Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»
Высшая школа технологии и энергетики
Кафедра прикладной математики и информатики

ПРОГРАММИРОВАНИЕ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ В ЭЛЕКТРОЭНЕРГЕТИКЕ

РЕАЛИЗАЦИЯ ЧИСЛЕННЫХ МЕТОДОВ НА ЯЗЫКЕ PYTHON

Выполнение контрольной работы

Методические указания для студентов заочной формы обучения (срок обучения 3 года 6 месяцев) по направлению подготовки 13.03.02 — Электроэнергетика и электротехника

Составитель
А. Н. Маслобоев

Санкт-Петербург
2025

Утверждено
на заседании кафедры ПМИ
26.06.2025 г., протокол № 10

Рецензент В. И. Сидельников

Методические указания соответствуют программам и учебным планам дисциплины «Программирование на языках высокого уровня в электроэнергетике» для студентов, обучающихся по направлению подготовки 13.03.02 «Электроэнергетика и электротехника». В указаниях представлен порядок выполнения и оформления контрольной работы, приведен список заданий для выполнения контрольной работы.

Методические указания предназначены для студентов заочной формы обучения (срок обучения 3 года 6 месяцев).

Утверждено Редакционно-издательским советом ВШТЭ СПбГУПТД в качестве методических указаний

Режим доступа: http://publish.sutd.ru/tp_get_file.php?id=202016, по паролю.
- Загл. с экрана.
Дата подписания к использованию 09.09.2025 г. Рег.№ 5284/25

Высшая школа технологии и энергетики СПбГУПТД
198095, СПб., ул. Ивана Черных, 4.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель дисциплины «Программирование на языках высокого уровня в электроэнергетике» – сформировать компетенции обучающегося в области современных технологий высокоуровневого программирования.

В практическом аспекте данной дисциплины студент должен:

- изучить основные принципы работы в интегрированной среде разработки программного обеспечения IDLE Python;
- освоить базовые принципы и конструкции языка Python, знание которых необходимо для разработки приложений на этом языке;
- получить практические навыки разработки приложений для применения их в области электротехники и электроэнергетики.

Методические указания разработаны в соответствии с программой курса «Программирование на языках высокого уровня в электроэнергетике» Федерального государственного образовательного стандарта для бакалавров по направлению подготовки 13.03.02 «Электроэнергетика и электротехника».

Работа состоит из введения, разделов, описывающих порядок выполнения контрольной работы и представления ее результатов, требований к оформлению контрольной работы, пяти практических заданий, заключения, библиографического списка и приложения.

ПОРЯДОК ВЫПОЛНЕНИЯ КОНТРОЛЬНОЙ РАБОТЫ И ПРЕДСТАВЛЕНИЯ РЕЗУЛЬТАТОВ

Учебная дисциплина «Высокоуровневые методы программирования в электроэнергетике», в рамках освоения которой выполняется данная контрольная работа, изучается на третьем курсе студентами, проходящими заочное обучение по сокращенной форме (срок обучения 3 года и 6 месяцев) по направлению подготовки 13.03.02 «Электроэнергетика и электротехника».

Целью контрольной работы является изучение численных методов, используемых для решения профессиональных задач в области электроэнергетики и способов реализации этих методов в виде программ, разработанных на языке Python в среде программирования IDLE Python.

Контрольная работа выполняется на мультипарадигменном высокоуровневом языке программирования Python в интегрированной среде разработки и обучения IDLE. Название IDLE представляет собой аббревиатуру полного названия данной среды Integrated Development and Learning Environment. Установка данной среды на компьютерах обучающихся не должна вызвать никаких затруднений, поскольку IDLE относится к категории свободного программного обеспечения. Скачать дистрибутив интегрированной среды IDLE можно на официальном сайте разработчиков языка Python, который находится во всемирной сети по адресу **python.org**.

Перед выполнением контрольной работы обучающемуся следует получить базовые знания по программированию на высокоуровневом языке Python, для чего рекомендуется использовать учебные пособия, которые приведены в библиографическом списке, приложенном к данным методическим указаниями под номерами 2, 5 и 7.

Практические задания для выполнения контрольной работы по вариантам представлены в данных методических указаниях в разделе «Содержание контрольной работы».

После выполнения практических заданий обучающийся должен представить готовую контрольную работу преподавателю как в электронном виде, так и в письменном. Письменный вариант контрольной работы следует распечатать на принтере и сброшюровать.

Первым листом контрольной работы должен быть титульный лист. На титульном листе обучающемуся необходимо указать свою фамилию и инициалы, номер группы и номер своего варианта задания, фамилию и инициалы преподавателя, которому студент сдает контрольную работу. Образец оформления титульного листа представлен в приложении к данным методическим указаниям.

Номер варианта индивидуального задания студент определяет по двум последним цифрам номера своей зачетной книжки в соответствии с представленной ниже таблицей 1.

Таблица 1 – Определение номера варианта задания

Последняя цифра номера зачетной книжки	Варианты заданий по предпоследней цифре номера зачетной книжки	
	Четная цифра	Нечетная цифра
0	1	11
1	2	12
2	3	13
3	4	14
4	5	15
5	6	16
6	7	17
7	8	18
8	9	19
9	10	20

Например, при номере зачетной книжки 220-686 нужно выполнить задание для варианта № 7, а при номере 220-713 – задание для варианта № 14.

После титульного листа в контрольной работе должно быть представлено выполнение всех пяти практических заданий в соответствии с вариантом. Для каждого задания необходимо указать последовательность действий, привести расчетные формулы, исходный код программы на языке Python и результаты выполнения этой программы. Результаты должны быть представлены в виде копии экрана, которую следует оформить как иллюстрацию в соответствии с требованиями, приведенными в следующем разделе методических указаний.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

Текст контрольной работы должен быть распечатан на принтере на одной стороне стандартного листа формата А4. Также работа должна быть представлена в электронном виде, при этом все требования, касающиеся ее оформления должны быть соблюдены точно таким же образом, как и для печатного варианта.

Приведем далее основные требования к оформлению. Текст контрольной работы должен быть набран в текстовом процессоре Microsoft Word шрифтом с кеглем 14 пунктов с полуторным межстрочным интервалом. Заголовки разделов контрольной работы должны быть набраны полужирным шрифтом. Для основного текста работы рекомендуется использовать шрифт с засечками (например, Times New Roman), а для заголовков лучше применить шрифт без засечек (например, Arial). Для программного кода, приводимого в работе, следует использовать один из моноширинных шрифтов (например, Courier New). Шрифт не должен содержать различные спецэффекты такие, как использование тени, дополнительного контура, имитация объемного текста и т.п. Цвет всех букв, цифр и прочих символов должен быть черным.

Между соседними словами нужно ставить только один пробел. Между знаками препинания (точкой, запятой, точкой с запятой, двоеточием) и предшествующим им словом не должно быть пробела. Пробел должен находится после знака препинания. В тексте следует различать дефис и тире. Дефис используется при образовании сложных слов, обозначении диапазонов чисел. В других случаях (например, для отделения друг от друга частей предложения) используется тире. Для его набора можно использовать комбинацию клавши Ctrl + Alt + «минус» на дополнительной цифровой клавиатуре. В отличие от других знаков препинания, при использовании тире пробел должен стоять и до, и после этого знака.

Заголовки должны быть выравнены по центру, а основной текст – по ширине. В начале каждого абзаца (кроме заголовков) должна быть красная строка размером 1,25 см. Не допускается формирование красной строки с помощью пробелов и табуляции. Для ее создания следует использовать в Word на вкладке «Главная» раздел «Абзац». Необходимо не допускать наличия в документе так называемых висячих строк. Это означает, что в конце страницы не должно быть одной строки, оторванной от абзаца, находящегося на следующей странице, а также нельзя допускать, чтобы на следующую страницу переносилась только одна строка от абзаца, который не помещается на данной странице целиком. Если какой-либо абзац целиком не помещается на текущей странице, то на следующую страницу должны переноситься, как минимум, две строки из этого абзаца. Так же, если на текущей странице помещается только начало абзаца, то на этой странице должно оставаться не менее двух строк данного абзаца. Если выполнить данное условие не получается, то лучше перенести абзац целиком на следующую страницу.

Каждый раздел (выполненное задание) контрольной работы должен начинаться с новой страницы. Поля страницы контрольной работы должны иметь следующие размеры: левое поле – 30 мм, правое – 10 мм, верхнее и нижнее – 20 мм. Ориентация страницы должна быть книжной. В курсовой работе следует использовать сквозную нумерацию страниц. Номера проставляются, начиная со страницы №2 – оглавления (на титульном листе номер проставлять не надо). Далее нумеруются все страницы. Номер страницы должен находиться внизу страницы (в ее нижнем колонтитуле) и должен быть выравнен по центру.

На все иллюстрации, имеющиеся в контрольной работе, должны быть сделаны ссылки в тексте пояснительной записки. Сами иллюстрации должны располагаться на той же странице, что и ссылки на них или, если это невозможно, максимально близко к ссылкам. Ко всем иллюстрациям, имеющимся в контрольной работе, должны быть сделаны подписи. Подпись отделяется одной строкой от рисунка и выравнивается по центру. Подпись должна начинаться со слова «рисунок», затем указывается его номер и далее после тире поясняется содержание рисунка. Все иллюстрации должны быть обязательно пронумерованы.

