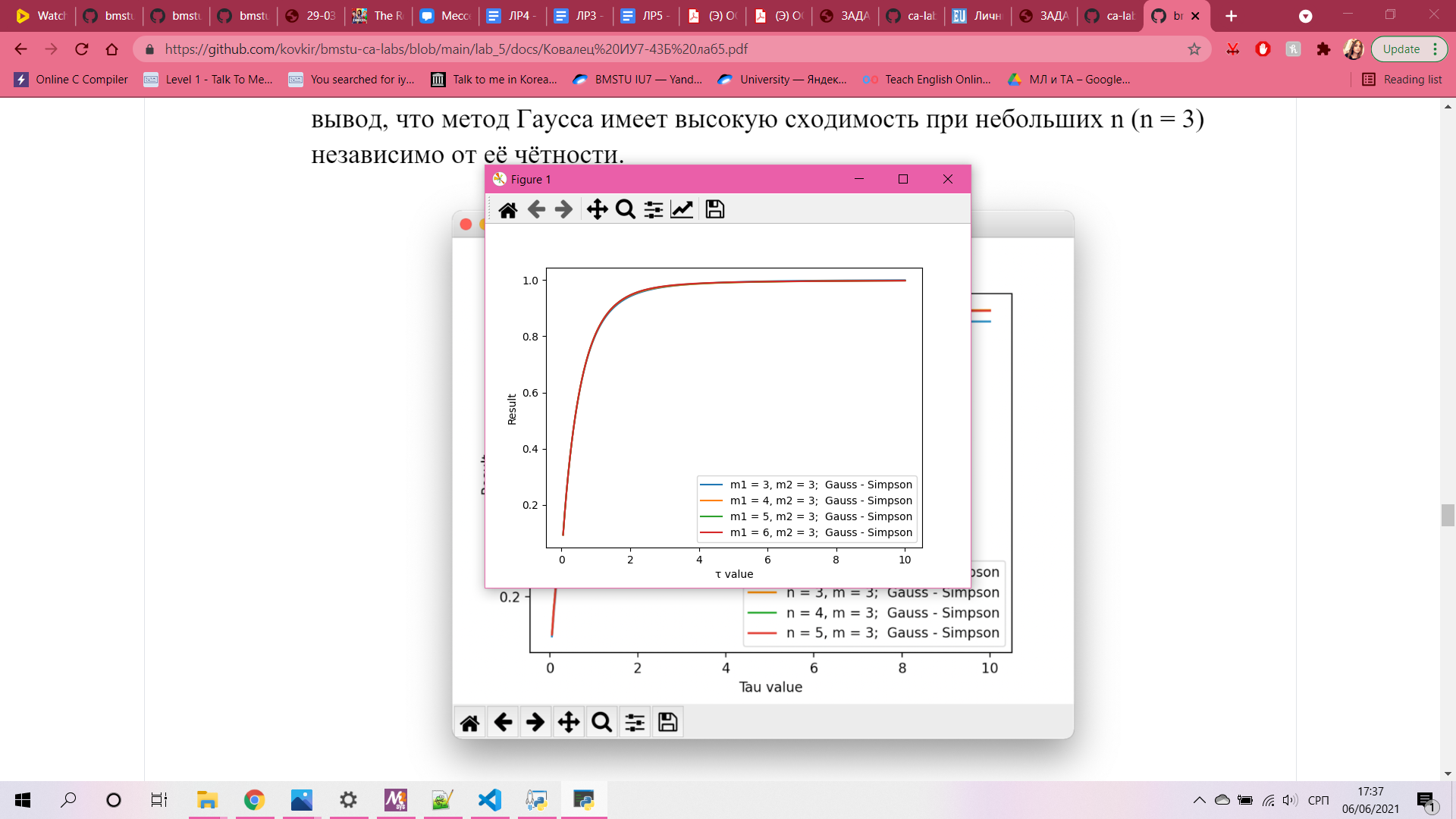
1. Исследовать влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов.



