**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**

**Тема** Построение и программная реализация алгоритмов численного интегрирования.

**Студент** Пронин А.С.

**Группа** ИУ7-42Б

**Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Преподаватель** Градов В.М.

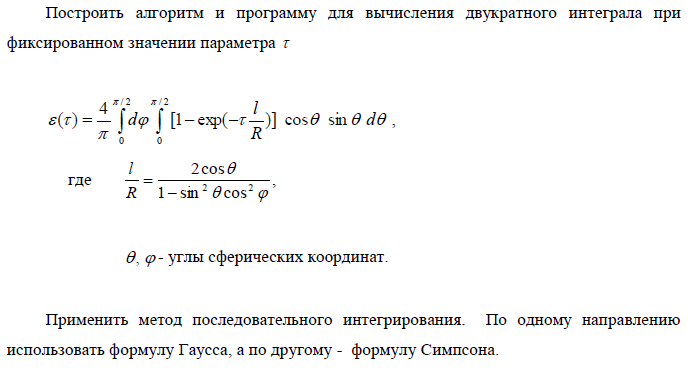
Москва.

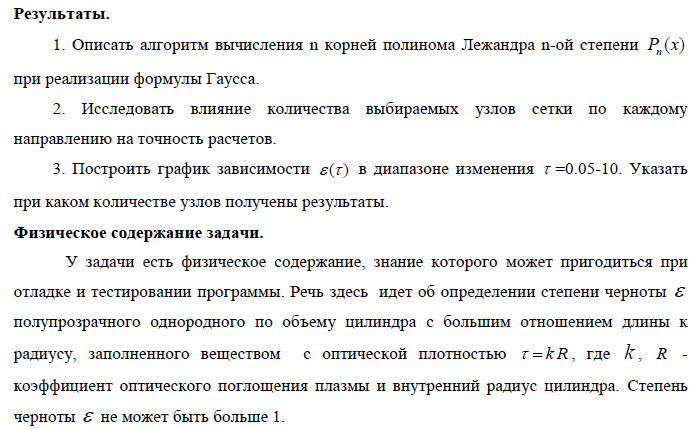
2021 г

**Цель работы:** Получение навыков построения алгоритма вычисления двукратного

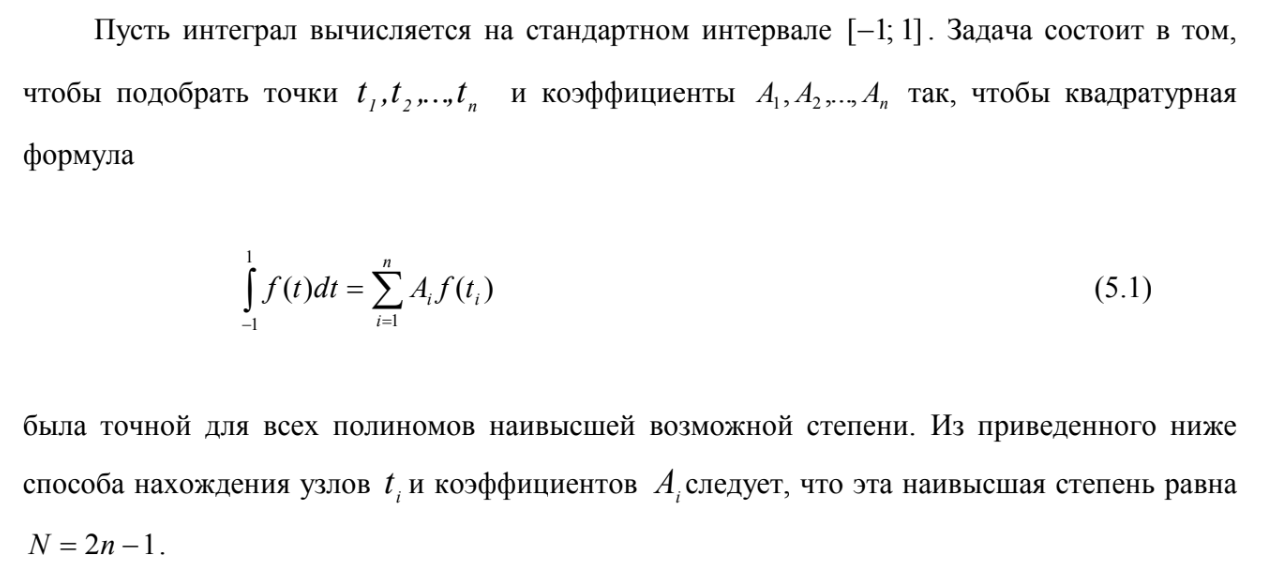
интеграла с использованием квадратурных формул Гаусса и Симпсона.