Имеющиеся в тексте контрольной работы уравнения и формулы следует помещать на отдельную строку. Для формул используется сквозная нумерация, а сам номер заключается в круглые скобки и выравнивается по правому краю листа. Нумеровать необходимо только те формулы, ссылки на которые имеются в тексте контрольной работы. Расшифровка значений символов, используемых в формуле, приводится сразу после формулы, причем в том же порядке, в котором они написаны в самой формуле. Первая строка этой расшифровки начинается со слова «где» без двоеточия.

Готовая контрольная работа обязательно должна быть передана преподавателю в сброшюрованном виде. Работа, представляющая собой набор листов, не скреплённых между собой, приниматься не будет.

СОДЕРЖАНИЕ КОНТРОЛЬНОЙ РАБОТЫ

Практическое задание № 1. Системы линейных уравнений

Математическое описание установившегося режима электрической сети обычно представляется в виде системы уравнений узловых балансов токов (или мощностей). Известными в этих уравнениях являются проводимости ветвей и мощности нагрузок всех узлов (кроме балансирующего), а неизвестными переменными – величины узловых напряжений. При определенном допущении нагрузки узлов могут быть рассчитаны через мощности и представлены в виде комплексных значений задающих токов. В этом случае система уравнений установившегося режима становится линейной

Системой линейных уравнений называется набор уравнений следующего вида:

$$a_{11}x_1 + a_{12}x_2 + \dots a_{1n}x_n = \mathrm{b}_1$$
$$a_{21}x_1 + a_{22}x_2 + \dots a_{2n}x_n = \mathrm{b}_2$$
$$\dots\dots\dots\dots\dots\dots\dots\dots$$
$$a_{\mathrm{n}1}x_1 + a_{n2}x_2 + \dots a_{nn}\mathrm{x}_n = \mathrm{b}_n$$

Одним из наиболее распространенных способов решения систем линейных уравнений является метод исключения переменных, называемый также методом Гаусса. Рассмотрим его на примере системы из 3 уравнений с 3 неизвестными.

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned}\right\}$$

Для решения данной системы в начале делим первое уравнение на коэффициент a_{11}. В результате преобразуем первое уравнение к следующему виду:

$$x_1 + a_{12}^{(1)} + a_{13}^{(1)}x_3 = b_1^{(1)},$$

где $a_{12}^{(1)} = \frac{a_{12}}{a_{11}}$, $a_{13}^{(1)} = \frac{a_{13}}{a_{11}}$, $b_1^{(1)} = \frac{b_1}{a_{11}}$, затем из левой и правой частей второго уравнения системы вычитаем соответственно левую и правую части первого уравнения, умноженные на коэффициент a_{21}. В результате второе уравнение системы преобразуется к следующему виду:

$$a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 = b_2^{(1)},$$

где $a_{22}^{(1)} = a_{22} - a_{12}^{(1)}a_{21}$, $a_{23}^{(1)} = a_{23} - a_{13}^{(1)}a_{21}$, $b_2^{(1)} = b_2 - b_1^{(1)}a_{21}$, то есть из второго уравнения исключается первое неизвестное.

Аналогичным образом поступаем с третьим уравнением системы, то есть из обеих частей этого уравнения вычитаем соответствующие части первого уравнения, умноженные на коэффициент a_{31}. Таким образом, из третьего уравнения также исключается неизвестное x_1. Третье уравнение преобразуется к виду:

$$a_{32}^{(1)} x_2 + a_{33}^{(1)} x_3 = b_3^{(1)},$$

где $a_{32}^{(1)} = a_{32} - a_{12}^{(1)} a_{31}, a_{33}^{(1)} = a_{33} - a_{13}^{(1)} a_{31}, b_3^{(1)} = b_1^{(1)} a_{31}.$

Затем второе уравнение системы делится на коэффициент $a_{22}^{(1)}$. Тогда второе уравнение можно привести к следующему виду:

$$x_2 + a_{23}^{(2)} x_3 = b_2^{(2)},$$

где $a_{23}^{(2)} = \frac{a_{23}^{(1)}}{a_{22}^{(1)}}, b_2^{(2)} = \frac{b_2^{(1)}}{a_{22}^{(1)}}.$

Теперь из обеих частей третьего уравнения вычитаем соответствующие части второго уравнения, умноженные на коэффициент, находящийся в третьем уравнении при x_2, то есть на $a_{32}^{(1)}$. Результатом данного вычитания будет следующее равенство:

$$a_{33}^{(2)} x_3 = b_3^{(2)},$$

где $a_{33}^{(2)} = a_{33}^{(1)} - a_{23}^{(2)} a_{32}^{(1)}, b_3^{(2)} = b_3^{(1)} - b_2^{(1)} a_{32}^{(1)}.$

Разделив обе части равенства на коэффициент $a_{33}^{(2)}$, получаем следующее соотношение:

$$x_3 = b_3^{(3)},$$

где $b_3^{(3)} = \frac{b_3^{(2)}}{a_{33}^{(2)}}.$

Таким образом, мы нашли одно из неизвестных системы уравнений. Последовательность действий, приведшая к нахождению данного неизвестного, называется прямым ходом решения системы линейных уравнений методом Гаусса.

Для нахождения оставшихся неизвестных в системе уравнений требуется исключить из предыдущих уравнений в начале неизвестную x_3, а затем x_2. Последовательность действий, требуемая для их исключения, называется обратным ходом решения системы уравнений методом Гаусса. В нашем примере для того, чтобы исключить из второго уравнения x_3, необходимо умножить обе части третьего уравнения на коэффициент при x_3 во втором уравнении и вычесть получившиеся величины из обеих частей второго уравнения. Тогда во втором уравнении останется только x_2, что позволит без труда определить ее величину. Затем останется умножить обе части второго уравнения на коэффициент при x_2 в первом уравнении и вычесть второе уравнение из первого. Таким образом, мы найдем последнюю неизвестную величину x_1.

Описанный выше метод решения может быть применен для решения любой линейной системы n уравнений с n неизвестными за исключением того варианта, при котором в каком-либо из уравнений все коэффициенты при неизвестных равны нулю, а свободный член в этом уравнении не равен нулю (в этом случае уравнение вообще не имеет решения). Однако метод Гаусса в некоторых случаях дает значительную погрешность при вычислении неизвестных, поэтому был разработан ряд модификаций данного метода,

наиболее распространенной из которых является метод Гаусса с выбором главного элемента в столбце.

Суть этого метода сводится к следующему: в начале вычислений при прямом ходе среди коэффициентов при неизвестной x_1 находится максимальный по модулю. Если этот коэффициент находится не в первом уравнении (первой строке системы), а в какой-либо *k*-й строке, то в системе уравнений первая и *k*-я строка меняются местами. Далее вычисления производятся по описанной выше схеме.

Затем на следующем этапе вычислений находят максимальный коэффициент при неизвестной x_2, причем поиск ведем, только начиная со второй строки системы уравнений. Та строка системы, в которой найден максимальный коэффициент, меняется местами со второй, и так далее вплоть до завершения прямого хода метода Гаусса.

Ниже приводится текст программы, которая решает систему линейных уравнений методом Гаусса.

```
    # Подключение библиотеки NumPy
    import numpy as np
    import sys

    # Определение количества неизвестных
    n = int(input('Введите количество неизвестных: '))

    # Создание двумерного массива n * n+1  для хранения
расширенной матрицы
    # и инициализация его нулями
    a = np.zeros((n,n+1))

    # Создание одномерного массива из n элементов для
хранения результатов
    # и присваивание элементам нулевых значений
    x = np.zeros(n)

    # Ввод коэффициентов расширенной матрицы
    print('Введите коэффициенты расширенной матрицы:')
    for i in range(n):
        for j in range(n+1):
           if j!=n:
               a[i][j] = float(input( 'a['+str(i+1)+']['+
str(j+1)+']='))
           else:
               a[i,j]=float(input('b['+str(i+1)+']='))

    # Метод Гаусса (прямой ход)
    for i in range(n):
        if a[i][i] == 0.0:
            sys.exit('Деление на ноль!')
```

```
        for j in range(i+1, n):
            ratio = a[j][i]/a[i][i]

            for k in range(n+1):
                a[j][k] = a[j][k] – ratio * a[i][k]

    # Метод Гаусса (обратный ход)
    x[n-1] = a[n-1][n]/a[n-1][n-1]

    for i in range(n-2,-1,-1):
        x[i] = a[i][n]

        for j in range(i+1,n):
            x[i] = x[i] – a[i][j]*x[j]

        x[i] = x[i]/a[i][i]

    # Вывод результатов
    print('\nНеизвестные равны: ')
    for i in range(n):
        print('X%d = %0.2f' %(i+1,x[i]), end = '\t')
```

Программа начинается с подключения библиотеки *numpy*. Данная библиотека предназначена для поддержки многомерных массивов (включая матрицы) и поддержки высокоуровневых математических функций, предназначенных для работы с многомерными массивами. Затем пользователю предлагается указать количество неизвестных в решаемой системе уравнений. Это количество присваивается переменной *n*.

Далее в программе создаются два массива. Один из них – это двумерный массив, предназначенный для хранения расширенной матрицы. Под расширенной матрицей понимается матрица, которая содержит как коэффициенты при неизвестных, так и столбец свободных членов. Матрица, состоящая только из коэффициентов при неизвестных, имеет размерность $n \cdot n$, но если матрица расширенная, то к ней добавляется еще один столбец и ее размерность становится равной n на $n+1$.