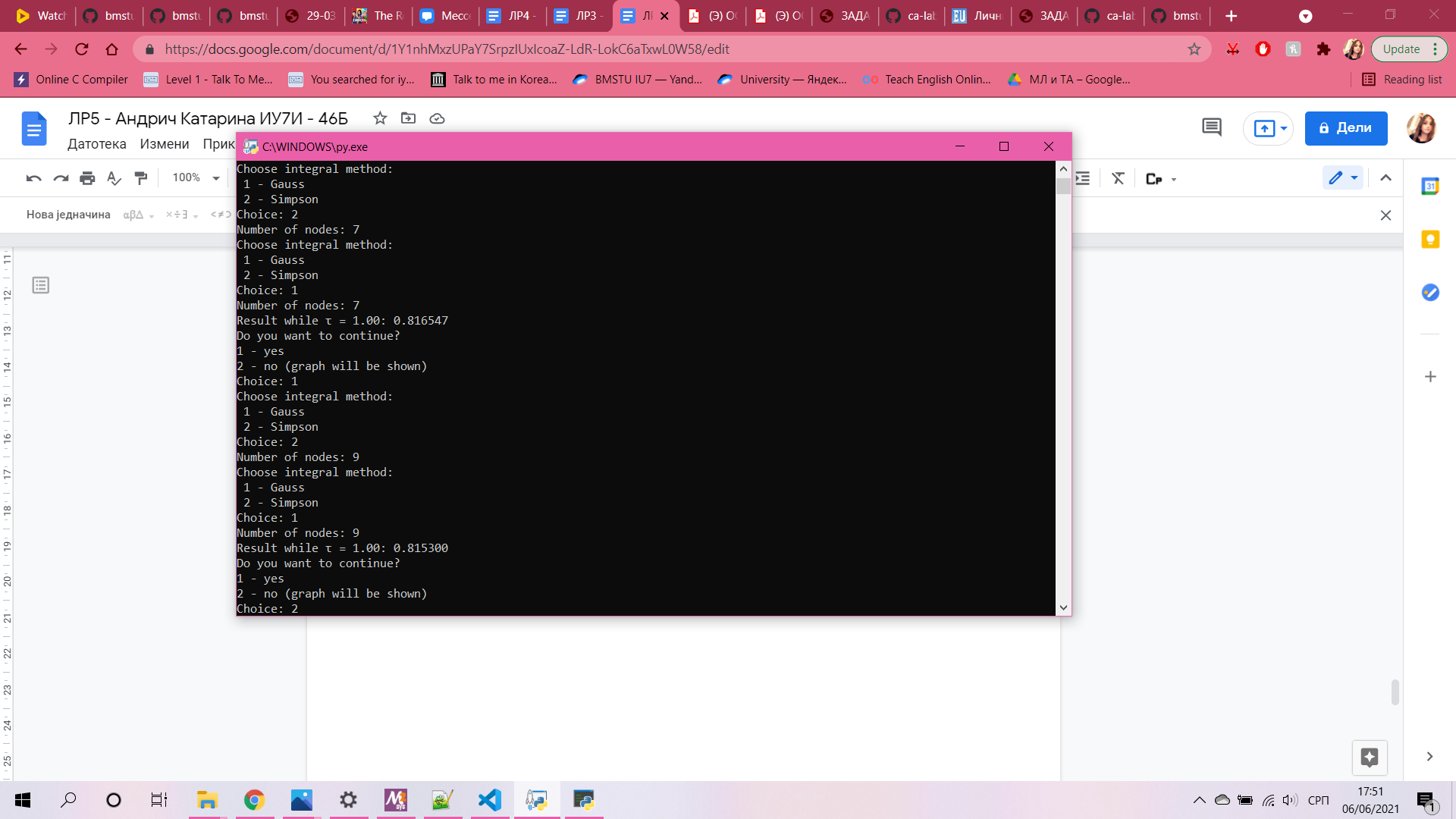
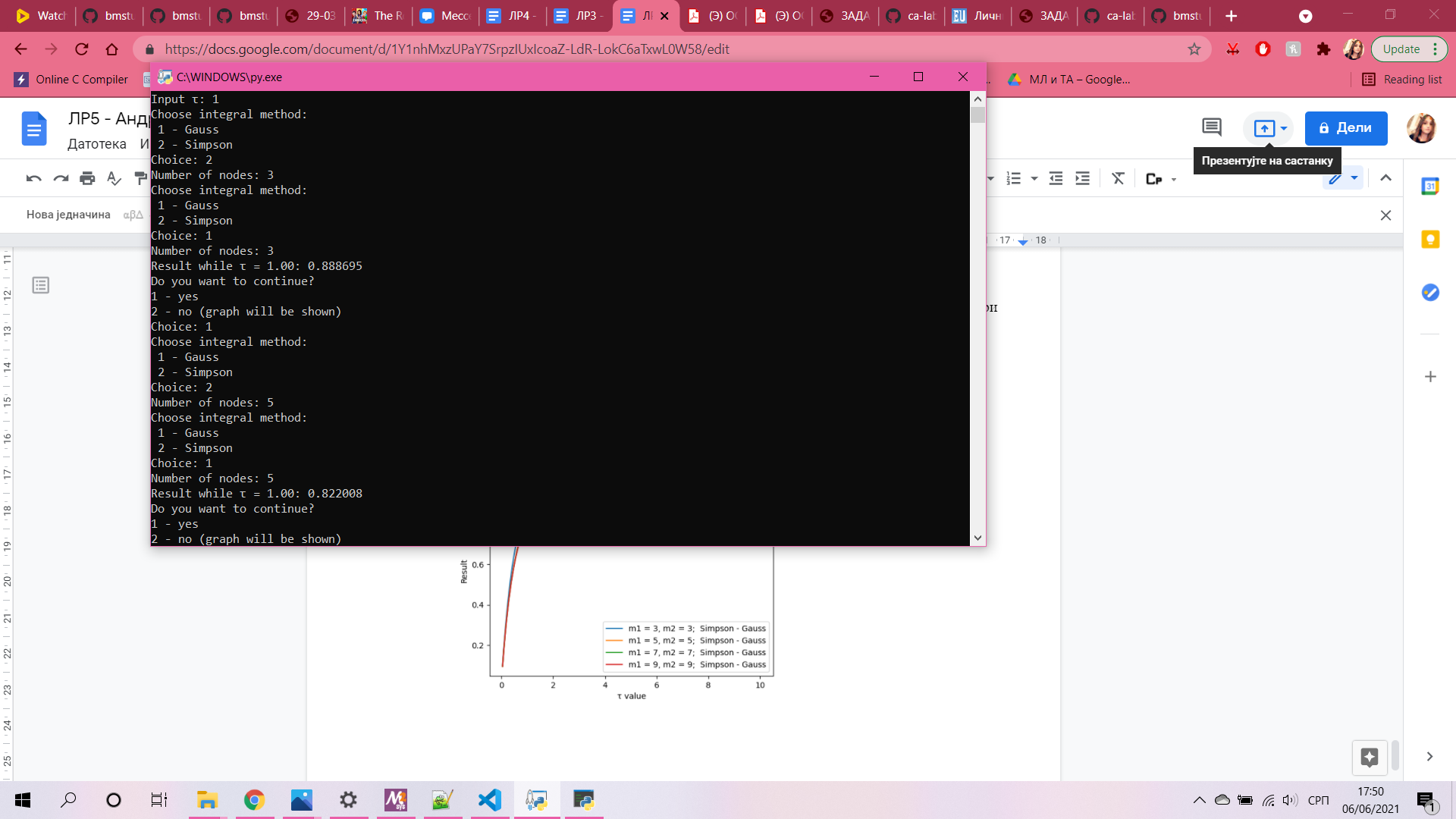
Здесь видим что при нечетных m результаты сильно отличаются от четных, получаем расхождение.

