# Приложение 1

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет математики и компьютерных наук им. проф. Н.И. Червякова

Кафедра математического моделирования

Допущен к защите

«\_\_\_»\_\_\_\_\_\_ 2025 г.

Зав. кафедрой

математического моделирования

П.А. Ляхов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

|  |  |
| --- | --- |
| Руководитель практики от профильной организации:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Выполнила:  Бабичева Дарья Олеговна  студентка 2 курса, группы ПМИ-б-о-23-1  направление подготовки  01.03.02 «Прикладная математика и информатика», направленность (профиль) «Вычислительная математика и математическое моделирование»  очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
|  | Руководитель практики:  Шапошников Алексей Вячеславович, доцент, доцент кафедры математического моделирования ФМКН имени профессора Н.И. Червякова  (ФИО, звание, должность)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Отчёт защищён с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_ Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2025 г.

# Приложение 2

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет математики и компьютерных наук им. проф. Н.И. Червякова

Кафедра математического моделирования

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) Вычислительная математика и математическое моделирование

**ЗАДАНИЕ**

на учебную практику

(практику по получению профессиональных умений и опыта профессиональной деятельности)

Студент: Бабичева Дарья Олеговна

Группа: ПМИ-б-о-23-1 Форма обучения: Очная

Сроки прохождения практики: с 23 июня по 19 июля

Место прохождения практики: Кафедра математического моделирования

Задание на практику Вариант 6

Руководитель практики от СКФУ: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики от организации: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Индивидуальное задание на учебную**

**(проектно-технологическую) практику**

1. Охарактеризуйте фазу реализации проекта и приведите основные функции по управлению в этой фазе.
2. Разработать проект программы численного решения нелинейного уравнения методом хорд, с детальной разработкой модуля визуализации полученного решения.

Задание утверждено на заседании кафедры математического моделирования (протокол №\_\_ от «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.).

Дата выдачи задания: «23» Июня 2025 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_» \_\_\_\_\_\_\_\_20\_\_ г.

(подпись руководителя)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_20\_\_ г.

(подпись студента)

# Приложение 3

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**ДНЕВНИК СТУДЕНТА ПО ПРАКТИКЕ**

1. Фамилия **Бабичева**
2. Имя, Отчество **Дарья Олеговна**
3. Факультет математики и компьютерных наук им. проф. Н.И. Червякова
4. Направление подготовки 01.03.02 «Прикладная математика и информатика»
5. Направленность (профиль) «Вычислительная математика и математическое моделирование»
6. Курс **2** Группа Первая Форма обучения: **очная**
7. Место прохождения практики: Кафедра математического моделирования
8. Вид, тип практики: учебная (проектно-технологическая) практика
9. Руководитель практики от Университета:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
10. Руководитель практики от организации: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
11. Сроки практики по учебному плану:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

# Календарный план прохождения практики

|  |  |  |  |
| --- | --- | --- | --- |
| Сроки (продолжительность) работ | Цех, отдел, лаборатория, рабочее место бакалавра | Виды работы бакалавра | Отчётность по выполненной работе |
| 23 июня 2025г.  (1 день) | СКФУ | Посещение установочной конференции  Прохождение инструктажа по технике безопасности | Выполнено |
| 24 июня – 26 июня (2 дня) | СКФУ | Изучение нормативно-технической документации | Выполнено |
| 24 июня – 26 июня (2 дня) | СКФУ | Знакомство с оборудованием, изучение информационной структуры кафедр | Выполнено |
| 27 июня – 5 июля (9 дней) | СКФУ | Знакомство с сетевыми решениями, применяемыми на кафедрах | Выполнено |
| 5 июля – 15 июля (11 дней ) | СКФУ | Выполнение индивидуального задания | Выполнено |
| 15 июля (1 день) | СКФУ | Оформление дневника и отчёта о прохождении производственной практики (практики по получению профессиональных умений и опыта профессиональной деятельности) | Выполнено |
| 16 июля (1 день) | СКФУ | Получение отзыва руководителя практики от предприятия | Выполнено |
| 16 июля (1 июля) | СКФУ | Доклад на итоговой конференции | Выполнено |

Задание утверждено на заседании кафедры математического моделирования (протокол №\_\_\_ от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.).

Дата выдачи задания: «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_ г.

Руководители практики:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

# Содержание

1. [Номер 1](#_НОМЕР_1)
   1. [Определение фазы](#_Определение_фазы_реализации)
   2. [Связь с другими фазами жизненного циклами](#_Связь_с_другими)
      1. [Жизненный цикл](#_Жизненный_цикл_включает)
      2. [Важно:](#_Важно:)
   3. [Этапы фазы реализации (детализация)](#_Этапы_фазы_реализации)
      1. [Запуск сводного плана работ](#_Запуск_сводного_плана)
      2. [Активация системы управления и контроля](#_Активация_системы_управления)
      3. [Организация коммуникаций](#_Организация_коммуникаций)
      4. [Внедрение системы мотивации](#_Внедрение_системы_мотивации)
      5. [Управление МТО (материально-техническим обеспечением)](#_Управление_МТО_(материально-техниче)
      6. [Решение возникающих проблем](#_Решение_возникающих_проблем)
      7. [Подтверждение завершения работ](#_Подтверждение_завершения_работ)
   4. [Участники и их обязанности](#_Участники_и_их)
   5. [Функции управления в фазе реализации](#_5._Функции_управления)
      1. [Управление исполнением работ](#_Управление_исполнением_работ)
      2. [Управление ресурсами](#_Управление_ресурсами)
      3. [Мониторинг и контроль](#_Мониторинг_и_контроль)
      4. [Управление рисками](#_Управление_рисками)
      5. [Управление коммуникациями](#_Управление_коммуникациями)
      6. [Управление качеством](#_Управление_качеством)
      7. [Управление изменениями](#_Управление_изменениями)
2. [Номер 2](#_Номер_2)
   1. [Теоретическая часть по решению практической задачи](#_Теоретическая_часть_по)
      1. [Нелинейное уравнение и метод хорд](#_Нелинейное_уравнение_и)
      2. [Метод хорд](#_Метод_хорд)
      3. [Алгоритм метода](#_Алгоритм_метода)
   2. [Блок-схема программы](#_Блок_–_схема)
   3. [Листинг программы (описание функций и структуры кода)](#_Листинг_программы_(описание)
      1. [Основные компоненты программы](#_Основные_компоненты_программы)
      2. [Функциональные методы](#_Функциональные_методы)
      3. [Вспомогательные методы](#_Вспомогательные_методы)
      4. [Особенности программы](#_Особенности_программы)
      5. [Используемые библиотеки](#_Используемые_библиотеки)
   4. [Программа](#_Программа)
3. Список литературы

# НОМЕР 1

**Охарактеризуйте фазу реализации проекта и приведите основные функции по управлению в этой фазе.**

## ****Определение фазы реализации****

**Реализация** — этап жизненного цикла проекта, на котором **планы трансформируются в действия**. Это период максимальной концентрации ресурсов (финансовых, человеческих, материальных) для выполнения задач, определенных в сводном плане.