# ****1. Техническое задание****

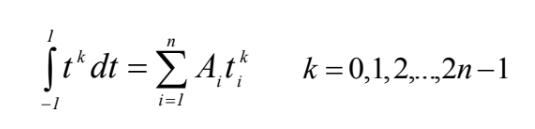




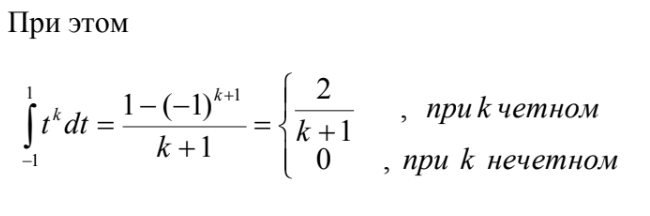
# 2. Алгоритм

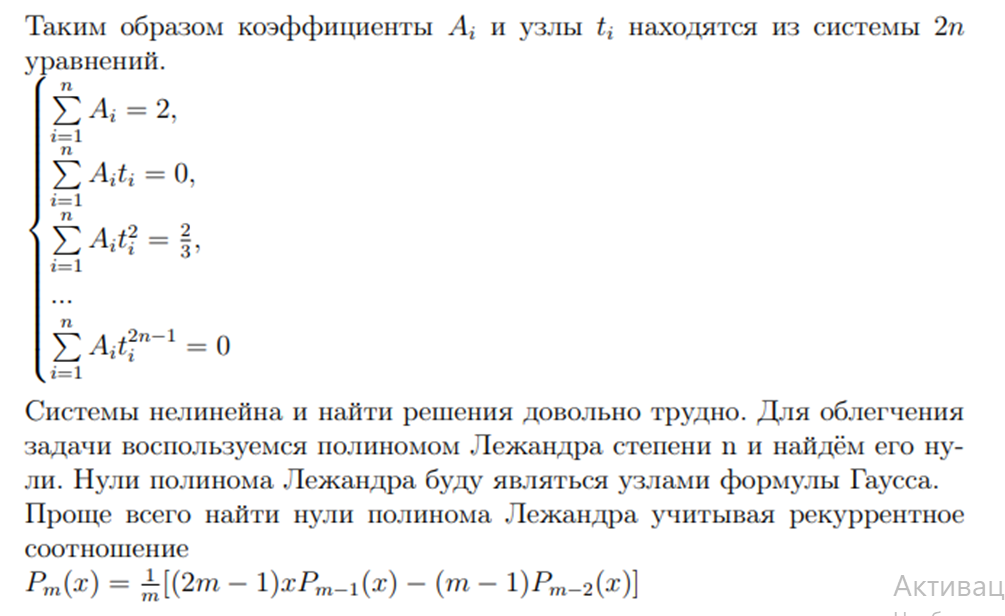


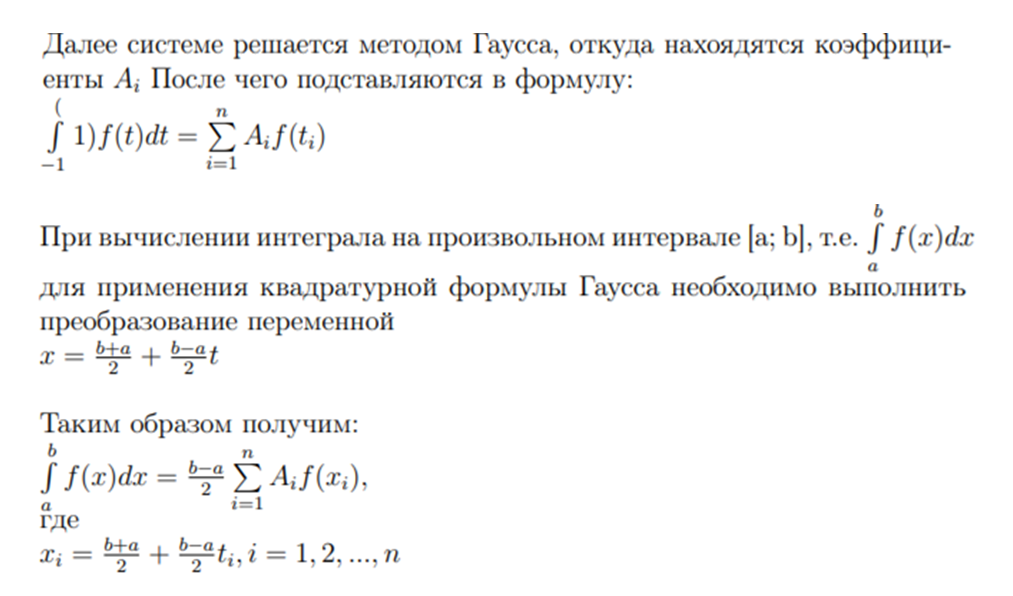
Полагая, согласно (5.1):

