МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет «Информатика и вычислительная техника»
(наименование факультета)

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»
(наименование кафедры)

Зав. кафедрой «ПОВТиАС»
_____ В.В. Долгов
(подпись) (И.О.Ф.)

«_____» _____ 2024 г.

ОТЧЕТ

по учебной технологической (проектно-технологической) практике
вид практики

в научно-образовательном производственном центре «Визуализация, анализ и защита информации» (НОПЦ «ВАиЗИ»)
наименование базы практики

Обучающийся _____ Н.А. Дышко
подпись, дата И.О.Ф.

Обозначение отчета УП.190000.000 Группа ВМО11

Направление 02.03.03 Математическое обеспечение и администрирование информационных систем
код наименование направления подготовки

Профиль Математическое обеспечение и администрирование информационных систем

Руководитель практики:

от предприятия руководитель _____ В.В. Долгов
должность подпись, дата имя, отчество, фамилия

от кафедры доцент _____ Т.А. Медведева
должность подпись, дата имя, отчество, фамилия

Оценка _____ _____ _____
дата подпись преподавателя

Ростов-на-Дону
2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет «Информатика и вычислительная техника»
(наименование факультета)

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»
(наименование кафедры)

ЗАДАНИЕ

на учебной технологической (проектно-технологической) практике
вид практики

в научно-образовательном производственном центре «Визуализация, анализ и защита информации» (НОПЦ «ВАиЗИ»)
наименование базы практики

в период с «24» июня 2024 г. по «6» июля 2024 г.

Обучающийся Дышко Никита Артёмович

Обозначение отчета УП.190000.000 Группа ВМО11

Срок представления отчета на кафедру «6» июля 2024 г.

Содержание индивидуального задания:

Программная реализация решения нелинейных уравнений двушаговым методом секущих

Руководитель практики от кафедры _____ (подпись, дата) Т.А. Медведева (И.О.Ф.)

Задание принял к исполнению _____ (подпись, дата) Н.А. Дышко (И.О.Ф.)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет «Информатика и вычислительная техника»
(наименование факультета)

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»
(наименование кафедры)

Зав. кафедрой «ПОВТиАС»
_____ В.В. Долгов
(подпись) (И.О.Ф.)
«_____» _____ 2024 г.

Рабочий график (план) проведения практики

№	Мероприятие	Срок выполнения
1	Прохождение вводного и первичного инструктажа по охране труда на рабочем месте, и инструктажа по пожарной безопасности на объекте	24.06.2024
2	Получение индивидуального задания и постановка Задачи	24.06.2024
3	Изучение теоретических материалов по численным методам решения нелинейных уравнений	С 25.06.2024 по 27.06.2024
4	Алгоритмическое конструирование решения нелинейных уравнений двушаговым методом секущих	С 28.06.2024 по 30.06.2024
5	Выбор среды разработки и реализация программного средства	С 01.07.2024 по 03.07.2024
6	Проверка работоспособности программы, составление отчета о проделанной работе	С 04.07.2024 по 05.07.2024
7	Защита итогового отчета по практике	6.07.2024

Руководитель практики:
от предприятия руководитель _____ В.В. Долгов
должность подпись, дата имя, отчество, фамилия

от кафедры доцент _____ Т.А. Медведева
должность подпись, дата имя, отчество, фамилия

Ростов-на-Дону
2024 г.

ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИЧЕСКОЙ ПОДГОТОВКИ

Дата	Место работы	Выполняемые работы	Оценка руководителя
24.06.2024	НОПЦ «ВАиЗИ»	Знакомство с предприятием, прохождение вводного инструктажа	
24.06.2024	НОПЦ «ВАиЗИ»	Ознакомление с территорией предприятия, прохождение первичного инструктажа по ТБ, ПБ	
24.06.2024	НОПЦ «ВАиЗИ»	Получение индивидуального задания и постановка задачи	
С 25.06.2024 по 27.06.2024	НОПЦ «ВАиЗИ»	Изучение теоретических материалов по численным методам решения нелинейных уравнений	
С 28.06.2024 по 30.06.2024	НОПЦ «ВАиЗИ»	Алгоритмическое конструирование решения нелинейных уравнений двушаговым методом секущих	
С 01.07.2024 по 03.07.2024	НОПЦ «ВАиЗИ»	Выбор среды разработки и реализация программного средства	
С 04.07.2024 по 05.07.2024	НОПЦ «ВАиЗИ»	Проверка работоспособности программы, составление отчета о проделанной работе	
6.07.2024	НОПЦ «ВАиЗИ»	Защита итогового отчета по практике	

Руководитель практики:

от предприятия руководитель _____ В.В. Долгов

должность подпись, дата имя, отчество, фамилия

ОТЗЫВ – ХАРАКТЕРИСТИКА

Обучающийся Дышко Никита Артёмович
фамилия, имя, отчество

1 курса группы ВМО кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Вид практики учебной технологической (проектно-технологической) практике

Наименование места практики научно-образовательный производственный центр «Визуализация, анализ и защита информации» (НОПЦ «ВАиЗИ»)
наименование предприятия, структурного подразделения

Обучающийся выполнил задания программы практики
Реализовано и протестировано программное средство с удобным интерфейсом для решения нелинейных уравнений двушаговым методом секущих.
Дополнительно ознакомился/изучил
Численные методы решения нелинейных уравнений, метод секущих, библиотеки Matplotlib, NumPy. Изучил среду разработки графических интерфейсов.

Заслуживает оценки

_____

Руководитель практики от предприятия

_____

«____»_____2024 г.

Б.П.

Содержание

					УП.190000.000		
Изм.	Лист	№ докум	Подпись	Дата			
Разраб.		Дышко Н. А.			Программная реализация решения нелинейных уравнений двушаговым методом секущих	Лит.	Лист 6 / Листов 32
Пров.		Медведева Т. А.					
Н. Контр.						ДГТУ кафедра «ПОВТиАС»	
Утв.							

Введение

При решении научных и прикладных задач в различных областях, таких как физика, химия, инженерия и экономика, часто требуется находить корни нелинейных уравнений. Нелинейные уравнения могут описывать сложные системы и процессы, где зависимость между переменными выходит за пределы линейных отношений. Решение таких уравнений аналитическими методами возможно лишь в ограниченном числе случаев, и даже тогда полученные решения могут быть громоздкими и неудобными для применения. В таких ситуациях на помощь приходят численные методы, которые позволяют находить приближенные решения с заданной точностью.

История численных методов решения нелинейных уравнений восходит к началу развития математического анализа и численных методов в XVII-XVIII веках, когда учёные начали искать способы приближенного решения уравнений, которые не поддавались аналитическому решению. В это время разработаны методы деления пополам и Ньютона-Рафсона, которые стали основой для дальнейших исследований и усовершенствований в области численного анализа. С тех пор численные методы значительно развились, включая такие подходы, как итерационные методы, методы касательных и полиномиальные интерполяции, что позволило эффективно решать широкий класс задач в различных областях науки и техники.