Второй массив является одномерным (вектором) и он предназначен для того, чтобы в него были помещены значения неизвестных после того, как эти неизвестные будут вычислены. Он содержит n элементов – по количеству неизвестных.

Заполнение расширенной матрицы данными производится с помощью конструкции, состоящей из двух вложенных циклов. Элементы матрицы, представляющие собой коэффициенты при неизвестных и элементы столбца свободных членов, вводятся построчно.

Затем производится вычисление значений неизвестных, которое выполняется по описанному ранее в этом разделе методических указаний методу Гаусса, который включает в себя прямой и обратный ход. После завершения вычислений найденные неизвестные выводятся из одномерного массива-вектора

на экран компьютера. Вывод полученных результатов также производится в цикле.

На рисунке1.1 приведен пример решения системы из трех уравнений с тремя неизвестными с помощью рассмотренной выше программы.

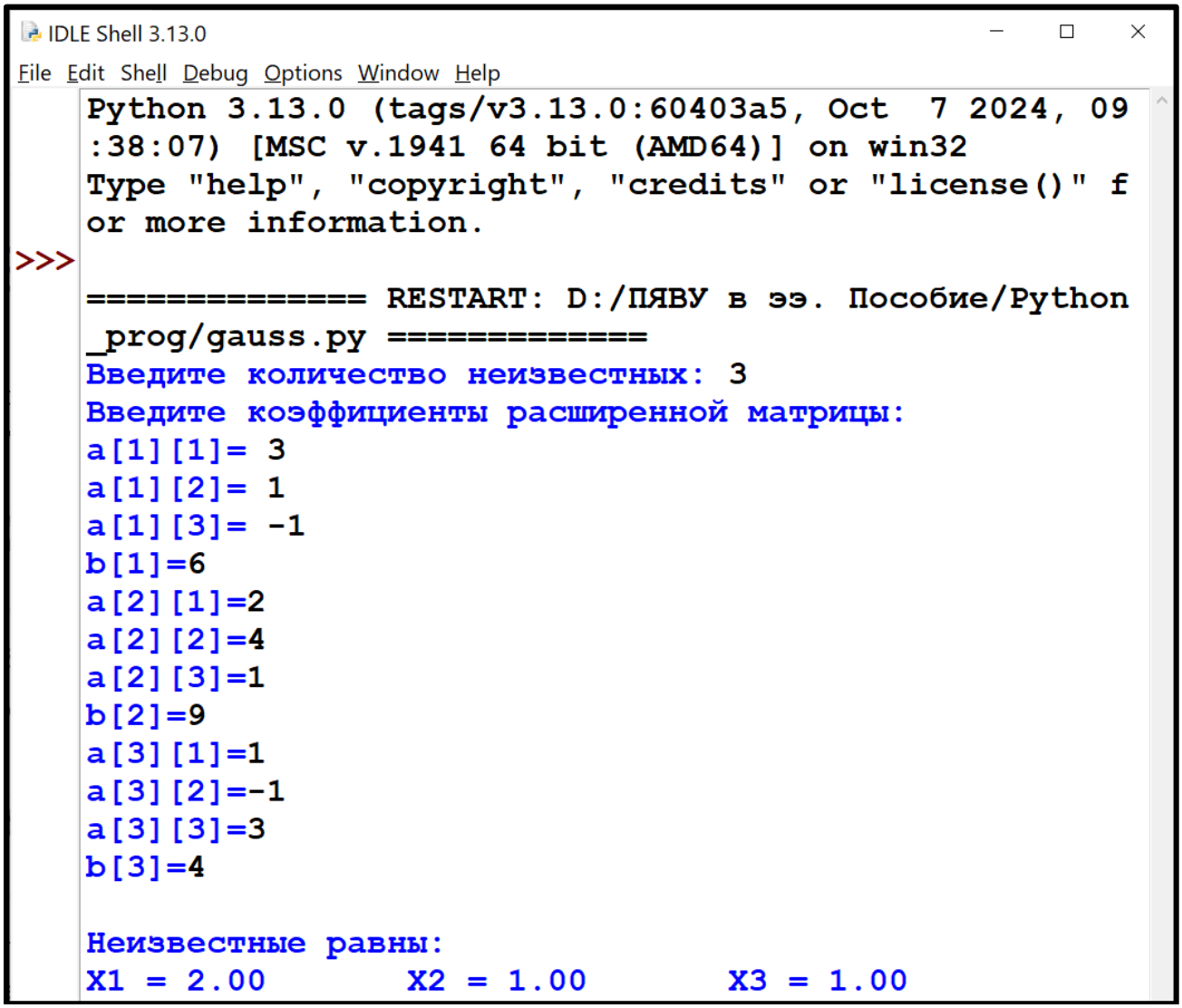

Рисунок 1.1 – Решение системы трех уравнений с тремя неизвестными

Задание

Для участка электросети (рис. 1.2) определить распределение токов в элементах схемы. Уравнения состояния электрической сети составить в матричной форме и решить методом Гаусса. Задающие токи узлов I_1, I_2, I_3 (А), напряжение базисного узла U_0 (В) и сопротивления ветвей Z_{ij} (Ом) указаны в таблице 2. Варианты выбираются по последней цифре в зачетной книжке студента.

Рисунок 1.2 – Схема участка электросети

Таблица 2 – Исходные данные к заданию

Параметры	Варианты									
	0	1	2	3	4	5	6	7	8	9
I_1	40	60	80	100	60	100	80	60	80	20
I_2	50	120	30	40	30	40	20	40	20	30
I_3	80	80	50	60	120	80	70	100	30	60
U_0	500	400	230	660	230	660	400	660	400	230
Z_{01}	0,3	0,3	0,4	0,5	0,2	0,2	0,1	0,4	0,1	0,3
Z_{02}	0,1	0,5	0,2	0,4	0,3	0,4	0,5	0,3	0,2	0,2
Z_{03}	0,1	0,2	0,1	0,5	0,5	0,5	0,2	0,5	0,1	0,5
Z_{21}	0,2	0,4	0,3	0,1	0,3	0,3	0,2	0,5	0,4	0,5
Z_{23}	0,4	0,2	0,3	0,2	0,1	0,4	0,4	0,2	0,3	0,1

Практическое задание № 2. Нелинейные уравнения

Для построения монтажных кривых в задаче механического расчета воздушных линий электропередачи необходимо многократно решать нелинейные уравнения состояния провода. Существует несколько методов решения нелинейных уравнений, но на практике, как правило, используются метод простой итерации и метод Ньютона (метод касательных). Далее рассмотрим оба этих метода.

Начнем рассмотрение этих методов с определения самого понятия нелинейного уравнения. Нелинейным уравнением называется алгебраическое или трансцендентное (тригонометрическое, показательное, логарифмическое) уравнение вида:

$$\varphi(x) = 0,$$

где x – действительное число, а $\varphi(x)$ – нелинейная функция.

Точные методы позволяют записать корни в виде некоторого конечного соотношения (формулы). Но, как известно, многие уравнения и системы уравнений не имеют аналитических решений. В первую очередь это относится к большинству трансцендентных уравнений. Доказано также, что нельзя построить формулу, по которой можно было бы решить произвольное алгебраическое уравнение выше четвертой степени. Кроме того, в некоторых случаях уравнение содержит коэффициенты, известные лишь приблизительно, и, следовательно, сама задача о точном определении корней уравнения теряет смысл. Для решения таких уравнений применяются численные методы с заданной степенью точности.

К числу методов, используемых для численного решения уравнений, относится метод простой итерации. Для выполнения метода простой итерации уравнение вида $\varphi(x) = 0$ нужно преобразовать к виду $x=F(x)$ (это преобразование можно выполнить для любого $\varphi(x)$). Алгоритм метода простой итерации заключается в следующем: выбирается некоторое начальное приближение X_0, затем вычисляется X_1, X_2 и так далее по формуле:

$$X_{i+1} = F(X_i); i = 0,1 \ldots$$

до тех пор, пока не будет выполнено условие $/ X_{i+1}\text{-}X_i / < \varepsilon$, где ε – некоторое заранее заданное маленькое число, которое называется точностью или погрешностью вычислений. Во многих случаях вычисление удается выполнить с абсолютной компьютерной точностью, то есть для некоторого i выполняется условие $X_{i+1}=X_i$, но такой точности удается добиться не всегда. Метод простой итерации сходится, то есть за конечное число итераций дает значение корня с заданной точностью, при удачном выборе вида функции $F(x)$ и начального приближения.

Рассмотрим использование метода простых итераций на примере уравнения $x^3 - 12x^2 - 5x + 15 = 0$. Для удобства дальнейших вычислений преобразуем его к следующему виду: $x = (12x^2 + 5x - 15)^{1/3}$. Именно в таком виде мы будем его использовать в приведенной ниже программе, которая реализует вышеописанный алгоритм.

```
import math
def fx(x):
    t=12*x*x+5*x-15
    s=math.pow(t,1/3)
    return s

def it(x0):
    x1=x0
    while True:
        x0=x1
        x1=fx(x0)
        if x1==x0:
            break
    return x1

# Начало основной части программы
while True:
  x0=eval(input('Введите начальное приближение'));
  rez=it(x0);
  print('Корень уравнения равен %.3f' %rez);
  print('Для продолжения вычислений введите 0');
  print('для завершения работы введите 1');
  n=int(input())
  if n==1:
      break
```

В этой программе для вычисления значения выражения $(12x^2 + 5x - 15)^{1/3}$ разработана функция *fx*. В функции *fx* для возведения основания $12x^2 + 5x - 15$ в дробную степень (1/3) вызывается функция *pow*, которая входит в состав модуля *math*. Поэтому в самом начале программы, еще до описания функций, необходимо импортировать этот модуль.