* **Ключевая характеристика**: Пик затрат и усилий команды (до 70% бюджета и трудовых ресурсов).
* **Цель**: Достижение запланированных результатов в рамках установленных ограничений (сроки, бюджет, качество).
* **Риски**: Высокая вероятность отклонений из-за внешних факторов (срыв поставок, изменения рынка) или внутренних ошибок.

## ****Связь с другими фазами жизненного цикла****

### Жизненный цикл включает 4 взаимосвязанные фазы:

|  |  |  |
| --- | --- | --- |
| **Фаза** | **Содержание** | **Переход к реализации** |
| **Концепция** | Инициация идеи, предварительный анализ, утверждение концепции. | Получение формального разрешения на детальную разработку. |
| **Разработка** | Формирование команды, детальное планирование, утверждение бюджета и графика. | Передача утвержденного сводного плана, ресурсов и полномочий менеджеру проекта. |
| **Реализация** | **Выполнение работ, контроль, решение проблем.** | Подтверждение готовности результатов для передачи на завершение. |
| **Завершение** | Сдача проекта, оценка эффективности, подготовка отчетов. | Закрытие контрактов, роспуск команды. |

### Важно:

Реализация зависит от качества выходов фазы разработки (детальный план, резервы).

Проблемы на этапе реализации могут потребовать возврата к разработке для корректировки плана.

## Этапы фазы реализации (детализация)

Каждый этап включает конкретные действия и инструменты управления:

### Запуск сводного плана работ

* **Содержание**: Начало выполнения задач согласно календарному графику.
* **Действия**:
  + Раздача заданий команде с указанием сроков и KPI.
  + Запуск систем учета времени (например, Toggl, Jira).
* **Риски**: Срыв сроков старта из-за недоступности ресурсов.
* **Инструменты**: Диаграмма Ганта, чек-листы контроля готовности.

### Активация системы управления и контроля

* **Содержание**: Мониторинг прогресса в режиме реального времени.
* **Действия**:
  + Ежедневные стендапы для оперативного выявления проблем.
  + Сравнение фактических показателей с плановыми (бюджет, сроки).
* **Риски**: Формальный контроль без анализа причин отклонений.
* **Инструменты**: Dashboards в Power BI ( изображение данных в графиках и диаграмах), отчеты EVM (Earned Value Management).

### Организация коммуникаций

* **Содержание**: Обеспечение прозрачности информации между участниками.
* **Действия**:
  + Создание регламента совещаний (еженедельные планерки, экстренные созвоны).
  + Использование единой платформы (Slack, Microsoft Teams).
* **Риски**: Искажение информации при передаче между отделами.
* **Инструменты**: RACI-матрица для распределения ответственности.

### Внедрение системы мотивации

* **Содержание**: Стимулирование команды для поддержания продуктивности.
* **Действия**:
  + KPI-премии за соблюдение сроков.
  + Нематериальная мотивация (благодарности, рейтинги лидеров).
* **Риски**: Демотивация из-за неравной нагрузки.
* **Инструменты**: Геймификация (система баллов), pulse-опросы для оценки настроения команды.

### Управление МТО (материально-техническим обеспечением)

* **Содержание**: Гарантия бесперебойных поставок ресурсов.
* **Действия**:
  + Контроль запасов на складах, аудит поставщиков.
  + Разработка плана B на случай форс-мажора (например, альтернативный поставщик).
* **Риски**: Простой работ из-за дефицита материалов.
* **Инструменты**: ERP-системы (SAP, 1С), логистические трекеры.

### Решение возникающих проблем

* **Содержание**: Оперативное реагирование на отклонения.
* **Действия**:
  + Анализ корневых причин сбоев (метод 5 Why).
  + Корректировка плана при согласовании со стейкхолдерами.
* **Риски**: Эскалация мелких проблем в кризис.
* **Инструменты**: Бэклог рисков, матрица приоритетов Eisenhower.

### Подтверждение завершения работ

* **Содержание**: Фиксация готовности результатов к сдаче.
* **Действия**:
  + Финальный аудит качества (чек-листы, тестирование).
  + Подписание актов выполненных работ с подрядчиками.
* **Риски**: Несоответствие продукта изначальным требованиям.
* **Инструменты**: User Acceptance Testing (UAT), протоколы сдачи-приемки.

## ****Участники и их обязанности****

|  |  |  |
| --- | --- | --- |
| **Роль** | **Профессиональный профиль** | **Обязанности** |
| **Менеджер проекта** | PMP-сертифицированный специалист | Контроль triple constraints (сроки/бюджет/качество), эскалация проблем спонсору. |
| **Технические специалисты** | Инженеры, разработчики, строители | Выполнение задач в соответствии с ТЗ, оперативная отчетность о рисках. |
| **Контракторы** | Юристы, закупочные менеджеры | Соблюдение условий договоров, управление отношениями с субподрядчиками. |
| **Координаторы** | Администраторы проектов | Организация встреч, документооборот, обновление информационных панелей. |
| **Эксперты качества** | QA-инженеры, аудиторы | Тестирование результатов, соответствие стандартам (ISO, ГОСТ). |
| **Стейкхолдеры** | Инвесторы, заказчики | Утверждение изменений плана, приемка этапов. |

## 5. Функции управления в фазе реализации

### Управление исполнением работ

* **Содержание**: Преобразование планов в действия.
* **Методы**:
  + **Agile-подходы** (Scrum, Kanban) для гибкой адаптации к изменениям.
  + **Критические пути (CPM)** для контроля за ключевыми задачами.
* **Пример**: При задержке поставки оборудования перераспределение ресурсов на другие задачи.

### Управление ресурсами

* **Содержание**: Оптимизация загрузки людей, техники, финансов.
* **Методы**:
  + **Ресурсное выравнивание** (resource leveling) для устранения перегрузок.
  + **Использование резервов** (например, привлечение фрилансеров при пиковой нагрузке).