Одним из наиболее эффективных и широко применяемых методов решения нелинейных уравнений является метод секущих. Этот метод относится к итерационным численным методам и является модификацией метода Ньютона, но в отличие от последнего метод секущих не требует вычисления производной функции в каждом шаге. Вместо этого используется аппроксимация производной разностью значений функций в двух точках, что делает метод секущих более универсальным и простым в реализации.

Целью данной работы является разработка программного средства с удобным интерфейсом для решения нелинейных уравнений с использованием двушагового метода секущих.

1 Теоретический обзор

В данном разделе представлен теоретический обзор численноых методов решения нелинейных уравнений. Подробно рассматривается двушаговый метод секущих.

1.1 Численные методы решения нелинейных уравнений

Численные методы решения нелинейных уравнений представляют собой важный инструмент в математическом анализе, позволяющий находить приближенные решения уравнений, для которых отсутствуют аналитические методы. Эти методы, такие как метод Ньютона и метод деления пополам, играют ключевую роль в различных областях науки и техники, обеспечивая эффективное решение сложных задач. Нелинейные уравнения представляют собой уравнения вида $f(x) = 0$, где функция $f(x)$ может иметь любую сложную форму. Такие уравнения часто возникают в прикладных задачах, но, как правило, их аналитическое решение невозможно. В связи с этим разработаны численные методы, которые позволяют получать приближённые решения. Основной задачей численных методов является нахождение такого значения x, при котором выполняется приближённое равенство $f(x) \approx 0$ с допустимой ошибкой.

Одними из распространенных численных методов решения нелинейных уравнений являются метод половинного деления и Ньютона.

Метод бисекции, или деления отрезка пополам, используется для поиска корня функции, если на концах отрезка функция $f(x)$ меняет знак, то есть $f(a) \cdot f(b) < 0$. Метод гарантирует сходимость и является простым в реализации, но его скорость работы относительно низкая.

Алгоритм метода:

1. Задать функцию, начальный интервал $[a, b]$, на котором она меняет знак и точность ε .
2. Вычислить середину отрезка: $c = \frac{a+b}{2}$.

3. Проверить значение функции в точке c:

   Если $f(c) = 0$, то c — корень.

   Если $f(a) \cdot f(c) < 0$, то корень находится в интервале $[a, c]$;

   присвоить $b = c$.

   Иначе корень находится в интервале $[c, b]$; присвоить $a = c$.

4. Если $|b - a| > \varepsilon$, то переход на пункт 2.

На каждом шаге длина отрезка делится пополам, и погрешность метода уменьшается экспоненциально с числом итераций. Однако скорость сходимости метода считается медленной.

Метод Ньютона (касательных) — это итерационный метод, использующий производную функции для нахождения корня. Этот метод быстрее, чем метод половинного деления, но его использование требует знания производной функции, и он может не сходиться для неудачно выбранных начальных точек.

Алгоритм метода:

1. Задать начальное приближение x_0, функцию и точность ε.
2. На каждом шаге вычислять новое приближение по формуле:
3. $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
4. Повторять процесс, пока разность $|x_{n+1} - x_n|$ не станет меньше заданной точности ε.

Метод Ньютона обладает квадратичной сходимостью, что означает, что с каждым шагом точность возрастает значительно быстрее, чем у метода половинного деления. Однако сходимость не гарантирована, если начальная точка x_0 выбрана далеко от корня или производная функции вблизи корня мала.

1.2 Двушаговый метод секущих

Двушаговый метод секущих — итерационный численный метод решения нелинейных уравнений, который основан на линейном приближении функции с помощью секущей, построенной по двум последовательно

выбранным точкам x_{n-1} и x_n. На каждом шаге секущая пересекает ось абсцисс в новой точке, которая служит приближением к корню. Формула двушагового метода секущих имеет вид:

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

где x_{n+1} — новое приближение, а x_n и x_{n-1} — два предыдущих значения.

Алгоритм работы метода:

1. Выбрать интервал $[a, b]$. Задать точность ε, которая используется для остановки итераций. Ввести ограничитель итераций N_{max}.
2. Присвоить $a := x1,\ b := x2$, ввести счетчик количества итераций $n = 0$.
3. На каждой итерации вычисляется новое приближение по формуле двушагового метода секущих:

$$x2 = x1 - f(x1) \cdot \frac{x1 - x0}{f(x1) - f(x0)}$$

4. Обновить значения $x0 := x1$ и $x1 := x2$ и присвоить $n := n + 1$.
5. Если достигнуто максимальное количество итераций $n > N_{max}$ или условие остановки $|x2 - x1| < \varepsilon$, то перейти к шагу 6, в противном случае вернуться к шагу 3.
6. После завершения итерационного процесса вывести найденное значение $x2$, как приближенное решение уравнения $f(x) = 0$, а также $f(x2), n$.

Метод секущих обладает сверхлинейной сходимостью, что делает его более быстрым по сравнению с методом половинного деления. Однако он может не сходиться, если начальные приближения x_0 и x_1 выбраны неудачно или если функция не является непрерывной в рассматриваемой области.

Приближение улучшает точность при каждом шаге. Теоретически, метод секущих имеет порядок сходимости около 1.618 (золотое сечение), что делает его быстрее, чем метод половинного деления, но медленнее, чем метод Ньютона для хорошо выбранных начальных условий.

Метод секущих полезен в задачах, где вычисление производной функции затруднительно или невозможно, так как он использует только значения функции и не требует дополнительных данных о её характеристиках.

1.3 Выводы по главе

В данной главе рассмотрен и изучен теоретический материал по численным методам решения нелинейных уравнений: метод половинного деления Ньютона (касательных) и более подробно двушаговый метод секущих.

1.4 Постановка задачи

Изучить теоретический материал, связанный с нелинейными уравнениями и численными методами их решений. Разработать программное средство (ПС), позволяющее с помощью интуитивно понятного интерфейса вводить функции и вычислять корни нелинейных уравнений на заданных интервалах с помощью двушагового метода секущих с визуализацией найденных корней на графике функции. Протестировать программу с различными нелинейными уравнениями и различными интервалами нахождения корней.

2 Алгоритмическое конструирование

В данном разделе описаны основные алгоритмы, применяемые для решения нелинейных уравнений двушаговым методом секущих.

2.1 Общая схема реализации программного средства

На рисунке 4 представлена общая схема алгоритма работы программы.

Рисунок 4 – Общая схема реализации программы

2.2 Схема метода секущих

На рисунке 5 показана схема двушагового метода секущих.