Для выполнения собственно итерации используется функция *it*, из которой производится обращение к функции *fx*. В основной части программы пользователь с клавиатуры вводит начальное приближение, которое используется в качестве аргумента функции *it*. Эта функция и вычисляет корень уравнения, который выводится на экран компьютера (корень выводится с точностью до третьего знака после точки).

Результаты работы данной программы (при начальном приближении равном 10) приведены на рисунке 2.1.

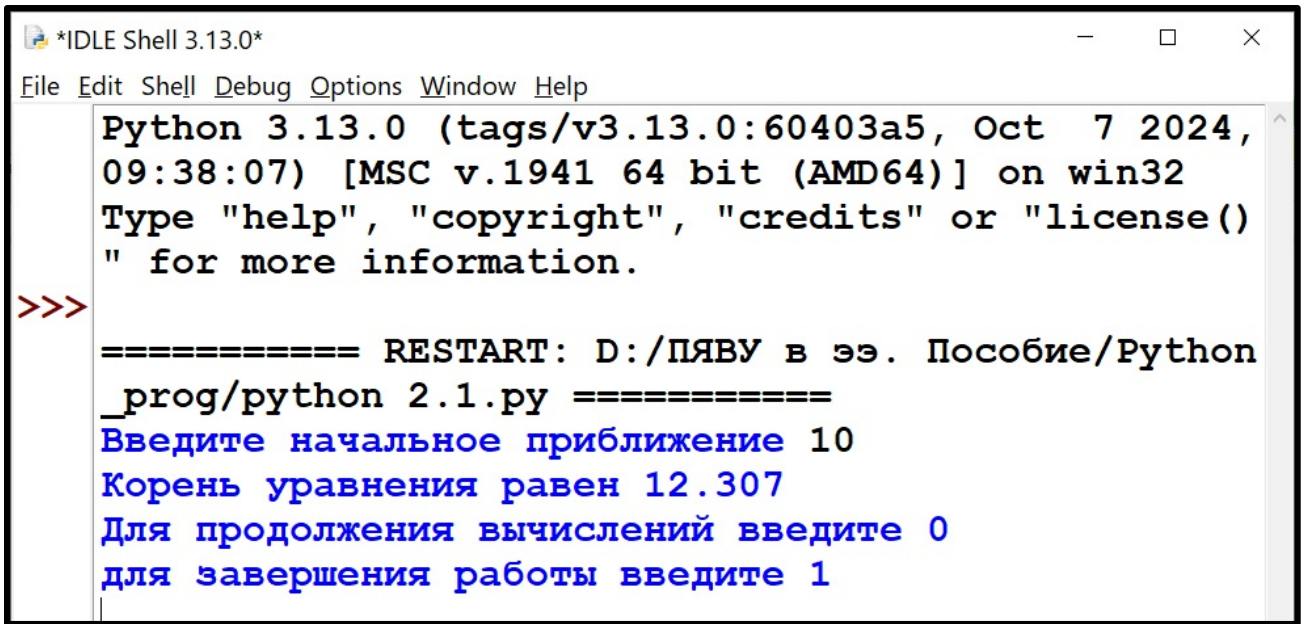

Рисунок 2.1 – Решение нелинейного уравнения методом простой итерации

С помощью данной программы можно найти один из корней данного уравнения, но поиск всех корней затруднен тем обстоятельством, что в качестве начального приближения нельзя вводить значения меньше 1, так как в результате в программе возникает ситуация, когда начинается вычисление логарифма отрицательного числа, что приводит к сбою в программе. Поэтому, для того чтобы гарантированно найти все три корня уравнения, воспользуемся другим численным методом.

Этим методом, часто употребляемым при численном решении нелинейных уравнений, является метод Ньютона (метод касательных). Метод Ньютона употребляется для уравнения в исходном виде $F(x)=0$ и записывается в следующем виде:

$$X_{i+1} = X_i - \frac{F(X_i)}{F'(X_i)};\ i = 1,2,\ldots,$$

где $F'(X_i)$ – производная функции.

При расчетах по методу Ньютона, так же как по методу простой итерации, задается начальное приближение X_0 и производятся итерации. Итерации выполняются либо до достижения абсолютной точности, либо до достижения заранее заданной погрешности ε. Метод Ньютона более трудоемок, так как он требует вычисления производной функции, но зато данный метод гарантирует сходимость к одному из корней при любом начальном приближении.

Данный метод реализован в приводимой ниже программе, которая решает то же нелинейное уравнение, что и рассмотренная ранее программа простой итерации.

```
import math

def fx(x):
  s=pow(x,3)
  kub=s
```

```
  f=kub-12*x*x-5*x+15
  return f

def dx(x):
  d=3*x*x-24*x-5
  return d

def new(x0):
  x1=x0;
  while True:
      x0=x1;
      x1=x0-fx(x0)/dx(x0)
      if x1==x0:
          break
  return(x1)

# Начало основной программы
while True:
    x0=eval(input('Введите начальное приближение'))
    rez=new(x0)
    print('Корень уравнения равен %.3f'%rez)
    print('Для продолжения вычислений введите 0')
    print('для завершения введите 1')
    n=int(input())
    if n==1:
      break
```

Перед тем как приступить к составлению этой программы, необходимо аналитическим путем найти производную функции $f(x)=x^3 – 12x2 – 5x + 15$. Полученная при вычислениях производная будет равна $3x^2 – 12x – 5$. Вычисление значения самой функции в программе производится в функции *fx*, а вычисление значения производной – в функции *dx*. Сам метод Ньютона реализован в функции *new*, из которой по мере необходимости производится обращение к функциям *fx* и *dx*.

В основной части программы осуществляется только ввод начального приближения, обращение к функции *new* для осуществления вычислений и вывод полученного значения корня. Так как корней несколько, то предусматривается неоднократное выполнение программы с различными начальными приближениями, для чего в программе использован цикл while True. Операторы, составляющие основную часть программы, будут все время повторяться до тех пор, пока с клавиатуры не будет введено число 1.

На рисунке 2.2. показан результат выполнения программы, которая решает нелинейное уравнение методом Ньютона.

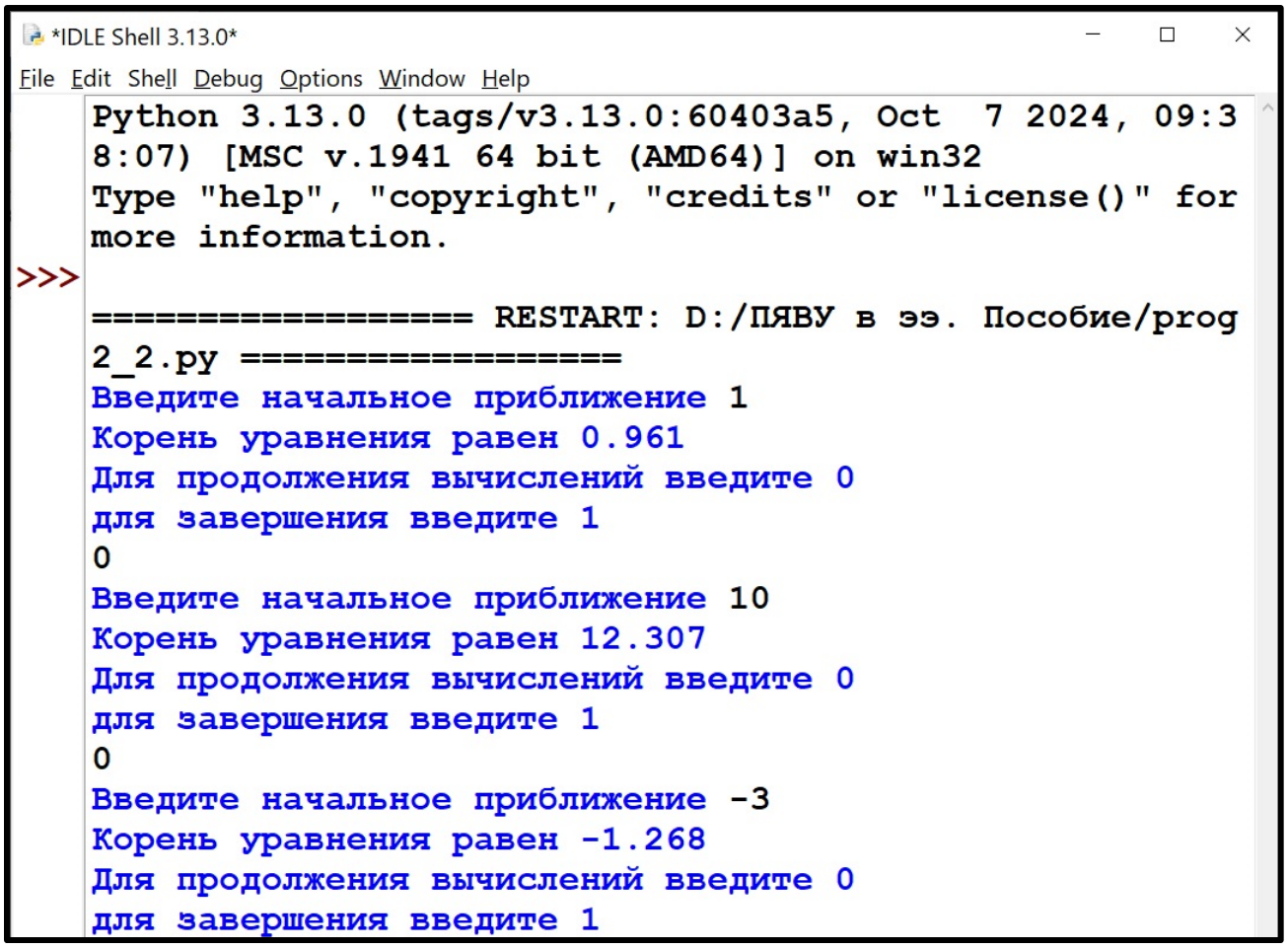

Рисунок 2.2 – Решение нелинейного уравнения методом Ньютона (касательных)

Последовательно вводя различные начальные приближения, можно найти все три корня данного нелинейного уравнения, что и продемонстрировано на рисунке 2.2.