### Мониторинг и контроль

* **Содержание**: Сравнение «план vs факт».
* **Метрики**:
  + **CV (Cost Variance)**: Отклонение затрат. CV = EV – AC.
  + **SV (Schedule Variance)**: Отклонение по срокам. SV = EV – PV.
* **Инструменты**: Прогнозные отчеты EAC (Estimate at Completion).

### Управление рисками

* **Содержание**: Минимизация угроз проекту.
* **Тактики**:
  + **Mitigate**: Снижение вероятности риска (например, дублирование поставщиков).
  + **Transfer**: Передача риска третьей стороне (страхование).

### Управление коммуникациями

* **Содержание**: Обеспечение прозрачности.
* **Правило**: «Одна версия правды» — все данные аккумулируются в единой системе (например, Confluence).

### Управление качеством

* **Содержание**: Соответствие результатов стандартам.
* **Методы**:
  + **Статистический контроль** (Six Sigma).
  + **Инспекции и аудиты** по чек-листам.

### Управление изменениями

* **Содержание**: Корректировка плана при отклонениях.
* **Процедура**:
  + Фиксация запроса на изменение (RFC).
  + Оценка влияния на triple constraints.
  + Утверждение Change Control Board.

# Номер 2

Разработать проект программы численного решения нелинейного уравнения методом хорд, с детальной разработкой модуля визуализации полученного решения.

## Теоретическая часть по решению практического задания (Математическая модель)

### Нелинейное уравнение и метод хорд

**Нелинейное уравнение** — это уравнение вида , где f(x)— нелинейная функция. Такие уравнения часто не имеют аналитического решения, поэтому для их нахождения применяют **численные методы**, один из которых — **метод хорд** (метод секущих в модификации).

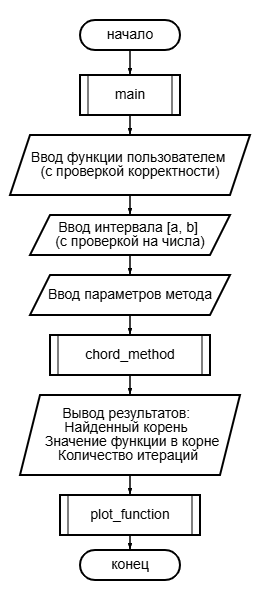
### Метод хорд

Метод хорд — метод приближённого нахождения корня нелинейного уравнения на заданном отрезке [a,b], где функция меняет знак .

### Алгоритм метода

1. **Выбираем начальные точки:** найдем отрезок , на котором функция меняет знак: .
2. **Находим точку пересечения хорды** с осью x:
3. **Проверяем условие сходимости**:
   * Если , то корень находится в , и .
   * Иначе ​.
4. **Повторяем**, пока (где ε — заданная точность).

## Блок – схема программы



|  |  |
| --- | --- |
| chord\_method(f, a, b, epsilon=1e-6, max\_iter=1000) | Реализует численный метод хорд для нахождения корня нелинейного уравнения f(x)=0 на интервале [a,b]. |
| plot\_function(f, a, b, root, iterations) | Визуализирует график функции и процесс сходимости метода |
| main() | Управляет основным потоком программы: ввод данных, вызов методов, вывод результатов. |

## Листинг программы (описание функций и инструкций программы)

### Основные компоненты программы

#### Класс ChordMethodApp

Главный класс приложения, содержащий весь функционал.

#### Методы интерфейса

* **\_\_init\_\_(self, root)**
* Инициализирует главное окно приложения
* Устанавливает размеры и заголовок окна
* Вызывает методы создания меню, виджетов и подсказок
* **create\_menu(self)**
* Создает меню "Справка" с подпунктами:
  + "Примеры функций" - показывает примеры ввода функций
  + "О программе" - информация о программе
* **create\_widgets(self)**
* Создает основные элементы интерфейса:
  + Поля ввода для функции, интервала, точности и максимального числа итераций
  + Кнопки "Решить" и "Пример"
  + Область для вывода результатов
  + Область для графиков (использует matplotlib)
* **create\_tooltips(self)**
* Добавляет всплывающие подсказки к полям ввода

### Функциональные методы

#### load\_example(self)

* Загружает пример уравнения для демонстрации

#### solve(self)

* Основной метод решения уравнения:
  1. Получает данные из полей ввода
  2. Проверяет корректность интервала
  3. Вызывает метод хорд для решения
  4. Выводит результаты
  5. Строит графики

#### chord\_method(self, f, a, b, epsilon=1e-6, max\_iter=1000)

* Реализация метода хорд:
  + Проверяет условие f(a)\*f(b) < 0
  + Итеративно находит приближение корня
  + Возвращает корень и историю итераций

#### plot\_function(self, f, a, b, root, iterations)

* Строит два графика:
  1. График функции с отмеченным корнем
  2. График сходимости метода

### Вспомогательные методы

show\_examples(self), show\_about(self), show\_info\_window(self, title, message)

Методы для отображения справочной информации

### Переменные

#### 1. Основные переменные класса ChordMethodApp

self.root - ссылка на главное окно приложения (экземпляр tk.Tk()).

self.func\_entry - поле ввода (ttk.Entry) для функции f(x).

self.a\_entry, self.b\_entry - поля ввода для границ интервала [a, b].

self.eps\_entry - поле ввода для точности вычислений (ε).

self.max\_iter\_entry - поле ввода для максимального числа итераций.

self.result\_text - текстовое поле (tk.Text) для вывода результатов.

self.figure - объект фигуры matplotlib (содержит графики).

self.ax1, self.ax2 - оси (subplots) для графиков:

ax1 - график функции и корня,

ax2 - график сходимости метода.

self.canvas - холст (FigureCanvasTkAgg) для отображения графиков в tkinter.

#### 2. Локальные переменные в методах

**Метод \_\_init\_\_**

menubar - главное меню окна (tk.Menu).

help\_menu - подменю "Справка" (tk.Menu).

**Метод create\_widgets**

input\_frame - фрейм для параметров метода (ttk.LabelFrame).

result\_frame - фрейм для результатов (ttk.LabelFrame).

graph\_frame - фрейм для графиков (ttk.LabelFrame).

**Метод create\_tooltips**

tooltips - словарь с подсказками для виджетов.

style - объект стиля (ttk.Style).

label - метка с подсказкой (ttk.Label).