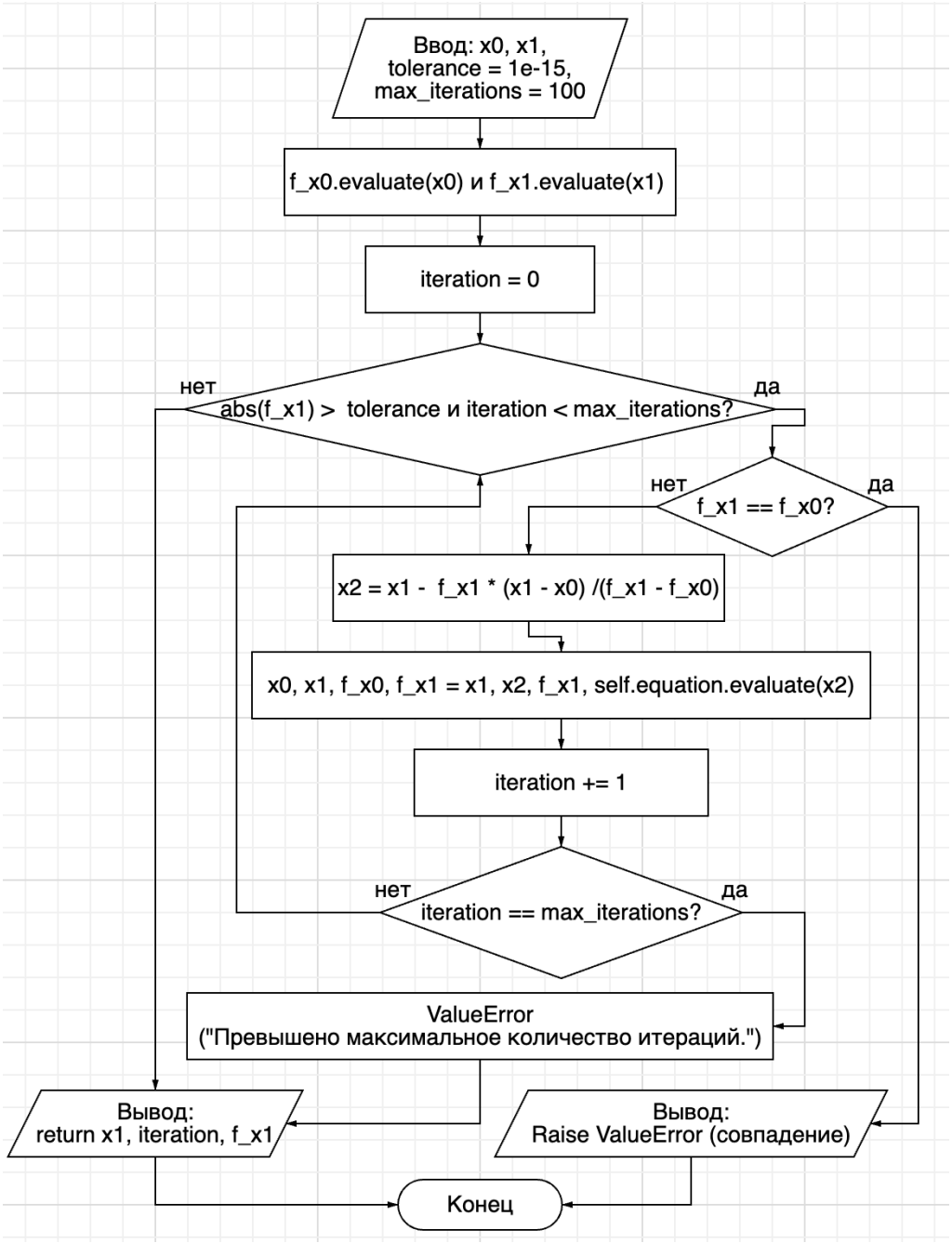

Рисунок 5 – Схема двушагового метода секущих

2.3 Выводы по главе

В данной главе представлены основные схемы алгоритмов, использованные в программном средстве. С помощью двушагового метода секущих может быть осуществлено решение нелинейного уравнения.

3 Программное конструирование

Данная глава посвящена обоснованию выбора языка программирования и среды разработки для реализации программного средства. Подробно описан программный продукт и использованные в нем библиотеки.

3.1 Выбор языка программирования и среды разработки

Для разработки программного обеспечения, особенно при работе с численными методами решения нелинейных уравнений, выбор Python в качестве языка программирования и PyCharm в качестве интегрированной среды разработки является обоснованным.

PyCharm – это интегрированная среда разработки (IDE) для языка программирования Python, созданная компанией JetBrains. PyCharm поддерживает Pandas, Numpy, Matplotlib, Pillow, Sys, Sympy и другие библиотеки для научных вычислений. PyCharm предоставляет мощные инструменты, такие как подсветка синтаксиса, автодополнение кода и отладчик, что значительно упрощает процесс написания и отладки кода. Python, в свою очередь, известен своей простотой и понятным синтаксисом, что делает его идеальным выбором для новичков в программировании и опытных разработчиков. PyCharm является одним из лидирующих продуктов на рынке программных средств, используемых для разработки программного обеспечения.

Python – это интерпретируемый, интерактивный, объектно-ориентированный и высокоуровневый язык программирования общего назначения с динамической типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода, а также на обеспечение переносимости написанных на нем программ. Python нашел широкое применение в научных и инженерных расчетах, анализе данных, машинном обучении, веб-разработке и многих других областях благодаря своему простому и

интуитивно понятному синтаксису. Это делает его идеальным выбором как для начинающих программистов, так и для опытных специалистов, которые ценят эффективность и удобство в разработке программного обеспечения.

Основные преимущества выбора PyCharm и Python для разработки программ, включают:

- Высокая скорость разработки: по сравнению с компилирующими или строго типизированными языками, такими как C, C++ и Java, Python во много раз повышает производительность труда разработчика. Объем программного кода на языке Python обычно составляет треть или даже пятую часть эквивалентного программного кода на языке C++ или Java. Это означает меньший объем ввода с клавиатуры, меньшее количество времени на отладку и меньший объем трудозатрат на сопровождение. Кроме того, программы на языке Python запускаются сразу же, минуя длительные этапы компиляции и связывания, необходимые в некоторых других языках программирования, что еще больше увеличивает производительность труда программиста.
- Обширная библиотека и экосистема: Python имеет обширную стандартную библиотеку, а также большое количество сторонних библиотек и фреймворков. Это позволяет разработчикам использовать готовые решения для различных задач, ускоряя процесс разработки и снижая объем написанного кода.
- Простой и понятный синтаксис: Python имеет простой и понятный синтаксис, который делает его доступным даже для новичков в программировании. Читаемость кода на Python помогает разработчикам легко понимать и модифицировать программы.
- Кроссплатформенность: Большая часть программ на языке Python выполняется без изменений на всех основных платформах. Перенос программного кода из операционной системы Linux, Windows и macOS обычно заключается в простом копировании программного

кода сценария с одной машины на другую. Более того, Python предоставляет массу возможностей по созданию переносимых графических интерфейсов, программ доступа к базам данных, веб приложений и многих других типов программ. Даже интерфейсы операционных систем, включая способ запуска программ и обработку каталогов, в языке Python реализованы переносимым способом.

- Большое сообщество: Python находится под опекой активного сообщества разработчиков, где можно получить необходимую помощь, задать вопросы, найти решения и обменяться опытом. Эта поддержка помогает разработчикам эффективно решать возникающие проблемы и быть в курсе последних тенденций и инноваций в области разработки.
- Интеграция PyCharm с такими инструментами, как Git для контроля версий и виртуальные окружения для управления зависимостями, дополняет его функциональность и обеспечивает эффективный рабочий процесс. Благодаря активному сообществу разработчиков Python, пользователи могут легко находить поддержку, делиться опытом и следить за последними тенденциями в разработке программного обеспечения.