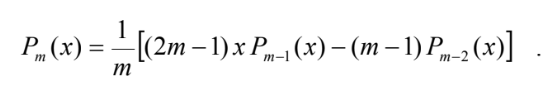


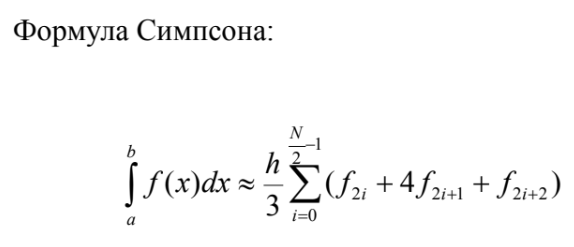
Данная система даёт *2n* соотношений для определения *2n* неизвестныхи .





  
  
Для нахождения корней полинома Лежандра находится такой отрезок, что на его концах функция принимала значения разных знаков. После этого происходит бинарный поиск точки, значение в которой, с учётом погрешности равно 0. Шаг основан на рекуррентном соотношении:





# ****3. Исходные данные****

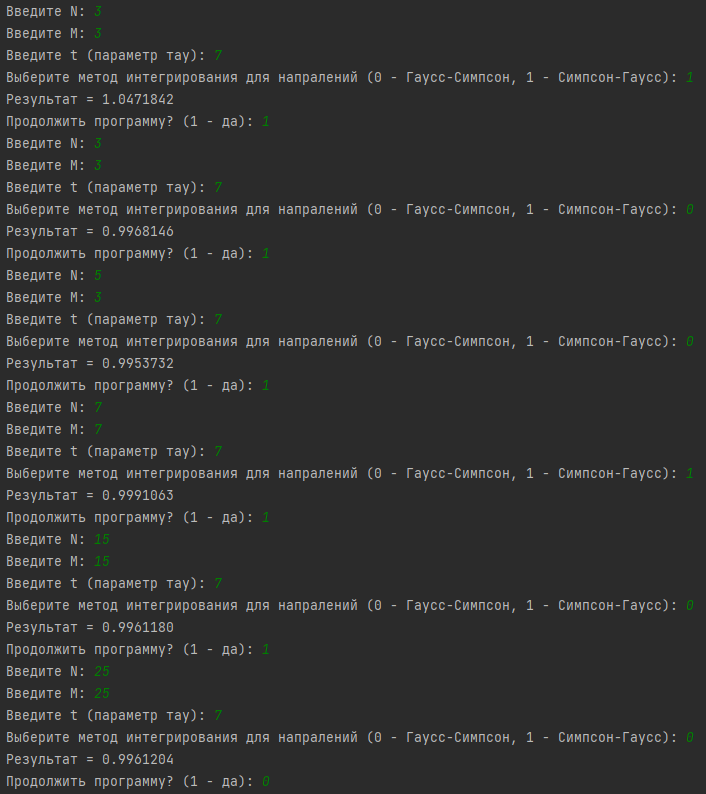
Пользователь вводит кол-ва узлов N, M, значение параметра t, и выбирает методы для интегрирования.