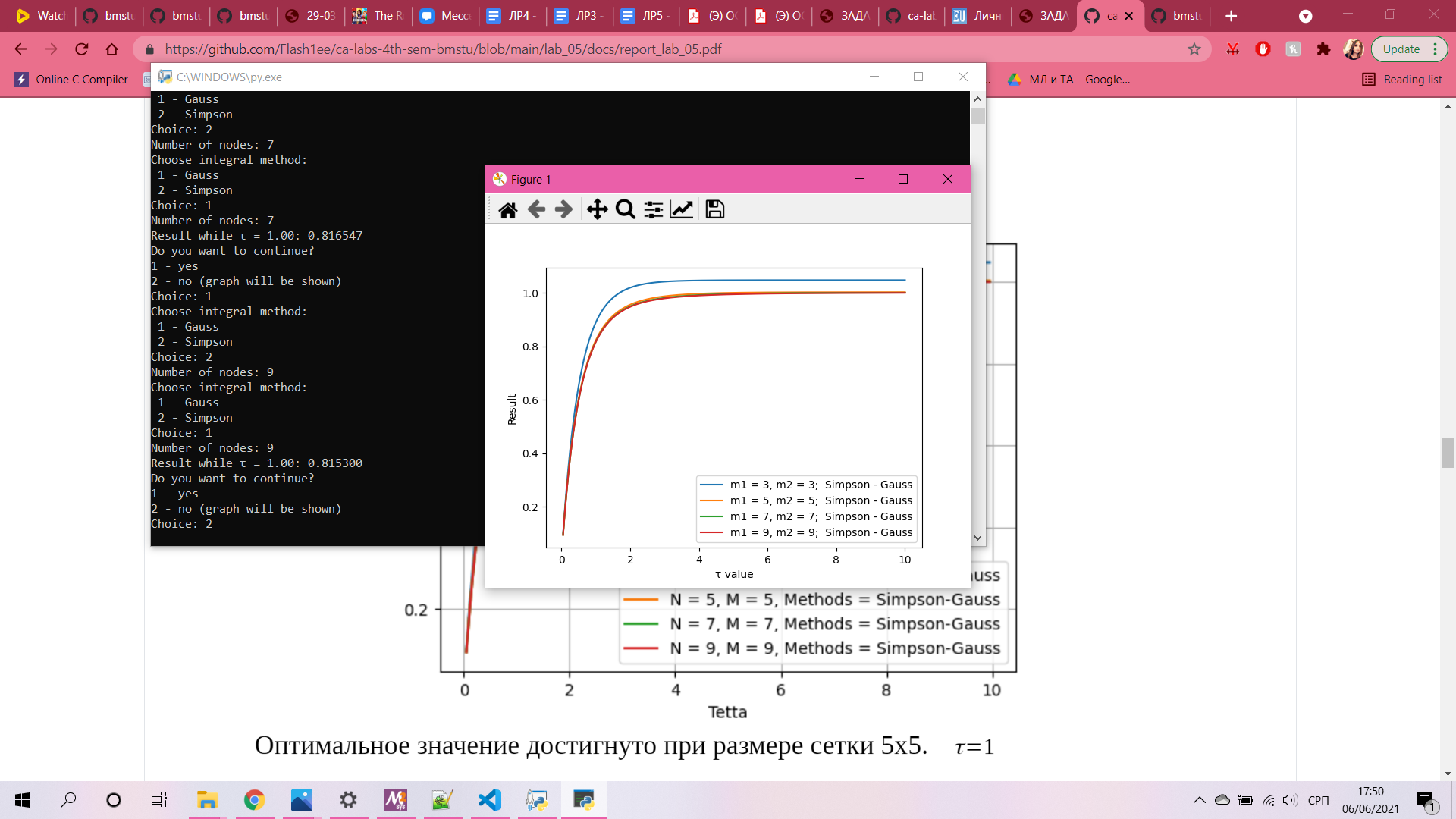
 



Здесь видно что у методы Гаусса высокая сходимость независимо от четности.

1. Построить график зависимости () в диапазоне изменения =0.05-10. Указать при каком количестве узлов получены результаты.