Таким образом, использование Python и PyCharm для разработки программ обеспечивает высокую производительность и удобство в работе благодаря их мощным функциональным возможностям и поддержке сообщества разработчиков.

3.2 Описание программы

Разработанная программа позволяет решать нелинейные уравнения с помощью двушагового метода секущих.

Для выполнения вычислений в программе требуется ввести нелинейную функцию и интервал нахождения корней. На рисунке 7 представлен основной интерфейс программы.

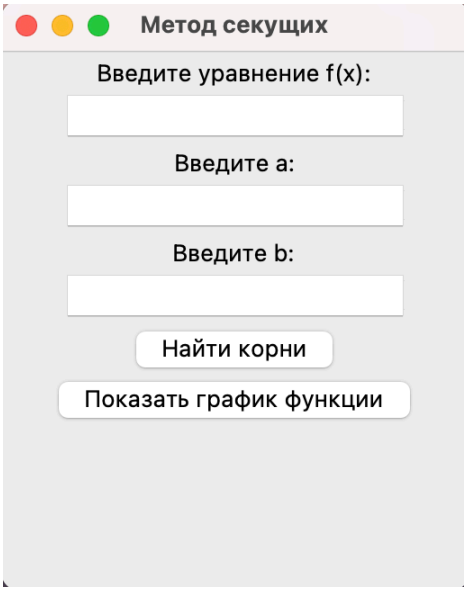

Рисунок 7 – Интерфейс программы

После ввода всех значений необходимо нажать на кнопку «Найти корни». Далее на экран выводятся найденные корни нелинейного уравнения. Пример приведён на рисунке 8.

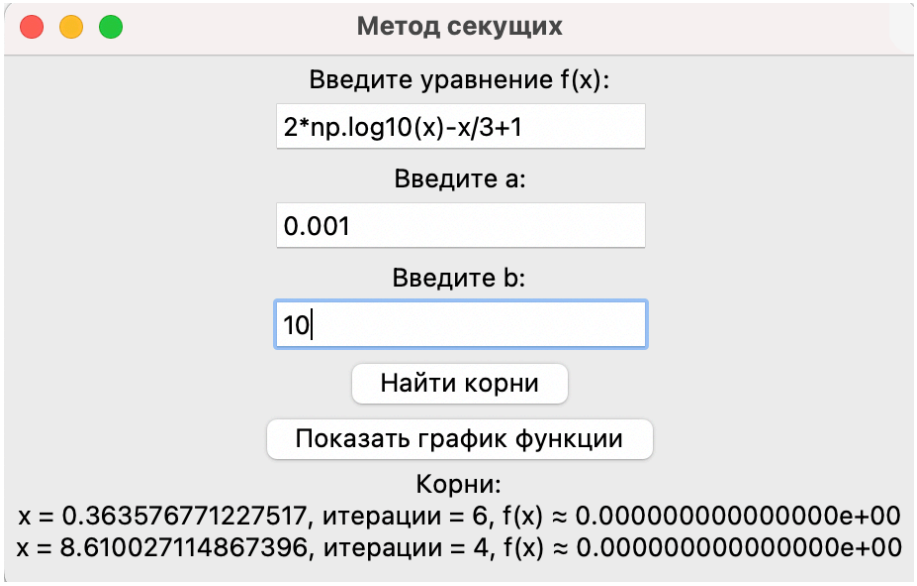

Рисунок 8 – Пример вывода найденных корней

При нажатии на кнопку «показать график функции», строится график функции, на котором отмечены найденные корни нелинейного уравнения.

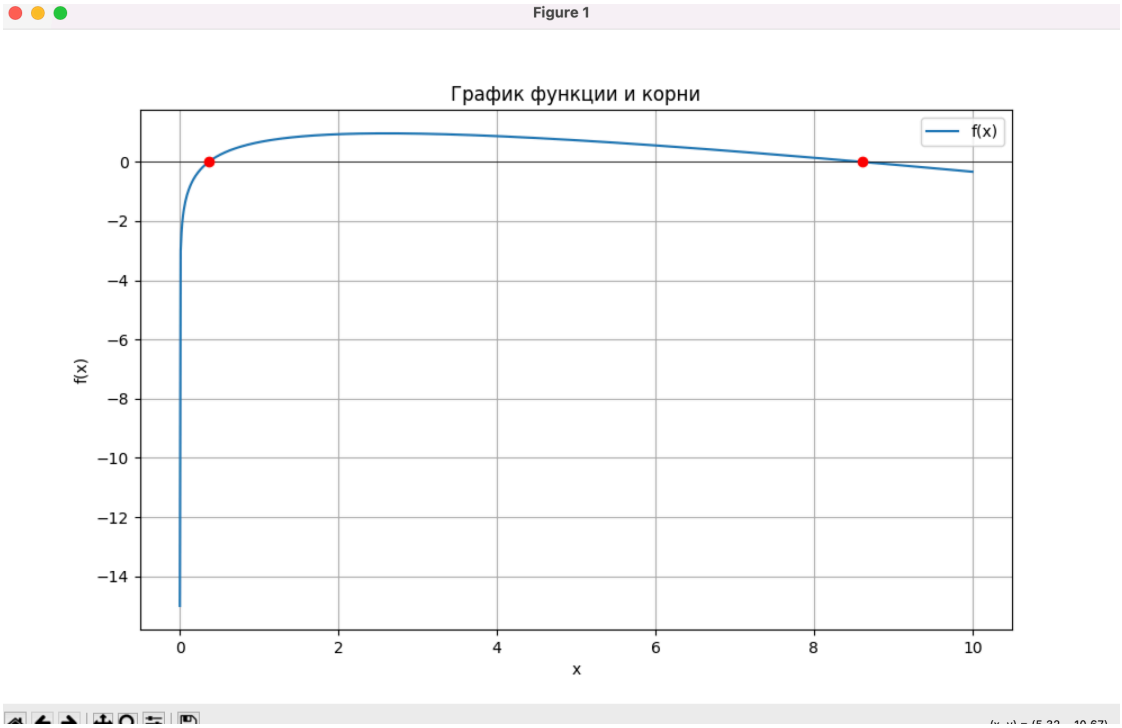

Рисунок 9 – Вывод графика функции с отображенными корнями

При некорректном вводе интерфейс программного средство отображает окно с пояснением об ошибке. На рисунках 10-11 представлены примеры неправильных вводов.