Задание

Определить все вещественные корни функции *f(x)* методом касательных. Вид *f(x)* указан в таблице 3. Допустимая погрешность вычисления корней составляет ε=0,001. Начальные приближения принять самостоятельно.

Таблица 3 – Аналитический вид *f(x)*

Вариант	*f(x)*	Вариант	*f(x)*
1, 17	$x-\cos(x)$	9	x^3-3x^2+2x-1
2, 18	$x-tg(x)$	10	$x^3+3x^2+2x-8{,}448$
3, 19	$x+\sin(x)-1$	11	$x^3-3x^2+1{,}5x+0{,}649$
4, 20	x^3-e^x-1	12	x^4+2x^2-6x+2
5	x^3-5x-1	13	x^4-4x+1
6	$x^3+10x-10$	14	x^4-6x^2+4x+1
7	x^3-4x+2	15	$x^4-2x^3-x^2+x+0{,}51$
8	x^3+x^2+x-1	16	$x^5-x-0{,}2$

Практическое задание № 3. Численное интегрирование

При математическом моделировании ряда электрофизических процессов необходимо вычислять интегралы функций, аналитический вид которых сложен или неизвестен, а сами значения функций получаются расчетным путем или заданы таблично. Поэтому для успешного решения задачи из области моделирования обучающемуся необходимо овладеть методами численного интегрирования.

Приближенное вычисление значения определенного интеграла

$$\int_a^b f(x)dx$$

производится следующим образом: участок от a до b разбивается на n равных по величине отрезков. Длина каждого отрезка $h = (b - a)/n$. Значения аргумента в точках разбиения будут соответственно равны: $x_0 = a$, $x_1 = a + h$, …, $xi = x_0 + ih$, …, $x_n = b$. Значения функции $f(x)$ в точках x_i будут соответственно обозначаться y_i. Тогда согласно, например, формуле трапеций определенный интеграл можно будет вычислить следующим образом:

$$\int_a^b f(x)dx \approx h(\frac{y_0 + y_n}{2} + y1 + ...y_i + ... + y_n)$$

Программа на языке Python, реализующая вычисление определенного интеграла функции $y = \frac{1}{1+x}$ по методу трапеций, приведена ниже.

```
    print('Программа вычисления определенного интеграла для функции 1/(1+x)')
    a=eval(input('Введите начальный предел интегрирования'))
    b=eval(input('Введите конечный предел интегрирования'))
    n=int(input('Введите число отрезков, на которые будет разбит участок интегрирования'))
    h=(b-a)/n
    yp=0
    x=a
    for i in range(1,n):
        x=x+h
        yp=yp+1/(1+x)
    yn=1/(1+a)
    yk=1/(1+b)
    s=((yk+yn)/2+yp)*h
    print('Приближенное значение интеграла равно %.3f'%s)
```

В начале данной программы пользователь должен ввести с клавиатуры значения пределов интегрирования a и b и величину n – количество участков, на которые будет разбит промежуток интегрирования. Последнюю величину

рекомендуется брать достаточно большой, так как чем больше число участков, тем точнее будет полученный результат. Затем вычисляется величина *h* – длина участка, на которые разбивается отрезок.

Далее в программе определяется сумма промежуточных значений функции *y*, то есть сумма всех значений функции в точках x_i, кроме первого и последнего. Эта величина обозначается в программе *yp*. Начальное ее значение берется равным нулю, а итоговое значение этой величины вычисляется в цикле с заданным числом повторений. В цикле сперва определяется значение аргумента в каждой из точек x_i. Это значение вычисляется путем прибавления к предыдущему значению аргумента величины *h* (начальное значение аргумента берется равным a). Затем вычисляется соответствующее аргументу значение функции, и это значение прибавляется к величине *yp*.

По завершении работы цикла вычисляются величины *yn* и *yk* – значения функции в начальной и конечной точках отрезка интегрирования. Полусумма этих величин прибавляется к *yp*, и полученная сумма умножается на *h*. Таким образом, мы получаем по формуле трапеций искомую величину *s* – значение определенного интеграла для функции *1/(x+1)* на заданном пользователем отрезке. Теперь остается только вывести полученное значение на экран компьютера оператором вывода.

Результаты работы программы для вычисления определенного интеграла на отрезке от 0 до 10 приведены на рисунке 3.1.

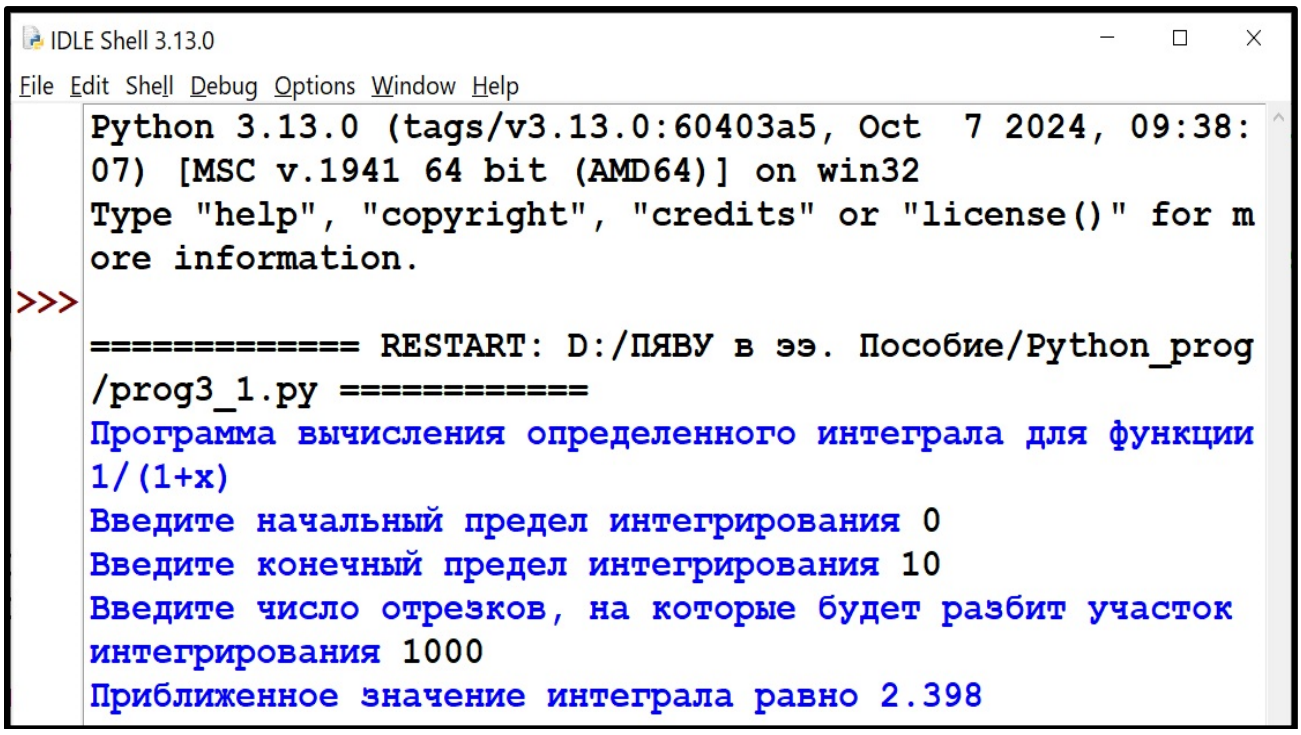

Рисунок 3.1 – Результаты вычисления определенного интеграла методом трапеций

Наряду с методом трапеций для приближенного вычисления определенного интеграла применяется также метод Симпсона. Суть этого метода заключается в том, что отрезок интегрирования разбивается на *2n* участков

длиной h. Затем на каждом участке, длина которого равна $2h$, подынтегральная функция заменяется параболой, которая проходит через начальную точку данного участка, его середину и конечную точку участка. То есть, если подынтегральная функция в точках с горизонтальными координатами x_i, x_{i+1} и x_{i+2} имеет соответственно значения y_i, y_{i+1} и y_{i+2}, то и аппроксимирующая ее парабола должна проходить через точки с координатами (x_i, y_i), (x_{i+1}, y_{i+1}) и (x_{i+2}, y_{i+2}).