**Метод solve**

func\_str - строка с функцией из поля ввода (например, "x\*\*3 - 2\*x - 5").

f - lambda-функция, созданная из func\_str (например, lambda x: x\*\*3 -2\*x- 5).

a, b - границы интервала (преобразованные в float).

epsilon - точность вычислений (float).

max\_iter - максимальное число итераций (int).

root - найденный корень уравнения (float).

iterations - список с историей итераций (каждый элемент — кортеж (номер\_итерации, приближение\_корня, значение\_функции)).

result\_str - форматированная строка с результатами.

**Метод chord\_method**

f - функция, для которой ищется корень.

a, b - текущие границы интервала (изменяются в процессе итераций).

epsilon - требуемая точность.

max\_iter - максимальное число итераций.

iterations - список для хранения истории итераций.

i - счетчик итераций.

x - текущее приближение корня.

**Метод plot\_function**

x\_vals - массив значений x (создается с помощью np.linspace).

y\_vals - массив значений f(x).

iter\_nums - номера итераций (для графика сходимости).

root\_approx - приближения корня на каждой итерации.

**Методы show\_examples, show\_about, show\_info\_window**

examples - строка с примерами функций.

about - строка с информацией о программе.

window - вспомогательное окно (Toplevel).

text - текстовое поле (tk.Text).

btn - кнопка "Закрыть" (ttk.Button).

### Особенности программы

#### Гибкий ввод функций:

* + Поддерживаются математические выражения с использованием math (sin, cos, exp, log и т.д.)
  + Есть примеры ввода различных типов функций

#### Визуализация:

* + График функции с отмеченным корнем
  + График сходимости метода

#### Пользовательский интерфейс:

* + Всплывающие подсказки для всех полей ввода
  + Примеры использования
  + Информация о программе

#### Обработка ошибок:

* + Проверка корректности входных данных
  + Информативные сообщения об ошибках

### Используемые библиотеки

* tkinter - для графического интерфейса
* matplotlib - для построения графиков
* numpy - для работы с числовыми данными
* math - для математических операций

## Программа

# Импорт библиотеки для создания графического интерфейса

import tkinter as tk

# Импорт тематических виджетов и диалоговых окон из tkinter

from tkinter import ttk, messagebox

# Импорт библиотеки для построения графиков

import matplotlib.pyplot as plt

# Импорт модуля для встраивания графиков matplotlib в tkinter

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

# Импорт библиотеки для математических вычислений

import numpy as np

# Импорт стандартной математической библиотеки

import math

# Импорт класса для создания дополнительных окон

from tkinter import Toplevel

# Основной класс приложения

class ChordMethodApp:

# Конструктор класса, вызывается при создании объекта

def \_\_init\_\_(self, root):

# Сохранение ссылки на главное окно

self.root = root

# Установка заголовка окна

self.root.title("Решение нелинейных уравнений методом хорд")

# Установка начального размера окна (ширина x высота)

self.root.geometry("1200x800")

# Вызов методов инициализации интерфейса

self.create\_menu()

self.create\_widgets()

self.create\_tooltips()

# Метод создания меню

def create\_menu(self):

# Создание главного меню

menubar = tk.Menu(self.root)

# Создание выпадающего меню "Справка"

help\_menu = tk.Menu(menubar, tearoff=0) # tearoff=0 отключает возможность отрыва меню

# Добавление пункта "Примеры функций" с привязкой к методу show\_examples

help\_menu.add\_command(label="Примеры функций", command=self.show\_examples)

# Добавление пункта "О программе" с привязкой к методу show\_about

help\_menu.add\_command(label="О программе", command=self.show\_about)

# Добавление выпадающего меню в главное меню

menubar.add\_cascade(label="Справка", menu=help\_menu)

# Установка созданного меню в главное окно

self.root.config(menu=menubar)

# Метод показа примеров функций

def show\_examples(self):

# Многострочный текст с примерами

examples = """Примеры ввода функций:

1. Полиномы:

x\*\*3 - 2\*x - 5

2\*x\*\*4 - 3\*x\*\*2 + x - 5

2. Тригонометрические функции:

math.sin(x) - 0.5

math.cos(x) - x

3. Экспоненциальные функции:

math.exp(-x) - math.sin(x)

x\*math.exp(-x) - 0.2

4. Логарифмические функции:

math.log(x) - 1

math.log10(x+1) - x/2

Примечание: используйте math. для математических функций"""

# Вызов метода для отображения окна с примерами

self.show\_info\_window("Примеры функций", examples)

# Метод показа информации о программе

def show\_about(self):

# Текст с информацией о программе

about = """Программа для решения нелинейных уравнений

методом хорд (секущих)

Версия 1.0

Автор: Ваше имя"""

# Вызов метода для отображения окна "О программе"

self.show\_info\_window("О программе", about)

# Метод создания информационного окна

def show\_info\_window(self, title, message):

# Создание нового окна

window = Toplevel(self.root)

# Установка заголовка окна

window.title(title)

# Установка размера окна

window.geometry("500x300")

# Создание текстового поля с переносом по словам

text = tk.Text(window, wrap=tk.WORD, padx=10, pady=10)

# Вставка текста в конец текстового поля

text.insert(tk.END, message)

# Блокировка редактирования текста

text.config(state=tk.DISABLED)

# Размещение текстового поля с заполнением всего пространства

text.pack(fill=tk.BOTH, expand=True)

# Создание кнопки "Закрыть"

btn = ttk.Button(window, text="Закрыть", command=window.destroy)

# Размещение кнопки с отступами

btn.pack(pady=10)

# Метод создания виджетов интерфейса

def create\_widgets(self):

# Создание фрейма для параметров с заголовком и отступами

input\_frame = ttk.LabelFrame(self.root, text="Параметры метода", padding=10)

# Размещение фрейма в сетке (0 строка, 0 столбец)

input\_frame.grid(row=0, column=0, padx=10, pady=10, sticky="nw")

# Создание метки для поля ввода функции

ttk.Label(input\_frame, text="Функция f(x):").grid(row=0, column=0, sticky="w")

# Создание поля ввода функции шириной 30 символов

self.func\_entry = ttk.Entry(input\_frame, width=30)

# Размещение поля ввода в сетке фрейма

self.func\_entry.grid(row=0, column=1, padx=5, pady=5)

# Вставка примера функции по умолчанию

self.func\_entry.insert(0, "x\*\*3 - 2\*x - 5")

# Создание метки для интервала

ttk.Label(input\_frame, text="Интервал [a, b]:").grid(row=1, column=0, sticky="w")