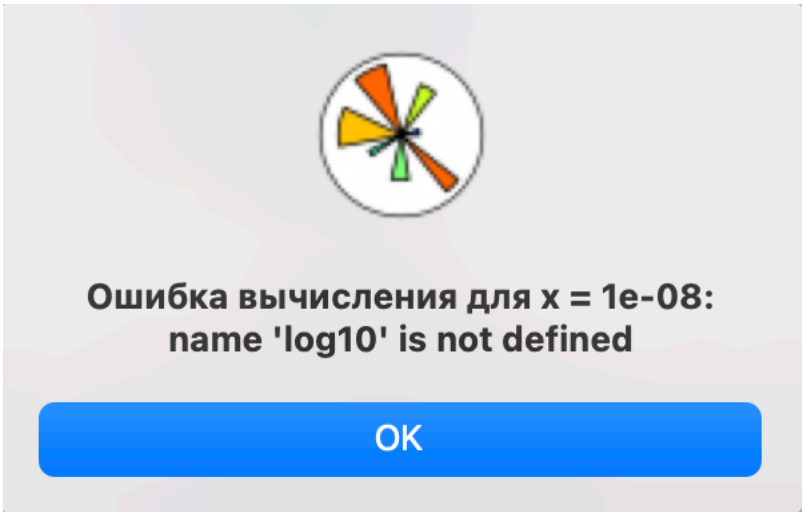

Рисунок 10 – Недопустимый синтаксис программы

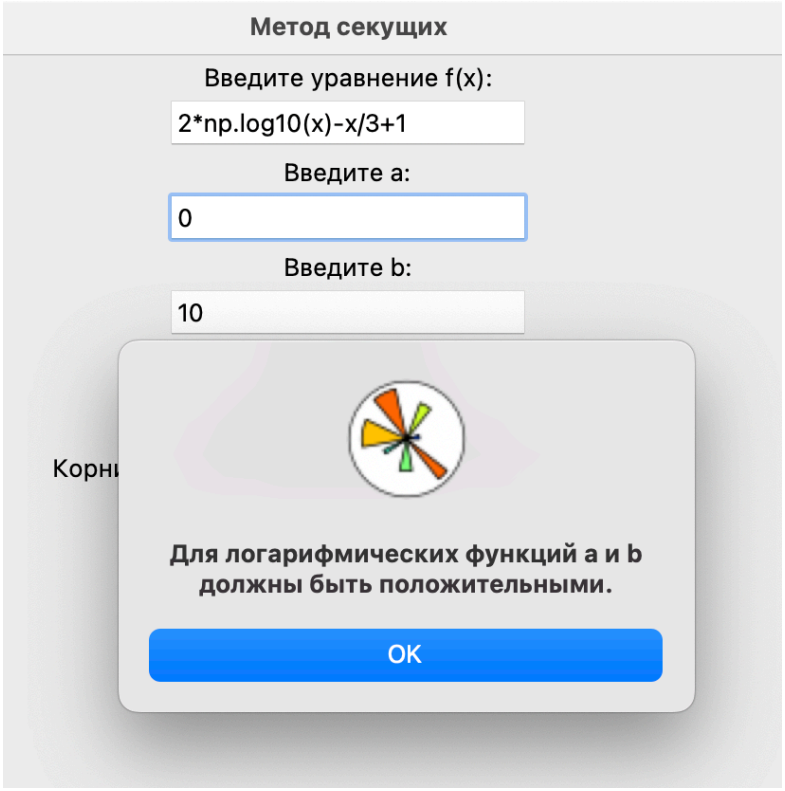

Рисунок 11 – Неверные интервалы

3.3 Описание функций программного средства

Исходный код программы содержит 7 функции, 2 из которых принадлежат к классам. Они реализуют интерфейс и метод решения нелинейных уравнений, их описание приведено в Листингах А.1 – А.7.

Функция *calculate_roots* – отвечает за получение параметров от пользователя через интерфейс (пределы интервала и функцию). Она вызывает: Проверку ограничений на допустимость логарифмических функций, нахождение всех корней на заданном интервале с помощью двушагового метода секущих, отображение корней, количество итераций и результат подстановки найденного корня $f(x)$ на интерфейсе, вызов функции для построения графика **с** отображением найденных корней, если в ходе вычислений возникает ошибка (например, превышение количества итераций), она отображается в виде всплывающего окна.

Функция *plot_function* – строит график заданной функции на интервале $[a, b]$, выделяет точки, соответствующие найденным корням, и выводит значение $f(x)$ от найденного корня. График отображается с помощью

библиотеки Matplotlib. Если при вычислении значения функции возникает ошибка, она выводится в интерфейсе с помощью messagebox.

Функция *plot_only_function* - используется для построения графика функции на интервале $[a, b]$, который вводит пользователь, без нахождения корней.

Функция *find_all_roots* - ищет все корни на интервале $[a, b]$, разбивая его на подынтервалы. Для каждого подынтервала выполняется: Проверка смены знака функции, чтобы обнаружить корень. Вычисление корня методом секущих с начальными значениями на концах подынтервала. Исключение повторяющихся корней с учетом заданной погрешности. Функция возвращает список найденных корней для дальнейшего отображения на интерфейсе.

Функция *check_log_conditions* - проверяет, содержится ли в уравнении логарифмическая функция (log, ln), и следит за тем, чтобы пределы a и b были положительными (так как логарифмы определены только для положительных значений). При нарушении условия выбрасывается ошибка с пояснением.

Класс *NonlinearEquation* - отвечает за создание объекта нелинейного уравнения на основе строки, введенной пользователем. Он использует:

- Функцию eval для вычисления выражений, таких как np.log10(x) или x**2.
- Метод evaluate для вычисления значения функции в конкретных точках.
- В случае ошибок в строке уравнения или при попытке вычислений, выбрасывается соответствующая ошибка.

Класс *OneStepSecantMethod* - реализует двушаговый метод секущих. Основной метод find_root находит корень на указанном подынтервале с начальными приближениями $x0$ и $x1$. Алгоритм выполняется до тех пор, пока:

- Абсолютное значение функции в точке корня не станет меньше заданной точности.
- Не будет превышено максимальное количество итераций. Если значения функции в начальных точках совпадают, выбрасывается

ошибка, чтобы избежать деления на ноль. В случае достижения предела итераций также выводится сообщение об ошибке.

Весь интерфейс программы создается с использованием библиотеки Tkinter. Окно программного средства содержит различные элементы, такие как текст (Label), поля ввода значений интервала (Entry), кнопки для запуска вычислений и отображения графика (Button) и для вывода сообщений об ошибках (messegebox), которые позволяют пользователю вводить параметры вычислений.

После ввода параметров и нажатия кнопки "Найти корни" программа выполняет необходимые вычисления, отображает результаты на экране. В случае нажатия кнопки "Показать график функции" строит график функции, без поиска корней.

3.4 Выводы по главе

В данной главе подробно рассмотрен и обоснован выбор языка программирования и среды разработки, а также детально описаны функции и классы программы. Созданное программное средство предназначено для решения нелинейных уравнений с использованием двушагового метода секущих. Программа не только выводит решения нелинейного уравнения, количество итераций и значения функции, а также визуализирует график функции с найденными корнями уравнения.

4 Проверка работоспособности программного средства

В данном разделе представлены примеры вычисления значений различных нелинейных уравнений с использованием двушагового метода секущих. Полученные результаты сопоставлены с точными решениями, вычисленными с помощью WolframAlpha – базы знаний и вопросно-ответной системы, включающей в себя набор вычислительных алгоритмов.