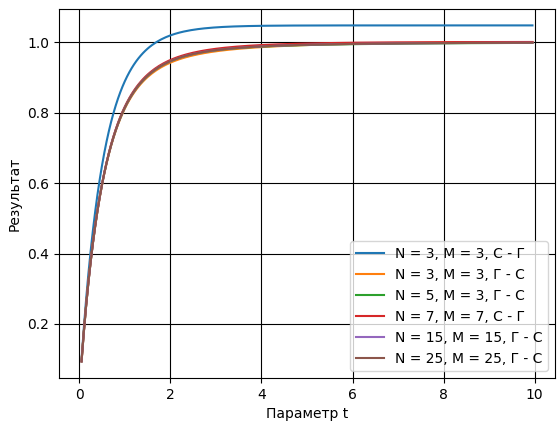
# 4. Код программы

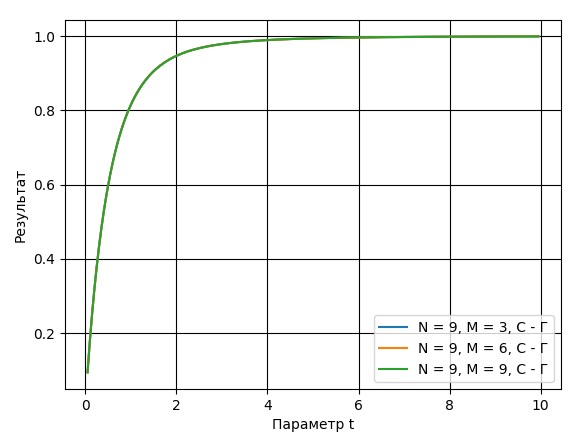
from math import cos, sin, exp, pi  
from scipy.special import roots\_legendre  
from typing import Callable as Call  
import matplotlib.pyplot as plt  
  
class Integral(object):  
 def \_\_init\_\_(self, lm: list[list[float]], n: list[int], fn: list[int]):  
 self.lm = lm  
 self.n = n  
 self.f1 = Integral.simpson if (fn[0]) else Integral.gauss  
 self.f2 = Integral.simpson if (fn[1]) else Integral.gauss  
  
 def \_\_call\_\_(self, p: float) -> float:  
 f = Integral.\_\_integrated(p)  
  
 inner = lambda x: self.f2(  
 lambda val1: f(x, val1),  
 self.lm[1][0],  
 self.lm[1][1],  
 self.n[1])  
 integ = lambda: self.f1(  
 inner,  
 self.lm[0][0],  
 self.lm[0][1],  
 self.n[0])  
  
 return integ()  
  
 @staticmethod  
 def \_\_integrated(p: float) -> Call[[float, float], float]:  
 t = lambda x, y: 2 \* cos(x) / (1 - sin(x) \*\* 2 \* cos(y) \*\* 2)  
 return lambda x, y: 4 / pi \* (1 - exp(-p \* t(x, y))) \* cos(x) \* sin(x)  
  
 @staticmethod  
 def simpson(f: Call[[float], float], a: float, b: float,  
 n: int) -> float:  
 if n < 3 or n % 2 == 0:  
 raise Exception("Wrong n value")  
  
 h = (b - a) / (n - 1.0)  
 x = a  
 res = 0.0  
  
 for i in range((n - 1) // 2):  
 res += f(x) + 4 \* f(x + h) + f(x + 2 \* h)  
 x += 2 \* h  
  
 return res \* h / 3  
  
 @staticmethod  
 def gauss(f: Call[[float], float], a: float, b: float,  
 n: int) -> float:  
 def p2v(p: float, c: float, d: float) -> float:  
 return (d + c) / 2 + (d - c) \* p / 2  
  
 x, w = roots\_legendre(n)  
 return sum([(b - a) / 2 \* w[i] \* f(p2v(x[i], a, b)) for i in range(n)])

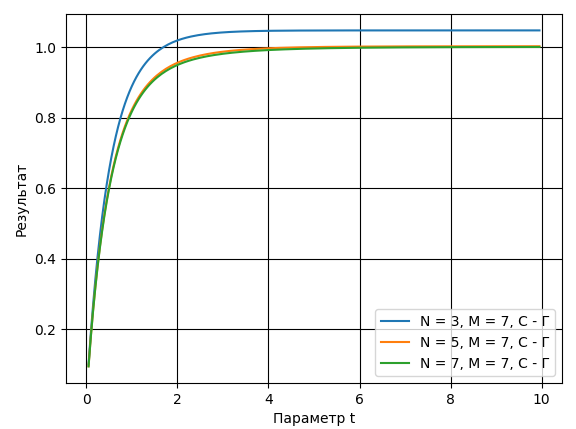
def plot(fs, sc, n\_mas, m\_mas, first\_methods, second\_methods):  
 plt.clf()  
  