Определение значения интеграла функции $f(x)$ по формуле Симпсона, называемой также формулой парабол, выглядит следующим образом:

$$\int_a^b f(x)dx \approx \frac{h}{6}((y_0 + y_{2n}) + 2(y_2 + y_4 + \cdots + y_{2n-2}) + 4(y_1 + y_3 + \cdots + y_{2n-1}))$$

На основании данной формулы можно составить программу, которая бы вычисляла значение определенного интеграла функции на заданном участке. Для упрощения процесса составления программы формулу Симпсона преобразуем к следующему виду:

$$\int_a^b f(x)dx \approx \frac{h}{3}(y_0 + y_{2n} + \sum_{i=1}^{2n-1}(3 + c_i)y_i),$$

где c – это переменная, которая принимает значение, равное 1, при нечетном значении i, и принимает значение, равное –1, при четном i.

Реализацию метода Симпсона на практике рассмотрим на примере составления программы, которая вычисляет значение определенного интеграла для подынтегральной функции $y = e^{-x^2}$.

Начальное и конечное значения отрезка интегрирования, а также количество участков, на которые разбивается отрезок интегрирования, вводятся пользователем с клавиатуры. Для вычисления значения заданной функции используется стандартная функция языка Python exp(). Для использования данной функции к программе необходимо подключить математический модуль math. Поскольку данная функция не является частью основного ядра Python, то каждый раз при ее использовании необходимо указывать имя модуля, в который она входит, а уже затем после точки имя самой этой функции.

Ниже приводится текст программы, которая выполняет поставленную задачу.

```
import math
print('Программа вычисления определенного интеграла ')
print('для функции y=exp(-X*X)')
a=eval(input('Введите начальное значение отрезка интегрирования'))
b=eval(input('Введите конечное значение отрезка интегрирования'))
```

```
    k=int(input('Введите количество участков, на которые будет разбит отрезок интегрирования'))
    x=a
    c=1
    s=0
    h=(b-a)/k
    m=k-1
    for i in range(1,m+1):
        x=x+h
        s=s+(3+c)*math.exp(-x*x)
        c=-c
    s1=math.exp(-a*a)+math.exp(-b*b)
    s=h*(s1+s)/3
    print('значение определенного интеграла на отрезке от %.2f до %.2f равно %.3f' %(a,b,s))
```

В данной программе после ввода исходных данных (значение количества участков разбиения отрезка присваивается переменной *k*, то есть $k = 2n$) производятся начальные присваивания. Аргументу *x* – значение начальной точки отрезка *a*. Переменной *s*, которая представляет собой вычисляемое в программе значение интеграла, будет присвоено начальное значение 0. Переменной *c* – начальное значение 1.

Затем в программе вычисляется значение суммы $\sum_{i=1}^{2n-1}(3 + c_i)\, y_i$. Для этого используется цикл с заранее заданным числом повторений. Начальное значение переменной цикла *i* равно единице, а конечное значение *m* равно $k - 1$, то есть $2n - 1$.

В цикле производятся следующие действия. Текущее значение аргумента увеличивается на *h*, к текущему значению переменной s прибавляется новая величина согласно приведенной формуле, текущее значение переменной с изменяется на противоположное. После завершения работы цикла находим величину *s1* – сумму начального и конечного значений подынтегральной функции.

Теперь подставим вычисленные величины в формулу и получим искомое значение *s*. Результаты работы программы при вычислении определенного интеграла экспоненциальной функции на отрезке от 0 до 1 методом Симпсона приведены на рисунке 3.2.

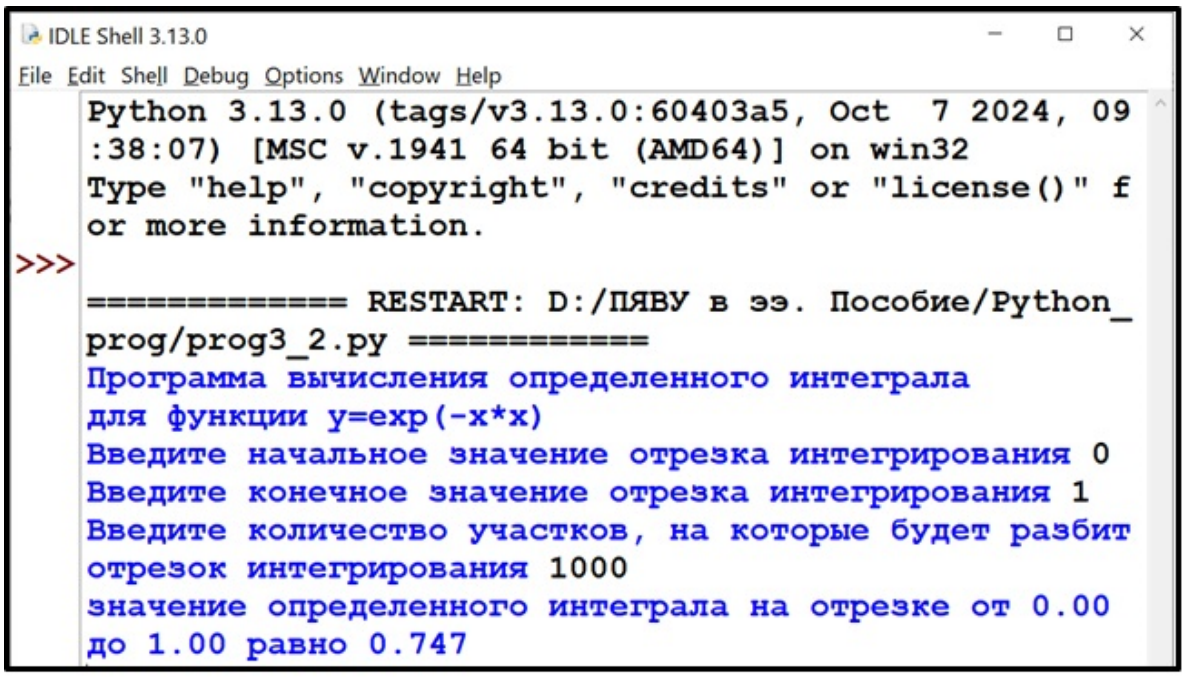

Рисунок 3.2 – Результаты вычисления определенного интеграла методом Симпсона

Задание

Значения телеизмерений (ТИ) нагрузки энергорайона p(t), выполненные в течении Δt = 60с с периодом обновления информации h = 10с, приведены в таблице 4. Составить программу для вычисления средней мощности $\bar{P}$(МВт) на интервале Δt. Средняя мощность определяется по формуле:

$$\bar{P} = \frac{1}{\Delta t}\int_0^{\Delta t} p(t)dt$$

Таблица 4 – Нагрузка энергорайона P(t), МВт

Номер ТИ	t, с	Варианты			
		1,5,9,13,17	2,6,10,14,18	3,7,11,15,19	4,8,12,16,20
		p(t)	p(t)	p(t)	p(t)
0	0	66	140	70	35
1	10	72	166	68	35
2	20	62	170	52	24
3	30	63	152	48	18
4	40	78	165	70	27
5	50	92	160	80	36
6	60	98	182	71	34

Практическое задание № 4. Интерполяция функции

Интерполяция в электроэнергетике используется для целей прогнозирования и в процедурах выбора оптимального шага при одномерном поиске минимума функций в расчетах режимов энергосистем.

В практической деятельности часто приходится работать с функциями, заданными таблично, когда для ряда значений аргумента $x_1,x_2,\ldots x_n$ известны значения функции $y_1=f(x_1)$, $y_2=f(x_2),\ldots,y_n=f(x_n)$. Подобные функции называются эмпирическими. Для того чтобы определить значение функции f в какой-либо точке x, отличной от заданных $x_1,x_2,\ldots x_n$, действуют следующим образом: строят функцию F, которая в заданных точках $x_1,x_2,\ldots x_n$ совпадает с заданными значениями $y_1,y_2,\ldots y_n$, то есть

$$F(x_i) = y_i, \text{ где } i = 1,2,\ldots n,$$

а при остальных x приближенно представляет эмпирическую функцию $f(x)$. Такая функция F называется интерполирующей, а точки $x_1,x_2,\ldots x_n$ называются узлами интерполяции.

Чаще всего функцию F задают в виде полинома, который называется интерполяционным полиномом Лагранжа и обозначается $L_n(x)$. Интерполяционный полином Лагранжа, построенный по таблице $(x_1,y_1),(x_2,y_2),\ldots(x_n,y_n)$, имеет следующий вид:

$$L_n(x) = \sum_{i=1}^{n} y_i \cdot \prod_{j=1}^{n} \frac{x - x_j}{x_i - x_j},$$

где $j \neq i$

В большинстве случаев интерполяционный полином строится не по всем заданным узлам $x_1,x_2,\ldots x_n$, а по некоторым выбранным узлам $x_1,x_2,\ldots x_m$. Здесь m – количество выбранных узлов, которое меньше, чем заданное n. При этом, как правило, в качестве узлов интерполяции выбираются ближайшие к x узлы, где x – это точка, в которой отыскивается приближенное значение функции $f(x)$.

На практике чаще всего применяются линейная интерполяция и квадратичная интерполяция. Для линейной интерполяции количество узлов берут равным 2, а для квадратичной – 3.