4.1 Результат работы программного средства

Для проверки корректной работы программы рассматривается решение нескольких нелинейных уравнений. Вариант 11: нелинейное уравнение $2lgx - x/3 + 1 = 0$ с интервалом $[0.0001;\ 10]$.

На рисунке 12 показан интерфейс программного средства после вычисления значения нелинейного уравнения двушаговым методом секущих.

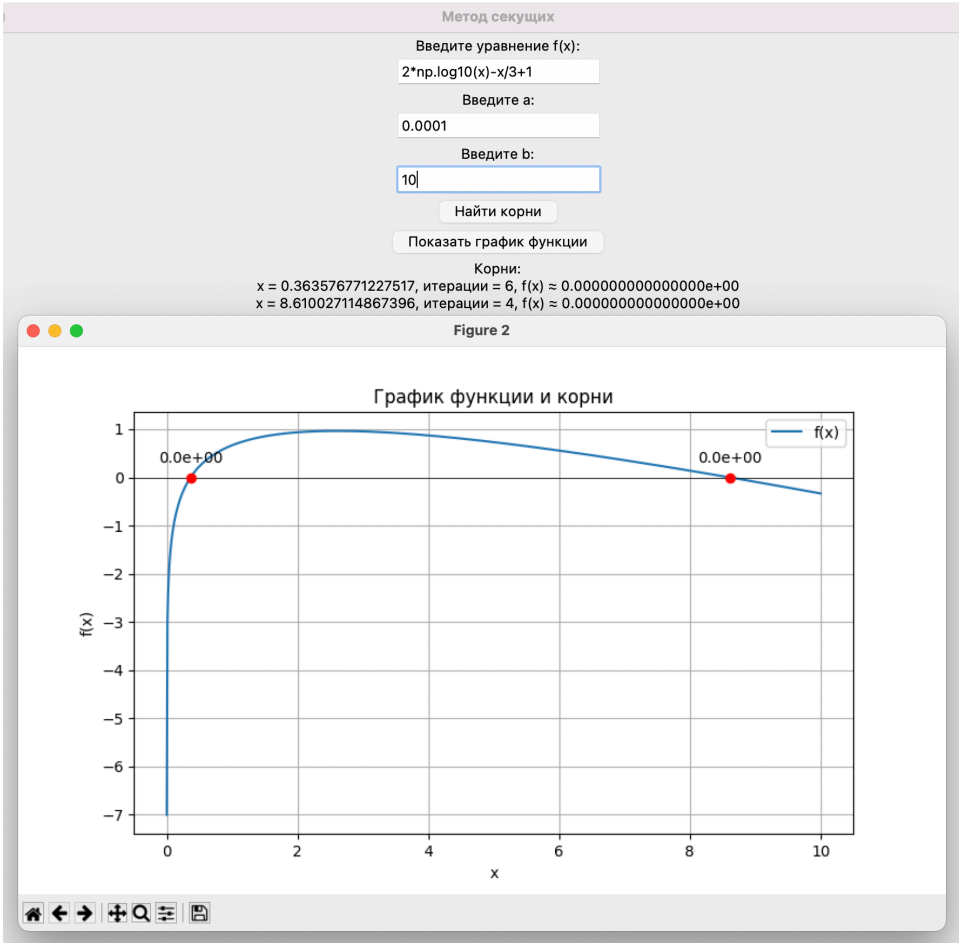

Рисунок 12 – Результат решения варианта 11 методом секущих

На рисунке 13 приведено решение, полученное с помощью WolframAlpha.

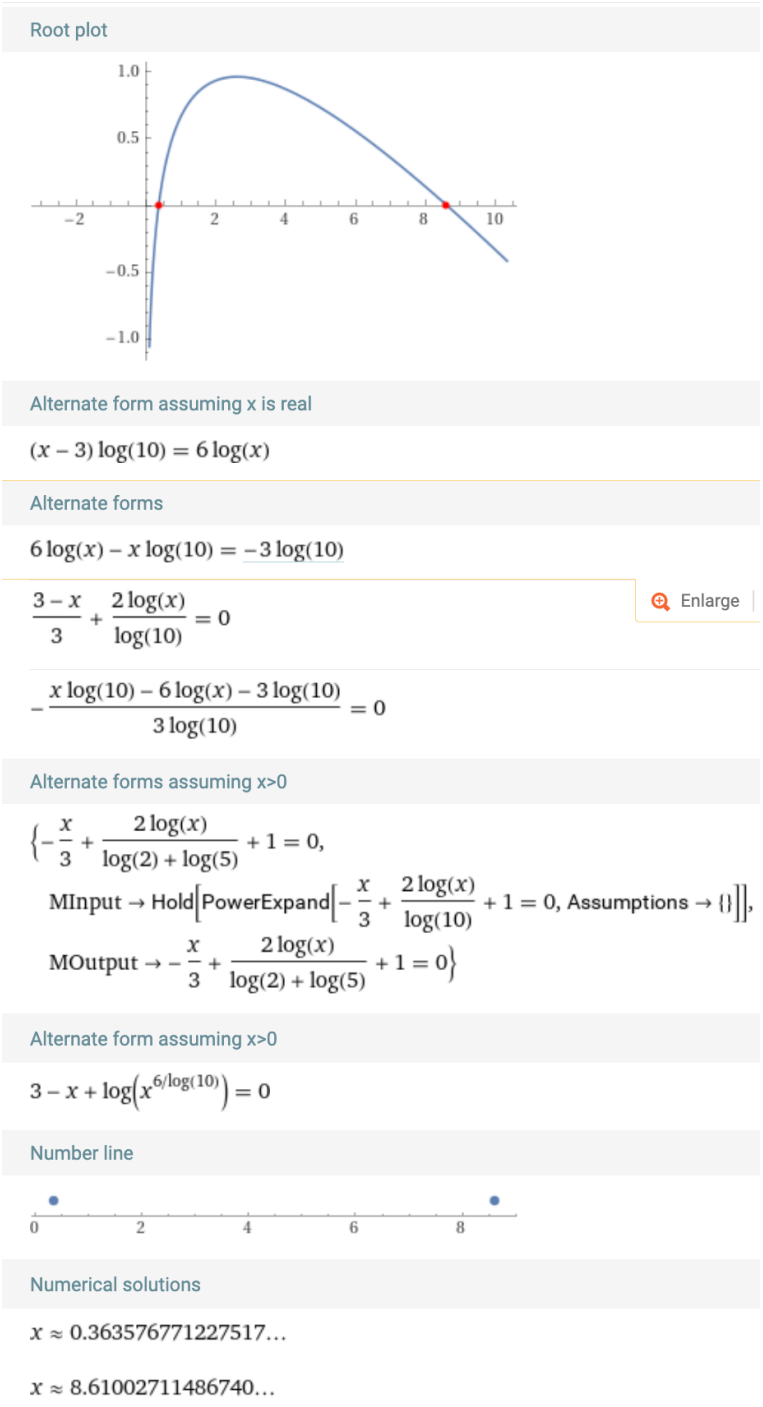

Рисунок 13 – Решение, полученное в WolframAlpha для варианта 11

Результат решения нелинейного уравнения, полученный с помощью программного средства, варианта 11 совпадает с результатом решения системы WolframAlpha.