# Создание поля ввода для 'a' шириной 10 символов

self.a\_entry = ttk.Entry(input\_frame, width=10)

# Размещение поля 'a' с выравниванием по левому краю

self.a\_entry.grid(row=1, column=1, padx=5, pady=5, sticky="w")

# Вставка значения по умолчанию для 'a'

self.a\_entry.insert(0, "1")

# Создание поля ввода для 'b' шириной 10 символов

self.b\_entry = ttk.Entry(input\_frame, width=10)

# Размещение поля 'b' с выравниванием по правому краю

self.b\_entry.grid(row=1, column=1, padx=5, pady=5, sticky="e")

# Вставка значения по умолчанию для 'b'

self.b\_entry.insert(0, "3")

# Создание метки для точности

ttk.Label(input\_frame, text="Точность:").grid(row=2, column=0, sticky="w")

# Создание поля ввода для точности

self.eps\_entry = ttk.Entry(input\_frame, width=10)

# Размещение поля точности

self.eps\_entry.grid(row=2, column=1, padx=5, pady=5, sticky="w")

# Вставка значения точности по умолчанию

self.eps\_entry.insert(0, "1e-6")

# Создание метки для максимального числа итераций

ttk.Label(input\_frame, text="Макс. итераций:").grid(row=3, column=0, sticky="w")

# Создание поля ввода для итераций

self.max\_iter\_entry = ttk.Entry(input\_frame, width=10)

# Размещение поля итераций

self.max\_iter\_entry.grid(row=3, column=1, padx=5, pady=5, sticky="w")

# Вставка значения по умолчанию

self.max\_iter\_entry.insert(0, "100")

# Создание кнопки "Решить" с привязкой к методу solve

ttk.Button(input\_frame, text="Решить", command=self.solve).grid(row=4, column=0, columnspan=2, pady=10)

# Создание кнопки "Пример" с привязкой к методу load\_example

ttk.Button(input\_frame, text="Пример", command=self.load\_example).grid(row=5, column=0, columnspan=2, pady=5)

# Создание фрейма для результатов

result\_frame = ttk.LabelFrame(self.root, text="Результаты", padding=10)

# Размещение фрейма результатов

result\_frame.grid(row=1, column=0, padx=10, pady=10, sticky="nw")

# Создание текстового поля для вывода результатов

self.result\_text = tk.Text(result\_frame, height=5, width=50)

# Размещение текстового поля

self.result\_text.grid(row=0, column=0, padx=5, pady=5)

# Создание фрейма для графиков

graph\_frame = ttk.LabelFrame(self.root, text="Графики", padding=10)

# Размещение фрейма графиков (объединение двух строк)

graph\_frame.grid(row=0, column=1, rowspan=2, padx=10, pady=10, sticky="nsew")

# Создание фигуры matplotlib с 1 строкой и 2 столбцами графиков

self.figure, (self.ax1, self.ax2) = plt.subplots(1, 2, figsize=(10, 4))

# Создание холста для встраивания графиков в tkinter

self.canvas = FigureCanvasTkAgg(self.figure, master=graph\_frame)

# Размещение холста с заполнением всего доступного пространства

self.canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)

# Настройка растягивания столбца 1 при изменении размеров окна

self.root.columnconfigure(1, weight=1)

# Настройка растягивания строк 0 и 1

self.root.rowconfigure(0, weight=1)

self.root.rowconfigure(1, weight=1)

# Метод создания всплывающих подсказок

def create\_tooltips(self):

# Словарь с подсказками для каждого поля ввода

tooltips = {

self.func\_entry: "Примеры: x\*\*3 - 2\*x - 5, math.sin(x) - 0.5",

self.a\_entry: "Левая граница интервала (число)",

self.b\_entry: "Правая граница интервала (число, должно быть > a)",

self.eps\_entry: "Точность решения (например, 1e-6)",

self.max\_iter\_entry: "Максимальное число итераций (целое число)"

}

# Создание объекта стиля

style = ttk.Style()

# Настройка стиля для подсказок

style.configure("Tooltip.TLabel",

background="#ffffe0", # Цвет фона

relief="solid", # Граница

padding=5, # Отступы

borderwidth=1) # Толщина границы

# Создание подсказок для каждого виджета

for widget, text in tooltips.items():

# Создание метки с подсказкой

label = ttk.Label(self.root,

text=text,

style="Tooltip.TLabel",

wraplength=200) # Максимальная ширина перед переносом

# Функция показа подсказки при наведении

def enter(event, lbl=label):

# Позиционирование подсказки рядом с курсором

lbl.place(x=event.x\_root - self.root.winfo\_rootx(),

y=event.y\_root - self.root.winfo\_rooty() + 20)

# Функция скрытия подсказки

def leave(event, lbl=label):

lbl.place\_forget()

# Привязка событий наведения и ухода курсора

widget.bind("<Enter>", enter)

widget.bind("<Leave>", leave)

# Метод загрузки примера

def load\_example(self):

# Очистка и установка примера функции

self.func\_entry.delete(0, tk.END)

self.func\_entry.insert(0, "math.sin(x) - x/2")

# Очистка и установка примера для 'a'

self.a\_entry.delete(0, tk.END)

self.a\_entry.insert(0, "1")

# Очистка и установка примера для 'b'

self.b\_entry.delete(0, tk.END)

self.b\_entry.insert(0, "3")

# Очистка и установка примера точности

self.eps\_entry.delete(0, tk.END)

self.eps\_entry.insert(0, "1e-6")

# Очистка и установка примера итераций

self.max\_iter\_entry.delete(0, tk.END)

self.max\_iter\_entry.insert(0, "100")

# Метод решения уравнения

def solve(self):

try:

# Получение строки функции из поля ввода

func\_str = self.func\_entry.get()

# Создание lambda-функции из строки с доступом к math

f = lambda x: eval(func\_str, {'x': x, 'math': math})

# Получение параметров из полей ввода

a = float(self.a\_entry.get())

b = float(self.b\_entry.get())

epsilon = float(self.eps\_entry.get())

max\_iter = int(self.max\_iter\_entry.get())

# Проверка корректности интервала

if a >= b:

raise ValueError("a должно быть меньше b")

# Вызов метода хорд для решения уравнения

root, iterations = self.chord\_method(f, a, b, epsilon, max\_iter)