Вначале рассмотрим линейную интерполяцию для таблицы (x_1,y_1), $(x_2,y_2),\ldots,(x_n,y_n)$, в которой все x_i различны. В этом случае приближенное значение функции $f(x)$ в точке x вычисляется по формуле:

$$y = y_{i1} + (y_{i2} - y_{i1}) \cdot \frac{x - x_{i1}}{x_{i2} - x_{i1}},$$

где x_{i1} и x_{i2} – ближайшие к x узлы из набора $x_1,x_2,\ldots x_n$. Ниже приводится текст программы линейной интерполяции функции:

```
x=[0,1,2]
y=[0,1,2]

def lin(x0,x1,x2,y1,y2):
```

```
    y0=y1+(y2-y1)*(x0-x1)/(x2-x1)
    li=y0
    return (li)
for i in range (1,3):
  print('Введите значение аргумента в точке ',i)
  x[i]=eval(input())
  print('Введите значение функции в точке ',i)
  y[i]=eval(input())
print('Введите аргумент, для которого')
print('нужно вычислить значение функции')
x[0]=eval(input())
y[0]=lin(x[0],x[1],x[2],y[1],y[2])
print('Значение функции в точке ',x[0],' равно ',y[0])
```

В данной программе аргументы функции описываются в виде списка *x*, а значения функции – в виде списка *y*. В основной части программы оба списка заполняются данными с помощью цикла с заранее известным числом повторений.

Для списка, содержащего аргументы, известны все три значения. Элементы списка *x[1]* и *x[2]* содержат значения аргумента в узлах интерполяции, а значение *x[0]* – в той точке, для которой нужно определить значение функции.

Для списка, содержащего значения функции, известны только значения элементов *y[1]* и *y[2]*. Эти элементы содержат значения функции в узлах интерполяции. Неизвестной величиной является элемент списка *y[0]* – значение функции в заданной точке *x*. Для определения этой неизвестной величины в основной части программы производится обращение к функции *lin*, которая по указанной выше формуле и вычисляет искомую величину, которая затем выводится на экран компьютера.

На рисунке 4.1 показан результат работы программы линейной интерполяции.

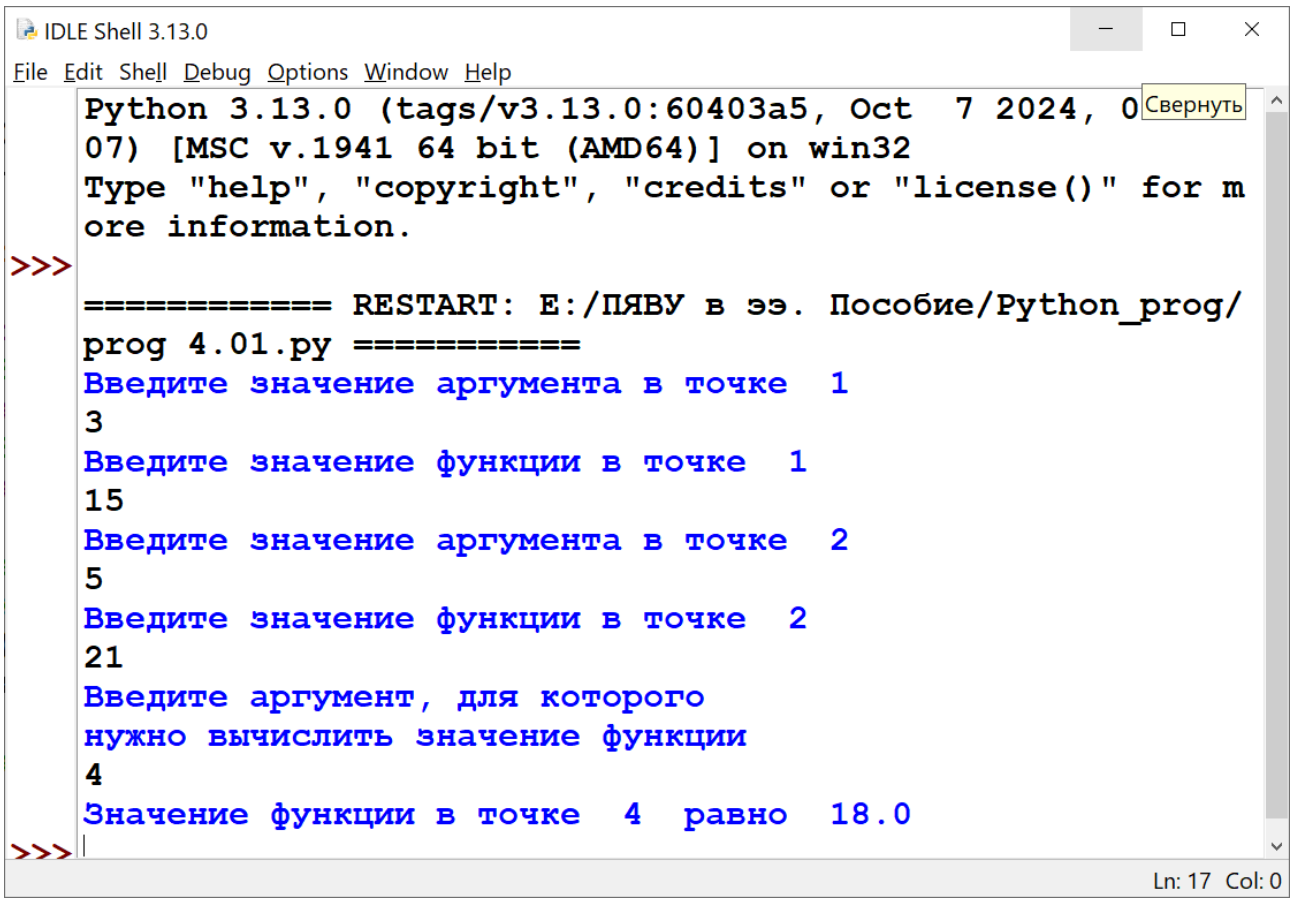

Рисунок 4.1 – Результаты работы программы линейной интерполяции функции

При квадратичной интерполяции функции для таблицы (x_1,y_1), $(x_2,y_2),\ldots,(x_n,y_n)$, в которой все x_i различны, приближенное значение функции $f(x)$ в точке x вычисляется по следующей формуле:

$$y = y_{i1} + (y_{i1} - y_{i2}) \cdot \frac{x - x_{i1}}{x_{i1} - x_{i2}} + (y_{i1} - y_{i2}) \cdot \frac{(x - x_{i2})(x - x_{i1})}{(x_{i1} - x_{i2})(x_{i1} - x_{i3})} - (y_{i2} - y_{i3}) \cdot \frac{(x - x_{i2})(x - x_{i1})}{(x_{i2} - x_{i3})(x_{i1} - x_{i3})},$$

где x_{i1},x_{i2} и x_{i3} – ближайшие к x узлы из набора $x_1,x_2,\ldots x_n$.

Ниже приводится текст программы квадратичной интерполяции функции.

```
x=[0,1,2,3]
y=[0,1,2,3]

def kvadr(x0,x1,x2,x3,y1,y2,y3):
    y0=-(y2-y3)*(x0-x2)*(x0-x1)/((x2-x3)*(x1-x3))
    y0=y0+(y1-y2)*(x0-x2)*(x0-x1)/((x1-x2)*(x1-x3))
    y0=y0+y1+(y1-y2)*(x0-x1)/(x1-x2)
    kvad=y0
    return(kvad)

for i in range (1,4):
  print('Введите значение аргумента в точке ',i)
```

```
    x[i]=eval(input())
    print('Введите значение функции в точке ',i)
    y[i]=eval(input())
print('Введите аргумент, для которого')
print('нужно вычислить значение функции')
x[0]=eval(input())
y[0]=kvadr(x[0],x[1],x[2],x[3],y[1],y[2],y[3])
print('Значение функции в точке ',x[0],' равно ',y[0])
```