Рассматривается решение нелинейного уравнения варианта 31: уравнение $x^2 - 5 + 0{,}4^{2x} = 0$ на интервале $[-2; 3]$.

На рисунке 14 показан интерфейс программного средства после решения нелинейного уравнения методом секущих.

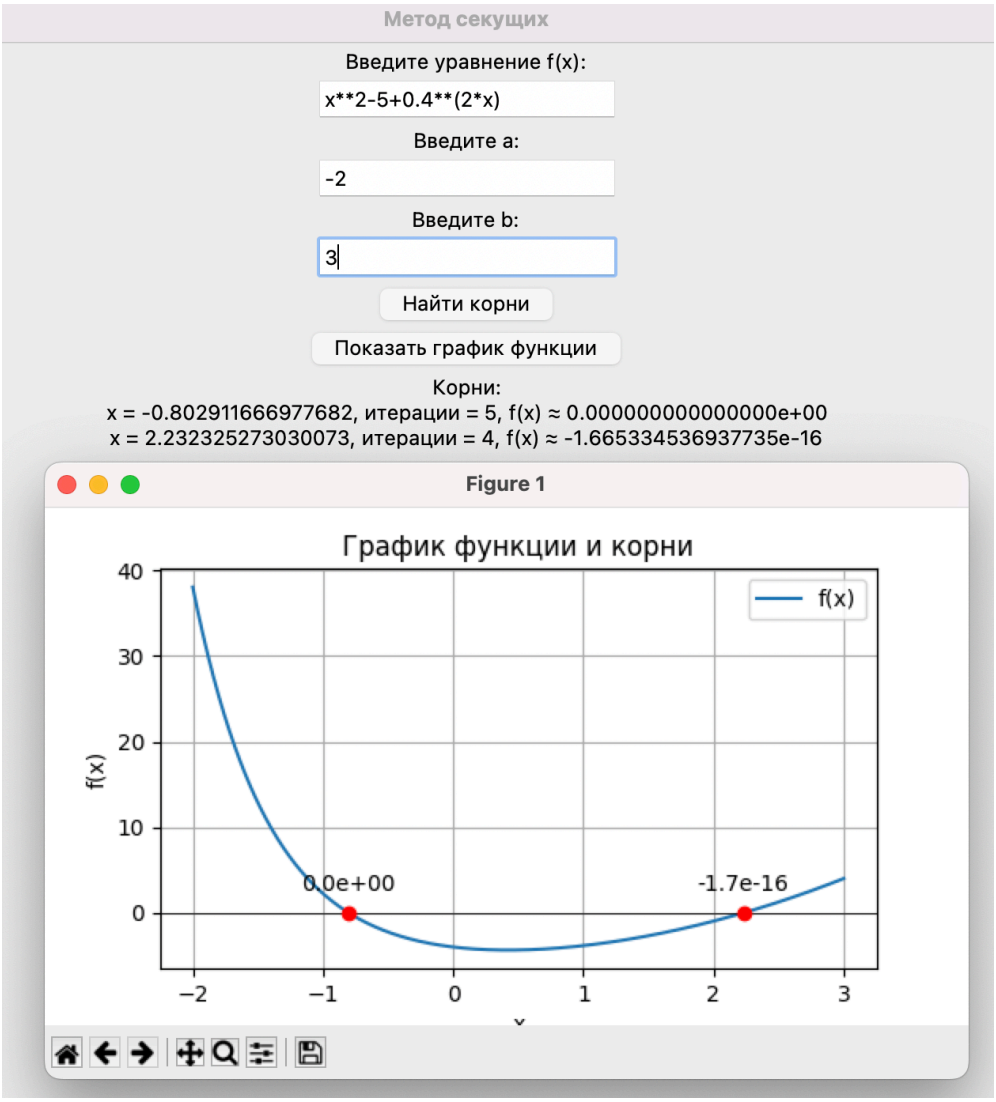

Рисунок 14 – Результат решения варианта 31 методом секущих

На рисунке 15 приведено решение, полученное в WolframAlpha.

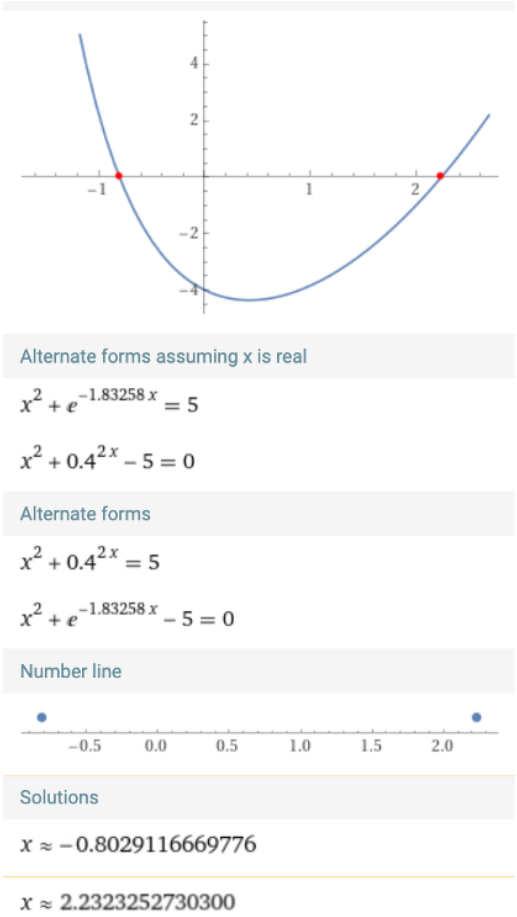

Рисунок 15 – Решение, полученное в WolframAlpha для варианта 31

Результат решения нелинейного уравнения, полученный с помощью программного средства, варианта 31 совпадает с результатом решения системы WolframAlpha.

4.2 Выводы по главе

В данной главе представлены результаты работы программного средства, подтверждающие корректность и точность вычислений.

Заключение

В рамках работы над учебной практикой проведен теоретический обзор численных методов решения нелинейных уравнений, более подробно душагового метода секущих. Этот метод, основанный на последовательном линейном приближении, демонстрирует высокую эффективность при решении уравнений, где нет возможности аналитического нахождения корней.

Проведено алгоритмическое конструирование программного средства. Метод секущих реализован на языке Python в среде разработки PyCharm. Разработанная программа протестирована на различных примерах, результаты вычислений сравнены с точными значениями, вычисленными с помощью системы WolframAlpha. Вычислительные данные представлены в графическом виде для более наглядного представления сходимости метода и демонстрации его эффективности.

Реализованное программное средство может использоваться в учебных целях в дисциплинах, связанных с численными методами для решения нелинейных уравнений в курсах вычислительной математики.