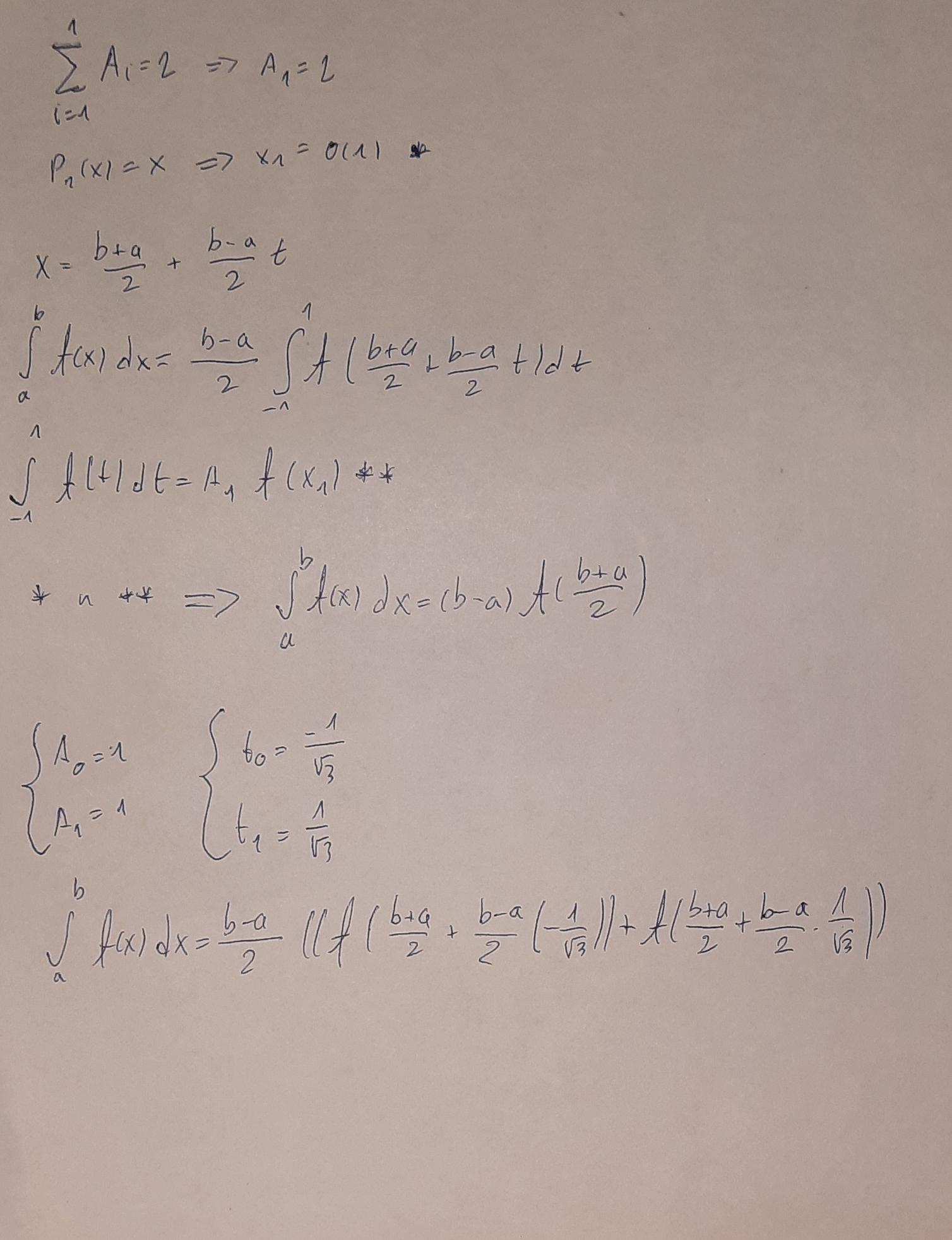
Оптимальное значение при 5х5

Вопросы при защите лабораторной работы

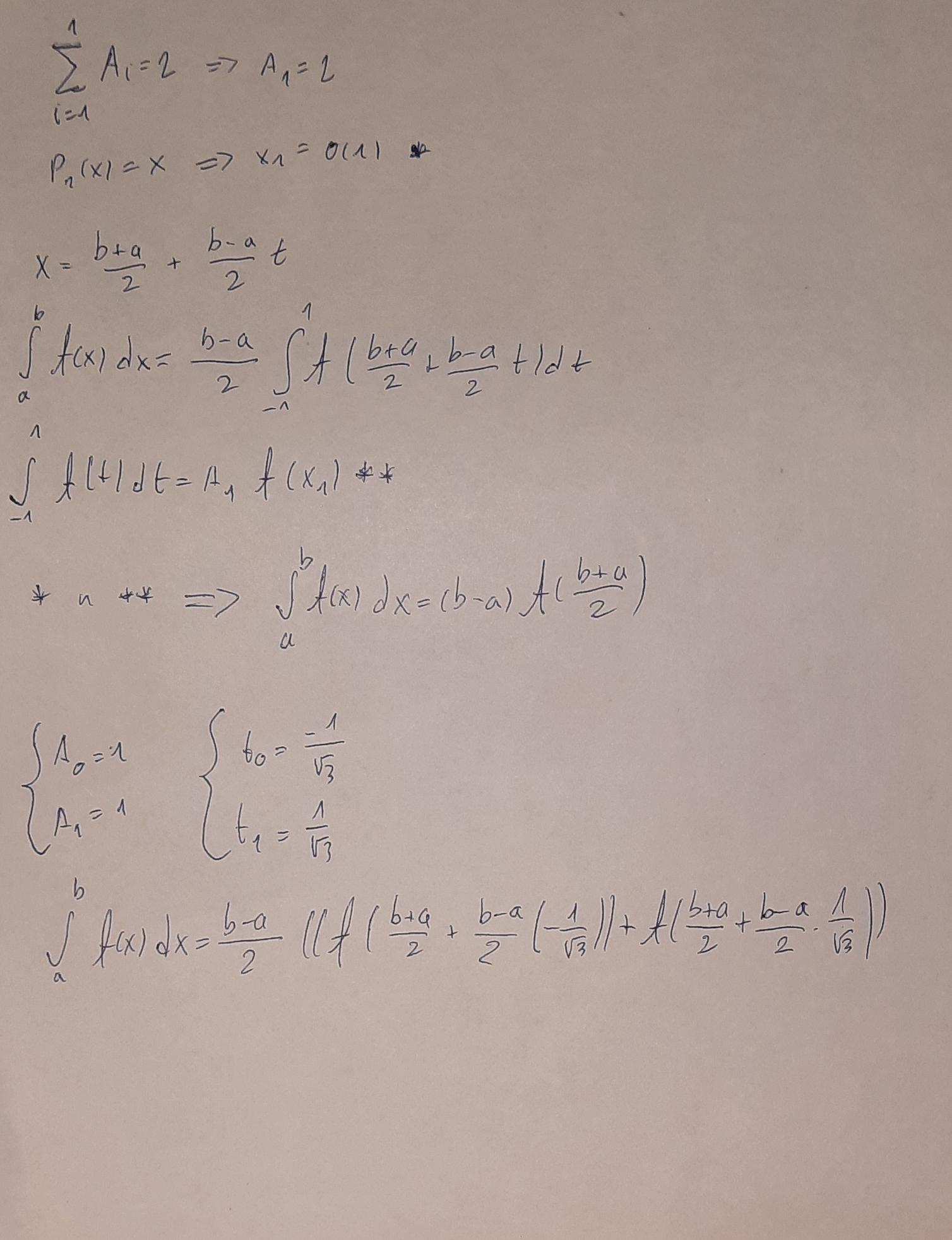
1. *В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается.*

Теоретический порядок квадратурных формул численного интегрирования не достигается если у подынтегральной функции нет соответствующих производных.

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



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



1. *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.*

