МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра прикладной математики и информатики

**Лабораторная работа № 0**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе

по дисциплине

Численные методы

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Куркина О.Е.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Блинов А.С.

(подпись) (фамилия, и.,о.)

23-ПМ-1

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2025

**Оглавление**

[Постановка задачи 3](#_Toc208959702)

[Теоретическая справка 4](#_Toc208959703)

[Реализация 6](#_Toc208959704)

[Результаты 9](#_Toc208959705)

[Заключение 13](#_Toc208959706)

[Список использованных источников 15](#_Toc208959707)

[Приложение 16](#_Toc208959708)

# **Постановка задачи**

Построить частичные суммы

ряда Тейлора для функции

на отрезке [0, 2π].

В фиксированной точке указанного отрезка исследовать, как изменяется разность

в зависимости от номера n частичной суммы.

Для фиксированного значения показать, как изменяется величина

в зависимости от номера n частичной суммы.

Как масштабный параметр k влияет на указанные величины?

Выполнить графические иллюстрации для демонстрации сходимости ряда к функции.

# **Теоретическая справка**

Ряд Тейлора для функции , бесконечно дифференцируемой в точке , представляет собой степенной ряд вида:

где  обозначает -ю производную функции  в точке .

Частичная сумма  ряда Тейлора порядка  определяется как:

Для функции  ряд Тейлора в точке  (ряд Маклорена) имеет вид:

Частичная сумма порядка  для этой функции:

Остаточный член характеризует погрешность приближения функции частичной суммой:

Остаточный член для этой функции:

Ряд Тейлора для функции синуса сходится к самой функции на всей числовой прямой:

Масштабный параметр влияет на:

* Скорость изменения функции (частоту колебаний)
* Радиус сходимости (для синуса ряд сходится при всех )
* Скорость сходимости ряда Тейлора
* Величину погрешности аппроксимации

При увеличении  требуется больше членов ряда для достижения той же точности аппроксимации на заданном интервале.

Для заданного  интервал , на котором гарантируется выполнение условия:

определяется из оценки остаточного члена и зависит от:

* Номера частичной суммы
* Значения параметра
* Требуемой точности

# **Реализация**

Для выполнения лабораторной работы и визуализации результатов было разработано приложение на языке C++ с использованием фреймворка Qt и библиотеки для построения графиков QCustomPlot.

Выбор инструментов:

* Qt был выбран как кроссплатформенный фреймворк, предоставляющий все необходимые компоненты для создания пользовательского интерфейса (кнопки, поля ввода, менеджеры компоновки).
* Библиотека QCustomPlot была выбрана для построения графиков благодаря своей простоте интеграции с Qt, высокой производительности и широким возможностям для кастомизации, что идеально подходит для задач визуализации математических функций.

Ход работы:

1. Проектирование интерфейса:

* Было создано главное окно приложения (MainWindow).
* В левой части окна размещена панель управления (QWidget), содержащая:
* Счетчик QDoubleSpinBox для задания масштабного параметра k.
* Счетчик QSpinBox для выбора максимального номра частичной суммы N (диапазон от 1 до 10).
* Кнопку QPushButton для запуска расчета.
* Поле для вывода результатов.
* Основную часть окна занимает виджет для построения графиков (QCustomPlot).

1. Реализация математического аппарата:

* Реализована функция double factorial(int n), вычисляющая факториал числа. Данная функция используется для вычисления знаменателя в формуле ряда Тейлора.
* Реализована ключевая функция double taylor(double x, double k, int nMax), которая вычисляет значение частичной суммы ряда Тейлора для функции ) в точке  до порядка nMax включительно. Формула реализации соответствует теоретической:

1. Визуализация и логика приложения:

* При нажатии на кнопку "Вычислить" вызывается функция calculate().
* В этой функции сначала очищаются все предыдущие графики.
* Затем вычисляются значения точной функции  на интервале  в 1000 точках. График этой функции строится красной сплошной линией.
* Для каждого порядка аппроксимации  от 1 до  выполняется цикл:
* Для каждой точки x на интервале вычисляется значение частичной суммы и абсолютная погрешность .
* Точки, в которых погрешность не превышает заданное значение , сохраняются в один массив (валидные точки), а точки с погрешностью больше  — в другой (невалидные точки).
* График частичной суммы отображается на графике двумя частями:
* Валидная часть (где ошибка ) рисуется сплошной линой уникального цвета и подписывается в легенде.
* Невалидная часть (где ошибка ) рисуется пунктирной линой того же цвета, но не добавляется в легенду для избежания загромождения. Это наглядно показывает, на каком отрезке гарантируется требуемая точность.
* В поле для вывода результатов для каждого n выводится интервал , на котором достигнута точность .
* В конце работы функция устанавливает удобный диапазон осей и обновляет график.

# **Результаты**

В ходе выполнения лабораторной работы был проведён численный эксперимент по исследованию сходимости ряда Тейлора для функции  на интервале . Были получены следующие результаты.

1. Визуальный анализ сходимости частичных сумм

На серии графиков (Рисунок 1 и Рисунок 2) наглядно продемонстрированы граф функции  и приближение частичных сумм  к функции  с увеличением порядка аппроксимации .

* При  частичная сумма  представляет собой прямую линию, которая удовлетворительно приближает функцию  лишь в очень малой окрестности точки .
* При  (что соответствует учёту членов ряда до 3-й степени) частичная сумма  уже принимает форму, характерную для синусоиды, на значительной части интервала.
* С дальнейшим увеличением  (до ) частичная сумма практически точно совпадает с функцией на всём интервале  для . Чем выше частота функции (больше ), тем больше членов ряда требуется для достижения схожего уровня точности на всём интервале.