# Формирование строки с результатами

result\_str = f"Найденный корень: {root:.8f}\n"

result\_str += f"Значение функции: {f(root):.3e}\n"

result\_str += f"Итераций: {len(iterations)}\n"

result\_str += f"Условие f(a)\*f(b) < 0: {f(a)\*f(b) < 0}"

# Очистка и вывод результатов в текстовое поле

self.result\_text.delete(1.0, tk.END)

self.result\_text.insert(tk.END, result\_str)

# Построение графиков

self.plot\_function(f, a, b, root, iterations)

except Exception as e:

# Показ сообщения об ошибке

messagebox.showerror("Ошибка", str(e))

# Реализация метода хорд

def chord\_method(self, f, a, b, epsilon=1e-6, max\_iter=1000):

# Проверка условия сходимости метода

if f(a) \* f(b) >= 0:

raise ValueError("Функция должна иметь разные знаки на концах интервала")

# Список для хранения истории итераций

iterations = []

# Основной цикл метода

for i in range(max\_iter):

# Формула метода хорд

x = a - f(a) \* (b - a) / (f(b) - f(a))

# Сохранение информации о текущей итерации

iterations.append((i, x, f(x)))

# Проверка достижения требуемой точности

if abs(f(x)) < epsilon:

return x, iterations

# Выбор нового интервала

if f(a) \* f(x) < 0:

b = x

else:

a = x

# Если не сошлось за max\_iter итераций

raise ValueError(f"Метод не сошелся за {max\_iter} итераций")

# Метод построения графиков

def plot\_function(self, f, a, b, root, iterations):

# Очистка предыдущих графиков

self.ax1.clear()

self.ax2.clear()

# Создание массива x-значений

x\_vals = np.linspace(a, b, 400)

# Вычисление y-значений для функции

y\_vals = [f(x) for x in x\_vals]

# Построение графика функции

self.ax1.plot(x\_vals, y\_vals, label='f(x)')

# Горизонтальная линия y=0

self.ax1.axhline(0, color='black', linewidth=0.5)

# Вертикальная линия корня

self.ax1.axvline(root, color='red', linestyle='--', label=f'Корень: {root:.6f}')

# Точки границ интервала

self.ax1.scatter([a, b], [f(a), f(b)], color='green', label='Границы интервала')

# Настройки первого графика

self.ax1.set\_title('График функции и найденный корень')

self.ax1.set\_xlabel('x')

self.ax1.set\_ylabel('f(x)')

self.ax1.legend()

self.ax1.grid()

# Подготовка данных для графика сходимости

iter\_nums = [it[0] for it in iterations] # Номера итераций

root\_approx = [it[1] for it in iterations] # Приближения корня

# Построение графика сходимости

self.ax2.plot(iter\_nums, root\_approx, 'o-', label='Приближение корня')

# Линия истинного корня

self.ax2.axhline(root, color='red', linestyle='--', label='Истинный корень')

# Настройки второго графика

self.ax2.set\_title('Сходимость метода хорд')

self.ax2.set\_xlabel('Номер итерации')

self.ax2.set\_ylabel('Приближение корня')

self.ax2.legend()

self.ax2.grid()

# Оптимизация расположения графиков

self.figure.tight\_layout()

# Обновление холста

self.canvas.draw()

# Точка входа в программу

if \_\_name\_\_ == "\_\_main\_\_":

# Создание главного окна

root = tk.Tk()

# Создание экземпляра приложения

app = ChordMethodApp(root)

# Запуск главного цикла обработки событий

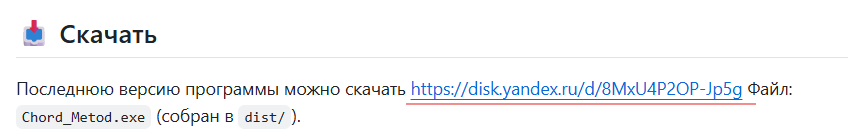
root.mainloop()

## Визуализация

### Скачать проект с GitHub:



### Перейти по QR-коду на страницу репозитория с проектом и зайти по ссылке на яндекс диск:



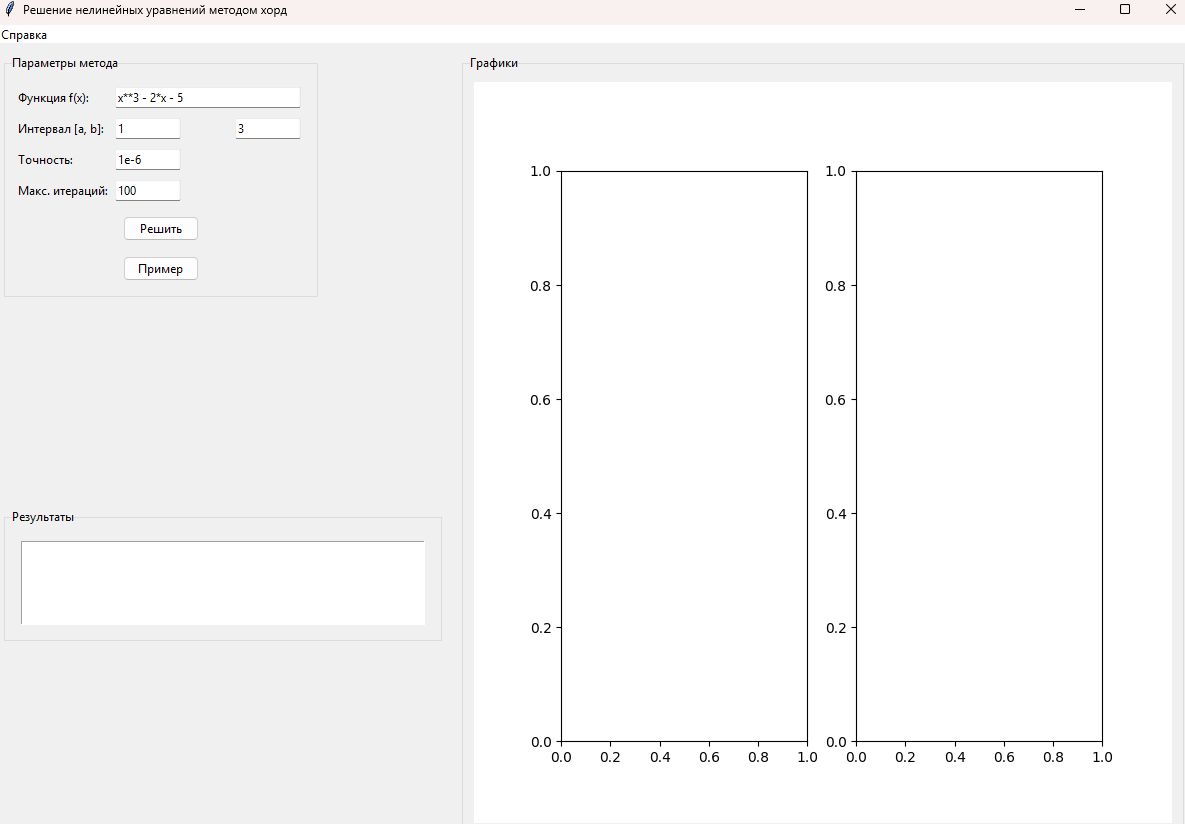
### В яндекс диске скачать один из архивов:



### Разархивировать проект:

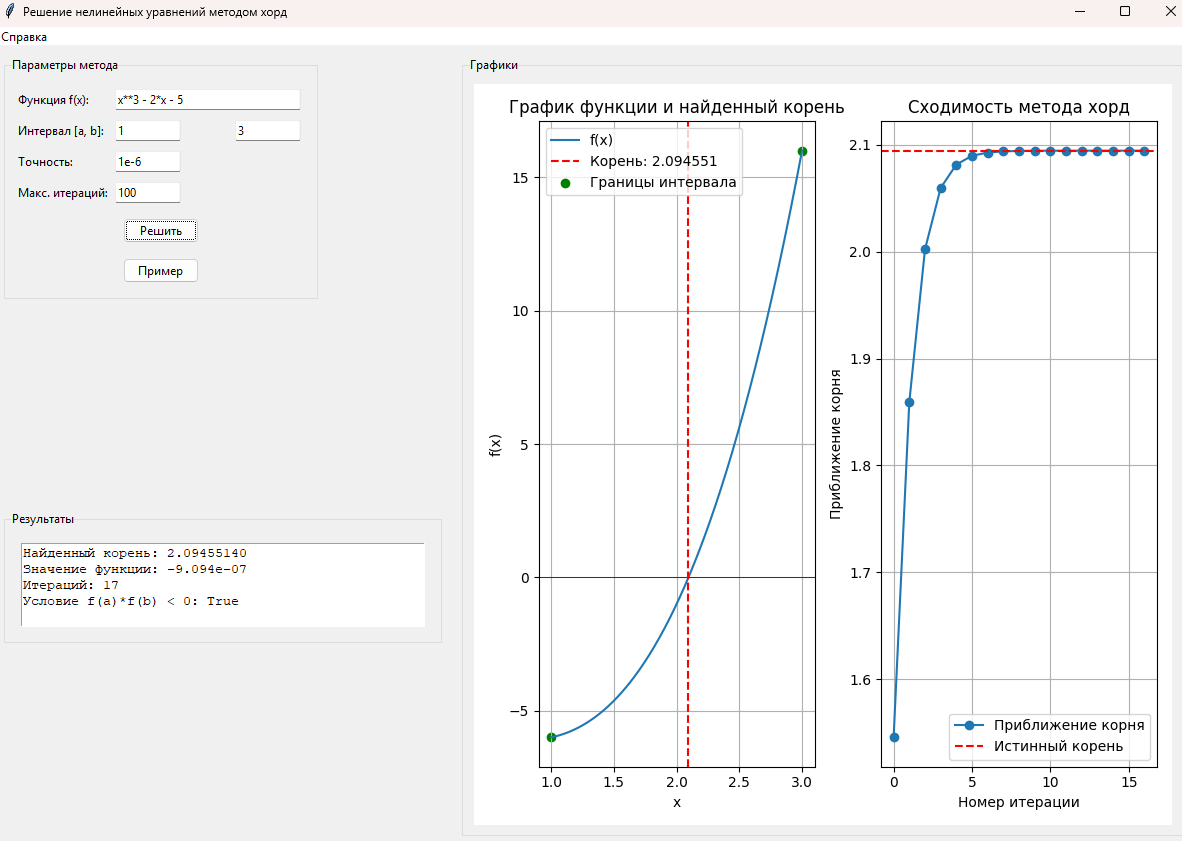


### Запустить приложение, где появится окно с вычислением уравнений по методу хорд:



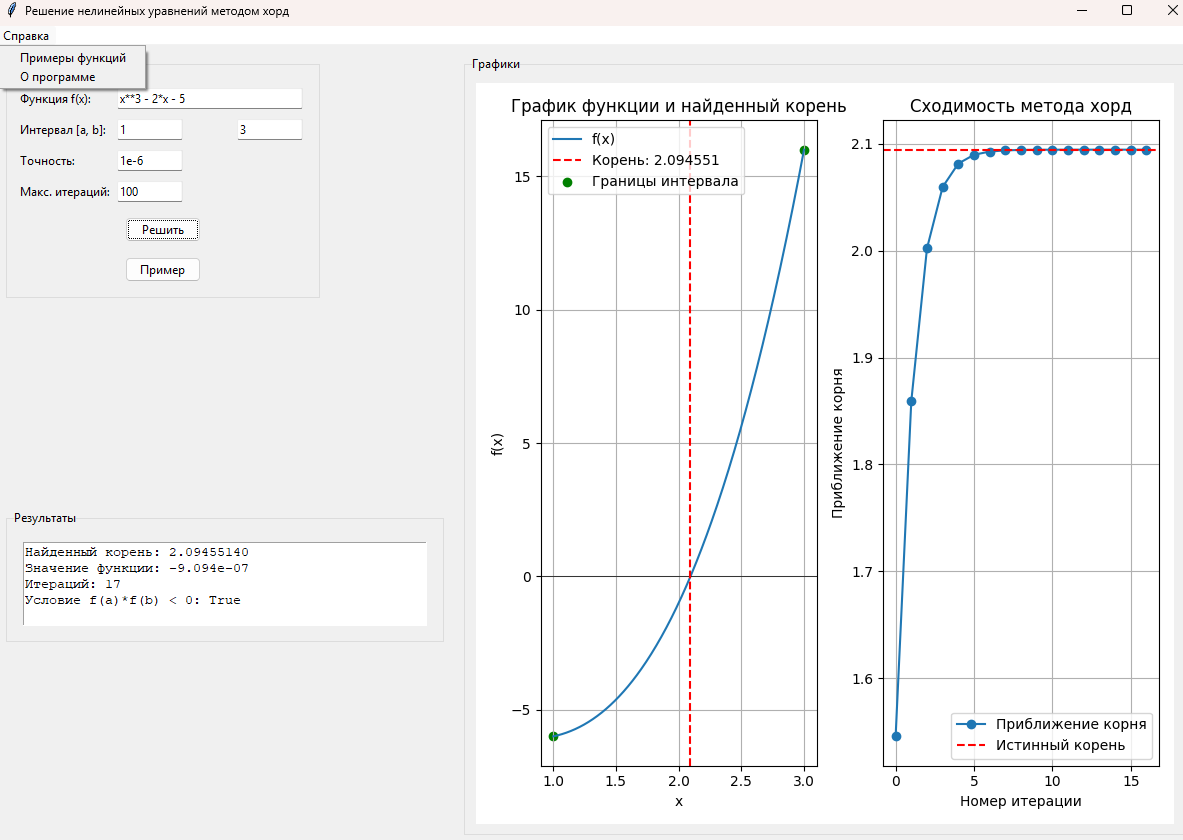
### Вбить необходимое уравнение

Для вычисления в строку «Функция» вбиваем необходимое уравнение, далее вбиваем интервалы, точность, с которой хотим найти ответ, и максимальное количество итераций, за которое программа должна найти ответ и нажимаем на кнопку решить:

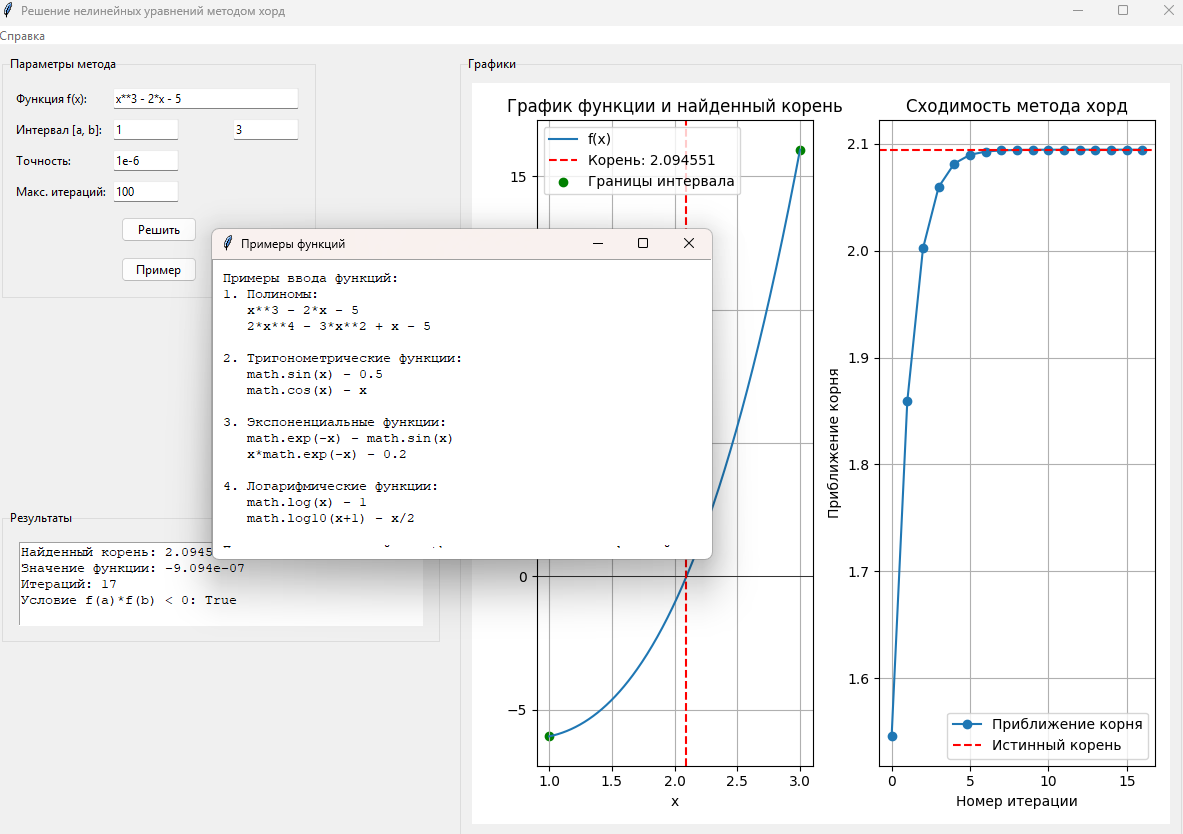


Как видим выше программа высчитываем необходимый корень, значение функции в данном корне, и количество итераций, за которое был найден ответ.

### Справка

Чтобы просмотреть какого рода функции можно вбивать в программу есть сверху слева кнопка справка:  


Нажать на примеры функций:



По окончанию работы нажать на крестик, чтобы выйти из программы.

# Список литературы

1. Основная литература
2. Самарский, А. А. Численные методы: учебное пособие для вузов / А. А. Самарский, А. В. Гулин. — Москва : Наука, 2023. — 432 с. — (Высшее образование). — ISBN 978-5-02-034710-4