Перечень использованных информационных ресурсов

1. Гурьев Е. К. Решение нелинейных уравнений. Методические указания к лабораторной и курсовой работам по курсам «Вычислительная математика» и «Информатика. Раздел «Численные методы» — Текст: электронный. URL: https://rstu.ru/metods/books/gur2.pdf
2. Онлайн база знаний и набор вычислительных алгоритмов, вопросно-ответная система WolframAlpha [Электронный ресурс] URL: https://www.wolframalpha.com/
3. Download Python [Электронный ресурс] URL: https://www.python.org/downloads
4. Онлайн калькулятор по направлениям для проверок решения по математическим дисциплинам. [Электронный ресурс] URL: https://math.semestr.ru/
5. Титов, А.Н. Решение задач линейной алгебры и прикладной математики в Python. Работа с библиотекой SciPy : учеб.-метод. пособие / Р.Ф. Тазиева; Казан. нац. исслед. технол. ун-т; А.Н. Титов . — Казань : КНИТУ, 2023 .— 124 с. — ISBN 978-5-7882-3319-2. URL: https://rucont.ru/efd/870401
6. Златопольский, Д. М. Основы программирования на языке Python : учебник / Д. М. Златопольский. - 2-е изд. - Москва : ДМК Пресс, 2018. - 396 с. - ISBN 978-5-97060-641-4. - Текст : электронный. URL: https://znanium.com/catalog/product/2012512
7. Биллиг, В. А. Программирование на Python : краткий курс / В. А. Биллиг. - Москва : ИНТУИТ, 2016. - 265 с. - Текст : электронный. URL: https://znanium.ru/catalog/product/2156683

Приложение А Исходный код программного средства

Листинг А.1 – Класс *NonLinearEquation* для представления нелинейного уравнения и класс *OneStepSecantMethod* для поиска корней методом секущих

```
import numpy as np
class NonlinearEquation:
    def __init__(self, function_str: str):
        self.function_str = function_str
        try:
            self.function = lambda x: eval(function_str, {"x":
x, "np": np})
        except Exception as e:
            raise ValueError(f"Ошибка в уравнении: {e}")

    def evaluate(self, x: np.ndarray) -> np.ndarray:
        try:
            return self.function(x)
        except Exception as e:
            raise ValueError(f"Ошибка вычисления для x = {x}:
{e}")

class OneStepSecantMethod:
    def __init__(self, equation: NonlinearEquation):
        self.equation = equation

    def find_root(self, x0: float, x1: float, tolerance: float =
1e-15, max_iterations: int = 100):
        f_x0, f_x1 = self.equation.evaluate(x0),
self.equation.evaluate(x1)
        iteration = 0

        while abs(f_x1) > tolerance and iteration <
max_iterations:
            if f_x1 == f_x0:
                raise ValueError("Начальные значения функции
совпадают.")
```

```
            x2 = x1 - f_x1 * (x1 - x0) / (f_x1 - f_x0)
            x0, x1, f_x0, f_x1 = x1, x2, f_x1,
self.equation.evaluate(x2)
            iteration += 1

        if iteration == max_iterations:
            raise ValueError("Превышено максимальное количество
итераций.")

        return x1, iteration, f_x1
```

Листинг А.2 – функция *plot_function* для построения графика функции

```
import matplotlib.pyplot as plt

def plot_function(equation: NonlinearEquation, a: float, b:
float, roots=None, num_points: int = 1000):
    x = np.linspace(a, b, num_points)
    try:
        y = equation.evaluate(x)
    except ValueError as e:
        messagebox.showerror("Ошибка", str(e))
        return

    plt.figure(figsize=(10, 6))
    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='black', linewidth=0.5)

    if roots:
        for root, _, f_root in roots:
            plt.scatter(root, 0, color='red', zorder=5)
            plt.annotate(f"{f_root:.1e}", (root, 0),
textcoords="offset points", xytext=(0,10), ha='center')

    plt.grid(True)
    plt.legend()
    plt.xlabel('x')
```

```
    plt.ylabel('f(x)')
    plt.title('График функции и корни')
    plt.show()
```

Листинг А.3 – Вспомогательная функция *check_log_conditions* для проверки корректности входных данных для логарифмических функций

```
def check_log_conditions(function_str: str, a: float, b: float):
    if any(log in function_str for log in ['log', 'ln']) and (a
<= 0 or b <= 0):
        raise ValueError("Для логарифмических функций a и b
должны быть положительными.")
```

Листинг А.4 – Функция *find_all_roots* для нахождения всех корней функции на заданном интервале

```
def find_all_roots(equation: NonlinearEquation, a: float, b:
float, subintervals: int = 100, tolerance: float = 1e-15):
    roots = []
    x_vals = np.linspace(a, b, subintervals)
    secant = OneStepSecantMethod(equation)

    for i in range(len(x_vals) - 1):
        x0, x1 = x_vals[i], x_vals[i + 1]
        if equation.evaluate(x0) * equation.evaluate(x1) < 0:
            try:
                root, iterations, f_root = secant.find_root(x0,
x1, tolerance)
                if abs(f_root) < tolerance:
                    if not any(abs(root - r[0]) < tolerance for
r in roots):
                        roots.append((root, iterations, f_root))
            except ValueError:
                continue
    return roots
```

Листинг А.5 – Функция *calculate_roots* для расчета и отображения корней, количества итераций и значения $f(x)$ при подстановке в нее корня

```
import tkinter as tk
from tkinter import messagebox

def calculate_roots():
    try:
        a, b = float(entry_a.get()), float(entry_b.get())
        function_str = entry_function.get()

        check_log_conditions(function_str, a, b)

        equation = NonlinearEquation(function_str)
        roots = find_all_roots(equation, a, b)

        result_text = "Корни:\n"
        for root, iterations, f_root in roots:
            result_text += f"x = {root:.15f}, итерации =
{iterations}, f(x) ≈ {f_root:.15e}\n"

        result_label.config(text=result_text)
        plot_function(equation, a, b, roots)
    except ValueError as e:
        messagebox.showerror("Ошибка", str(e))
```

Листинг А.6 – Функция *plot_only_function* для отображения графика функции

```
def plot_only_function():
    try:
        a, b = float(entry_a.get()), float(entry_b.get())
        function_str = entry_function.get()

        check_log_conditions(function_str, a, b)

        equation = NonlinearEquation(function_str)
        plot_function(equation, a, b)
    except ValueError as e:
        messagebox.showerror("Ошибка", str(e))
```

Листинг А.7 – Интерфейс пользователя для работы с методом секущих

```
import tkinter as tk

root_window = tk.Tk()
root_window.title("Метод секущих")

tk.Label(root_window, text="Введите уравнение f(x):").pack()
entry_function = tk.Entry(root_window)
entry_function.pack()

tk.Label(root_window, text="Введите a:").pack()
entry_a = tk.Entry(root_window)
entry_a.pack()

tk.Label(root_window, text="Введите b:").pack()
entry_b = tk.Entry(root_window)
entry_b.pack()

tk.Button(root_window, text="Найти корни",
command=calculate_roots).pack()
tk.Button(root_window, text="Показать график функции",
command=plot_only_function).pack()

result_label = tk.Label(root_window, text="")
result_label.pack()

root_window.mainloop()
```