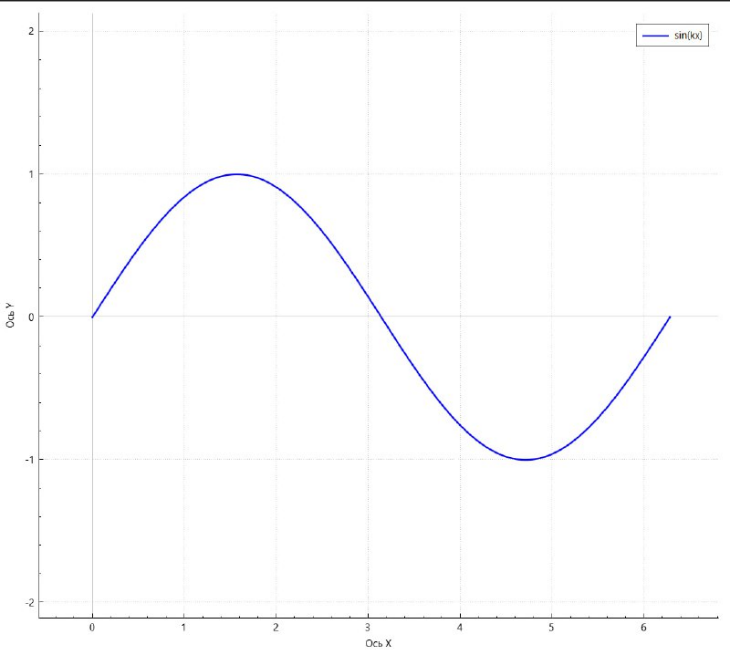


Рисунок 1 – Граф функции

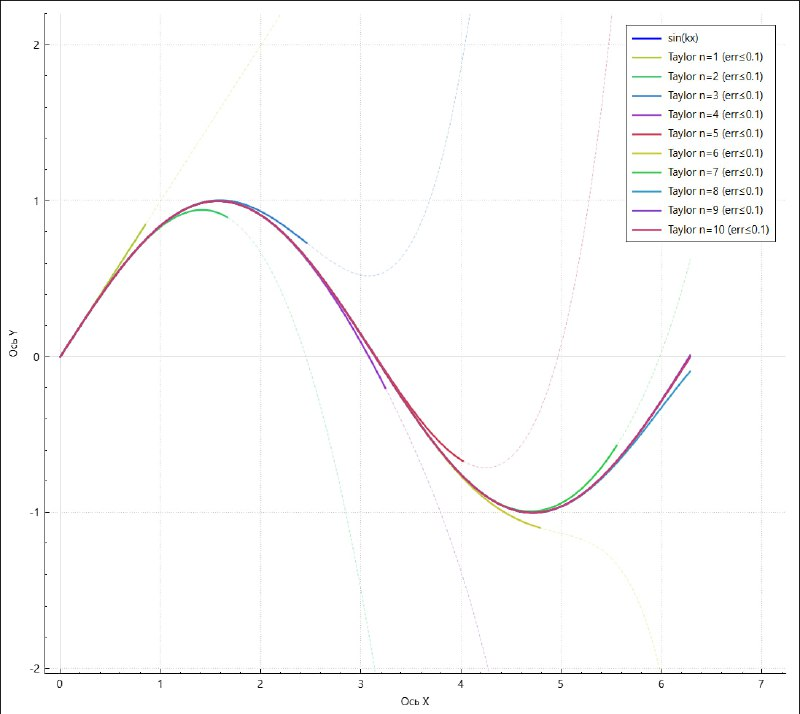


Рисунок 2 – Визуализация частичных сумм ряда Тейлора для функции при

1. Исследование погрешности в фиксированной точке

Для фиксированной точки   была исследована зависимость абсолютной погрешности   от номера  частичной суммы.

Было установлено, что с ростом n погрешность  монотонно убывает, стремясь к нулю. Однако, скорость этого убывания существенно зависит от значения   и параметра . Для точек, удалённых от центра разложения (), и для больших значений  сходимость в начальный момент (при малых ) может быть очень медленной. Только начиная с некоторого номера , погрешность начинает уменьшаться заметными темпами.

На рисунке 3 представлена зависимость погрешности.

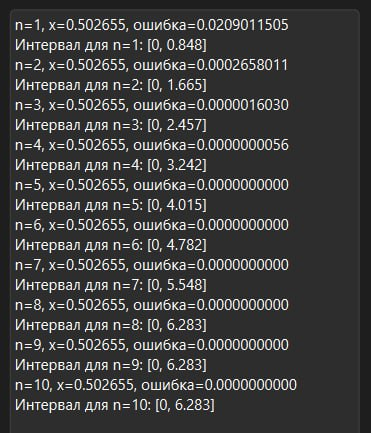


Рисунок 3 – Зависимость погрешности от для

3. Анализ интервала гарантированной точности

Было исследовано, как изменяется величина  (правая граница интервала, на котором погрешность аппроксимации не превышает ) в зависимости от номера n частичной суммы.

Результаты показывают, что с увеличением n интервал , на котором выполняется условие , монотонно расширяется, стремясь к cover всему интервалу . Это полностью согласуется с теоретическими предпосылками о сходимости ряда Тейлора для синуса на всей числовой оси.

4. Влияние масштабного параметра

Параметр  оказывает существенное влияние на процесс аппроксимации:

* Скорость сходимости: Для функции с большей частотой () ряд Тейлора сходится медленнее. Для достижения той же погрешности  в фиксированной точке x₀ или для охвата той же длины интервала  с заданной точностью требуется учитывать больше членов ряда (большее значение ) по сравнению с функцией  ().
* Интервал точности: При одинаковом  значение  для функции  с  будет всегда меньше, чем  для функции . Это объясняется тем, что рост  эквивалентен «сжатию» аргумента, из-за члена  в ряде Тейлора, который растёт быстрее для больших , требуя более высоких степеней для его компенсации.

# **Заключение**

В ходе выполнения лабораторной работы были успешно достигнуты все поставленные цели. Были построены и проанализированы частичные суммы ряда Тейлора для функции на интервале , исследована зависимость погрешности аппроксимации от количества членов ряда и влияния масштабного параметра .