3. Калиткин, Н. Н. Численные методы: учебник для бакалавриата / Н. Н. Калиткин. — 3-е изд., перераб. — Москва : Юрайт, 2024. — 380 с. — ISBN 978-5-534-12345-6.
4. Бахвалов, Н. С. Численные методы: учебное пособие / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. — Москва : ИНФРА-М, 2022. — 640 с. — (Высшее образование). — ISBN 978-5-16-015678-1.
5. Полевая, Н. М. Основы проектной деятельности: учебное пособие / Н. М. Полевая, В. В. Ситникова. — Новосибирск : НГУЭУ, 2023. — 180 с. — ISBN 978-5-7014-1100-3.
6. Алферова, Л. А. Основы проектной деятельности: учебник / Л. А. Алферова. — Москва : ИНФРА-М, 2024. — 210 с. — (Высшее образование). — ISBN 978-5-16-017890-5.
7. Метод хорд и его модификации // Электронный журнал "Вычислительные методы и программирование". — 2023. — № 4. — С. 45–58.
8. Литература по Python и визуализации
9. Маккинни, У. Python для анализа данных: учебное пособие / У. Маккинни. — 2-е изд. — Москва : Диалектика, 2023. — 544 с. — ISBN 978-5-907543-12-3.
10. Хэррис, Ч. Математическая визуализация с Matplotlib: практикум / Ч. Хэррис. — Санкт-Петербург : Питер, 2024. — 256 с. — ISBN 978-5-4461-2345-6.