 plt.xlabel("Параметр t")  
 plt.ylabel("Результат")  
 plt.grid(which='minor', color='k', linestyle=':')  
 plt.grid(which='major', color='k')  
  
 for i in range(len(fs)):  
 x, y = [], []  
 j = sc[0]  
 while j < sc[2]:  
 x.append(j)  
 y.append(fs[i](j))  
 j += sc[1]  
  
 m1 = "Г"  
 m2 = "С"  
 if (first\_methods[i] == 1):  
 m1 = "С"  
 m2 = "Г"  
 lbl = "N = {:}, M = {:}, ".format(n\_mas[i], m\_mas[i]) + m1 + " - " + m2  
 plt.plot(x, y, label = lbl)  
  
 plt.legend()  
 plt.savefig('rez.png')  
 plt.show()  
  
def main():  
 sc = [0.05, 0.05, 10.0]  
  
 n\_mas, m\_mas = [], []  
 first\_methods, second\_methods = [], []  
 integrals = []  
  
 end = '1'  
 while end == '1':  
 n\_mas.append(int(input("Введите N: ")))  
 m\_mas.append(int(input("Введите M: ")))  
  
 p = float(input("Введите t (параметр тау): "))  
  
 first\_methods.append(int(input("Выберите метод интегрирования для напралений (0 - Гаусс-Симпсон, 1 - Симпсон-Гаусс): ")))  
  
 if (first\_methods[0] == 1):  
 second\_methods.append(0)  
 else:  
 second\_methods.append(1)  
  
 lm = [[0, pi / 2], [0, pi / 2]]  
  
 integrals.append(Integral(lm, [n\_mas[-1], m\_mas[-1]], [first\_methods[-1], second\_methods[-1]]))  
  
 print("Результат = {:.2f}".format(p))  
  
 end = input("Продолжить программу? (1 - да): ")  
  
 plot(integrals, sc, n\_mas, m\_mas, first\_methods, second\_methods)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

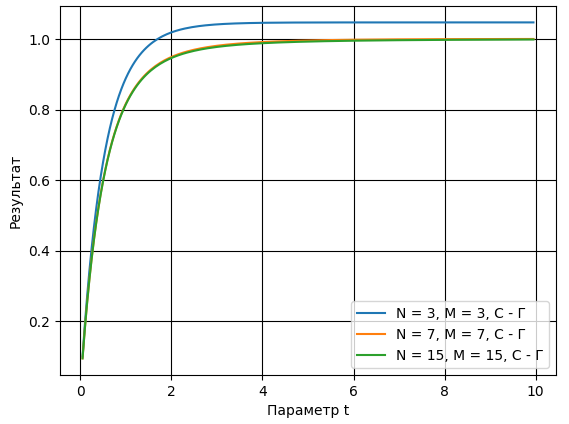
# 5. Результаты работы

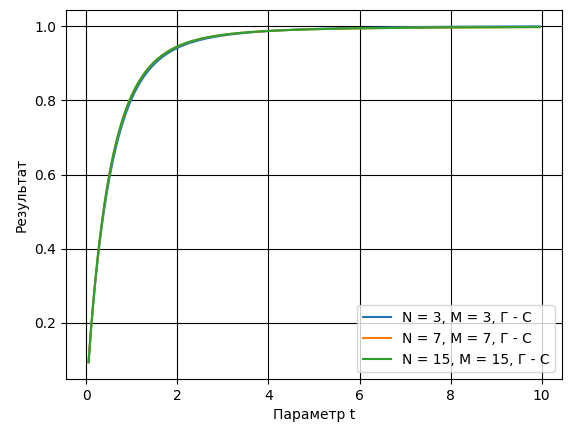












# 6. Вопросы при защите лабораторной работы

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

Заданный порядок точности всегда будет достигнут, если подынтегральная функция имеет необходимые производные.

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

=> (при t=0 для полинома Лежандра первой степени) =>

=>

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

т.к. для полинома Лежандра второй степени :

=>

***4. Получить обобщенную кубатурную формулу, на основе методе трапеций, с***

***тремя узлами на каждом направлении.***

*Где*