Основные выводы, сделанные в результате работы:

1. Подтверждение сходимости ряда: Экспериментально подтверждено фундаментальное теоретическое положение о том, что ряд Тейлора функции действительно сходится к самой функции на всей числовой прямой. Визуально это выражается в том, что с увеличением порядка аппроксимации график частичной суммы практически сливается с графиком исходной функции.
2. Зависимость точности от количества членов ряда: Установлена прямая зависимость между точностью аппроксимации и номером частичной суммы . Погрешность в любой фиксированной точке монотонно убывает с ростом , стремясь к нулю. Это подтверждает корректность реализации алгоритма вычисления частичной суммы.
3. Расширение интервала гарантированной точности: Показано, что интервал , на котором погрешность не превышает заданного значения, закономерно расширяется с увеличением .
4. Ключевое влияние параметра : Наиболее значимым результатом является исследование влияния масштабного параметра . Установлено, что увеличение (частоты функции) приводит к существенному замедлению скорости сходимости ряда. Для достижения той же точности на том же интервале функции с большим *k* требуется учитывать значительно больше членов ряда Тейлора. Это объясняется тем, что рост увеличивает значение старших производных в точке разложения, что, в свою очередь, увеличивает вклад старших членов ряда, без учёта которых погрешность остаётся высокой.

Таким образом, работа наглядно продемонстрировала как теоретические принципы разложения функций в степенные ряды, так и их практическую реализацию. Полученные результаты полностью соответствуют теоретическим ожиданиям и подтверждают, что ряд Тейлора является мощным инструментом аппроксимации, эффективность которого, однако, сильно зависит от свойств самой функции (в данном случае — от её частоты).

# **Список использованных источников**

1. Кудрявцев Л.Д. Курс математического анализа: В 3 томах. Том 2. – Москва: Дрофа, 2003.
2. Math24: Разложение функций в степенные ряды: Электронный ресурс // Math24.ru. – URL: https://math24.ru/power-series-expansion.html (дата обращения: 09.09.2025).
3. Math24: Разложение функций в степенные ряды: Электронный ресурс // Math24.ru. – URL: https://math24.ru/power-series-expansion.html (дата обращения: 10.09.2025).

# **Приложение**

1. main.cpp

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

1. mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <qcustomplot.h>

QT\_BEGIN\_NAMESPACE

namespace Ui {

class MainWindow;

}

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private:

QCustomPlot \*plotWidget;

QDoubleSpinBox \*kSpinBox;

QSpinBox \*nSpinBox;

QTextEdit \*outputText;

QTextEdit \*errorText;

QVector<double> x0Values;

QVector<QVector<double>> partialSums;

QVector<double> errorsAtX0;

QVector<double> XnValues;

double x0Fix = 0.502655;

double epsilon = 0.1;

double taylor(double x, double k,int nMax);

double factorial(int number);

void calculate();

};

#endif // MAINWINDOW\_H

1. mainwindow.cpp

#include "mainwindow.h"

#include <QGridLayout>

#include <QDoubleSpinBox>

#include <QPushButton>

#include <QDebug>

#include <cmath>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

{

setWindowTitle("Lab 0");

setMinimumSize(1200, 800);

QWidget \*central = new QWidget(this);

setCentralWidget(central);

QGridLayout \*gridLayout = new QGridLayout(central);

QWidget \*controlPanel = new QWidget();

controlPanel->setFixedWidth(300);

QVBoxLayout \*panelLayout = new QVBoxLayout(controlPanel);

kSpinBox = new QDoubleSpinBox();

kSpinBox->setRange(1.0, 10.0);

kSpinBox->setValue(1.0);

kSpinBox->setSingleStep(1.0);

nSpinBox = new QSpinBox();

nSpinBox->setRange(0, 10);

nSpinBox->setValue(0);

outputText = new QTextEdit();

outputText->setReadOnly(true);

outputText->setFixedHeight(400);

QPushButton \*plotButton = new QPushButton("Построить график");

plotButton->setFixedHeight(40);

panelLayout->addWidget(new QLabel("Коэффициент k:"));

panelLayout->addWidget(kSpinBox);

panelLayout->addWidget(new QLabel("Кол-во членов ряда n:"));

panelLayout->addWidget(nSpinBox);

panelLayout->addSpacing(20);

panelLayout->addWidget(plotButton);

panelLayout->addWidget(new QLabel("Результаты:"));

panelLayout->addWidget(outputText);

panelLayout->addStretch();

plotWidget = new QCustomPlot();

plotWidget->xAxis->setLabel("Ось X");

plotWidget->yAxis->setLabel("Ось Y");

plotWidget->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom | QCP::iSelectPlottables);

plotWidget->legend->setVisible(true);

gridLayout->addWidget(controlPanel, 0, 0, 1, 1);

gridLayout->addWidget(plotWidget, 0, 1, 1, 3);

gridLayout->setColumnStretch(1, 1);

connect(plotButton, &QPushButton::clicked, this, &MainWindow::calculate);

}

MainWindow::~MainWindow()

{

}

double MainWindow::factorial(int n)

{

if (n < 0) {

qCritical() << "Factorial error: negative input";

exit(1);

}

double fact = 1.0;

while (n > 1) {

fact \*= n--;

}

return fact;

}

double MainWindow::taylor(double x, double k, int nMax)

{

double sum = 0.0;

for (int i = 0; i < nMax; ++i) {

int power = 2 \* i + 1;

double term = std::pow(-1.0, i) \* std::pow(k \* x, power) / factorial(power);

sum += term;

}

return sum;

}

void MainWindow::calculate()

{

outputText->clear();

plotWidget->clearGraphs();

double k = kSpinBox->value();

int nMax = nSpinBox->value();

const int numPoints = 1000;

QVector<double> xData(numPoints + 1), exactData(numPoints + 1);

for (int i = 0; i <= numPoints; ++i) {

xData[i] = 2 \* M\_PI \* i / numPoints;

exactData[i] = std::sin(k \* xData[i]);

}

QCPGraph \*exactPlot = plotWidget->addGraph();

exactPlot->setData(xData, exactData);

exactPlot->setPen(QPen(Qt::blue, 2));

exactPlot->setName("sin(kx)");

for (int n = 1; n <= nMax; ++n) {

QVector<double> validPointsX, validPointsY, invalidPointsX, invalidPointsY;

for (int i = 0; i < xData.size(); ++i) {

double x = xData[i];

double exact = exactData[i];

double taylorRes = taylor(x, k, n);

double error = std::abs(exact - taylorRes);

if (error <= epsilon) {

validPointsX.append(x);

validPointsY.append(taylorRes);

if (x <= 0.502655 && x >= 0.502654) {

outputText->append(QString("n=%1, x=%2, ошибка=%3")

.arg(n)

.arg(x, 0, 'f', 6)

.arg(error, 0, 'f', 10));

}

} else {

invalidPointsX.append(x);

invalidPointsY.append(taylorRes);

}

}

if (!validPointsX.isEmpty()) {

outputText->append(QString("Интервал для n=%1: [0, %2]")

.arg(n)

.arg(validPointsX.last(), 0, 'f', 3));

QCPGraph \*validPlot = plotWidget->addGraph();

validPlot->setData(validPointsX, validPointsY);

QColor color = QColor::fromHsv((n \* 70) % 360, 180, 200);

validPlot->setPen(QPen(color, 2));

validPlot->setName(QString("Taylor n=%1 (err≤%2)").arg(n).arg(epsilon));

}

if (!invalidPointsX.isEmpty()) {

QCPGraph \*invalidPlot = plotWidget->addGraph();

invalidPlot->setData(invalidPointsX, invalidPointsY);

QColor color = QColor::fromHsv((n \* 70) % 360, 180, 200, 100);

invalidPlot->setPen(QPen(color, 1, Qt::DashLine));

invalidPlot->removeFromLegend();

}

}

plotWidget->xAxis->setRange(0, 2 \* M\_PI);

plotWidget->yAxis->setRange(-1.8, 1.8);

plotWidget->replot();

}