На рисунке 4.2 представлен результат работы программы квадратичной интерполяции

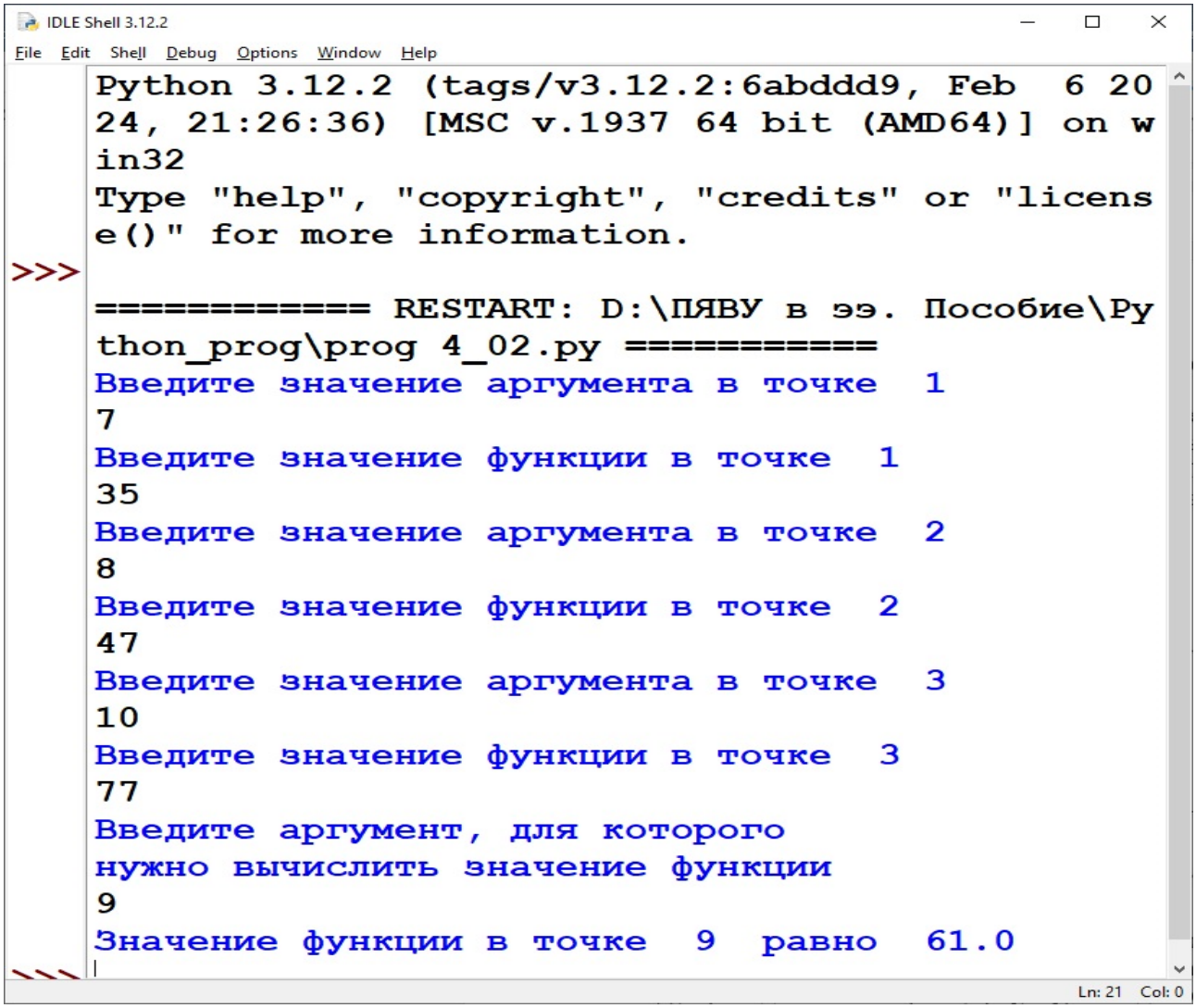

Рисунок 4.2 – Результаты работы программы квадратичной интерполяции функции

Задание

Эмпирическая функция $y(x)$ задана таблицей 5. Построить интерполяционный полином и найти значение $y(x)$ в точке x_k.

Таблица 5 – Исходные данные для задания

Вариант	Узлы интерполяции			x_k
	x_1	x_2	x_3	
	-2	-1	1	
	Значения $y(x_i)$			
	y_1	y_2	y_3	
1, 7, 13	16	5	1	1,5
2, 8, 14	-20	1	7	3
3, 9, 15	8	-2	2	2,5
4, 10, 16	9	2	6	1,5
5, 11, 17	-12	-1	3	1,5
6, 12, 18	13	3	1	0
7, 13, 20	-9	-1	3	3

Практическое задание № 5. Дифференциальные уравнения

Исследование переходных процессов в электрических системах требует умения решать дифференциальные уравнения.

Дифференциальным уравнением первого порядка называется уравнение следующего вида: $y'=f(x,y)$. Решением такого дифференциального уравнения называется функция $\varphi(x)$, при подстановке которой в уравнение данное уравнение обращается в тождество $\varphi'(x)=f(x,\varphi(x))$. В ряде случаев решением дифференциального уравнения является не одна функция, а множество функций. Нахождение функции $\varphi(x)$ называется аналитическим решением дифференциального уравнения.

Одним из частных (но достаточно распространенных при решении практических задач) случаев является ситуация, когда сама искомая функция неизвестна, но известно ее значение y_0 при некотором значении аргумента функции x_0. Такой частный случай решения дифференциального уравнения называется задачей Коши. Числа x_0 и y_0 называются начальными условиями. Полученное решение называется частным решением дифференциального уравнения при начальных условиях x_0 и y_0.

На практике часто используют не аналитическое решение задачи Коши, а численное, которое состоит в том, чтобы получить искомое решение в виде таблицы значений функции для ряда значений аргумента x на отрезке от a до b. Указанный отрезок разбивается на n участков, длина каждого из которых равна h. Точки $a, x_1, x_2, \ldots, x_i, \ldots b$, где $x_1 = a + h$, $x_2 = a + 2h$, $x_i = a + ih$ называются узловыми точками. В таком случае задача Коши сводится к нахождению множества значений функции y_i в узловых точках.

Существует ряд методов численного решения дифференциальных уравнений. Наиболее простым, но обладающим невысокой точностью, является метод Эйлера. Более точным, но и более трудоемким, является метод Рунге-Кутта.

При вычислении значений функции по методу Эйлера для нахождения значения аргумента и функции в i-й узловой точке используются следующие формулы:

$$x_i = x_{i-1} + h,\ y_i = y_{i-1} + hf(x_{i-1}, y_{i-1}),$$

где $i = 1, 2, \ldots n$

При вычислении значений функции с использованием классического метода Рунге-Кутта используется такая последовательность формул:

$$x_i = x_{i-1} + h,\ y_i = y_{i-1} + \Delta y_{i-1},$$

где $i = 1, 2, \ldots n$

Разница между текущим и предыдущим значением функции вычисляется по следующим формулам:

$$\Delta y_{i-1} = \frac{1}{6}(k_1^{i-1} + 2k_2^{i-1} + 2k_3^{i-1} + k_4^{i-1});$$

$$k_1^{i-1} = hf(x_{i-1}, y_{i-1});$$

$$k_2^{i-1} = hf(x_{i-1} + \frac{1}{2}h, y_{i-1} + \frac{1}{2}k_1^{i-1});$$

$$k_3^{i-1} = hf(x_{i-1} + \frac{1}{2}h, y_{i-1} + \frac{1}{2}k_2^{i-1});$$

$$k_4^{i-1} = hf(x_{i-1} + h, y_{i-1} + k_3^{i-1}).$$

Из приведенных формул видно, что каждая последующая величина k вычисляется, исходя из предыдущей. Ниже приводится текст программы, которая строит таблицу, содержащую ряд значений. Эти значения представляют собой решение задачи Коши для функции $y' = 2x + 3y$, полученное с помощью классического метода Рунге-Кутта. Начальные условия и число узловых точек задает пользователь.

```
# Создание массива (списка)
k=[0]*4

# Описание функции f
def f(x,y):
  fk=2*x+3*y
  return fk

# Начало основной части программы
# Ввод исходных данных
print('Программа решения дифференциального уравнения dy/dx=2x+3y')
print('с помощью классического метода Рунге-Кутта')
y=eval(input('Введите начальное значение функции y0'))
a=eval(input('Введите численное значение начала отрезка'))
b=eval(input('Введите численное значение конца отрезка'))
n=int(input('Введите количество узловых точек на отрезке'))

#Вычисления и вывод результатов
x=a
h=(b-a)/(n*1000)
print("Решение уравнения")
print('  x       y')
print("%3d %7d" %(x,y))
for i in range (0,n*1000+2):
  k[0]=(h*f(x,y))
  k[1]=(h*f(x+0.5*h,y+0.5*k[0]))
  k[2]=(h*f(x+0.5*h,y+0.5*k[1]))
  k[3]=(h*f(x+h,y+k[2]))
  y=y+(k[0]+2*k[1]+2*k[2]+k[3])/6
  x=x+h
  if (i%1000==1) and (i>1):
    print("%7.3f %7.3f" %(x,y))
```

Программа начинается с описания функции *f* с аргументами *x* и *y*, которая будет использована в дальнейших вычислениях.

Затем вводится начальное значение функции y_0, которое присваивается переменной *y*. Далее вводятся начальное и конечное значения отрезка, содержащего значения аргумента. Следующим вводится число узловых точек, которое присваивается переменной *n*. После этого начальное значение отрезка присваивается переменной *x* и вычисляется *h* – длина каждого из участков, на которые разбивается данный отрезок.

На следующем этапе работы программы на экран выводятся начальные значения аргумента и функции. Подсчет всех следующих величин аргумента и значений функции производится в цикле с заранее заданным числом повторений. Число повторений цикла можно было бы сделать равным числу заданных узловых точек, но в таком случае при небольшом количестве таких точек погрешность вычислений будет слишком велика. Это обстоятельство объясняется тем, что погрешность при вычислениях по методу Рунге-Кутта прямо пропорциональна длине участков, на которые разбивается отрезок. Чем меньше длина отрезка и чем соответственно больше количество узловых точек, тем точнее будет полученный результат. Поэтому в программе целесообразно увеличить число узловых точек, введенных пользователем, например, в тысячу раз с соответствующим уменьшением длины участка (поэтому при нахождении этой длины в программе мы делим отрезок не на *n*, а на *n**1000). Число повторений цикла (при каждом выполнении тела цикла производится вычисление значения функции в одной из узловых точек) также увеличивается в 1000 раз.

Подсчет каждого значения функции производится следующим образом. Вначале вычисляется первый из коэффициентов *k*, для нахождения которого достаточно знать величину *h* и значение выражения, содержащегося в правой части уравнения в предыдущей узловой точке. Это значение вычисляется с помощью созданной в программе функции *f*. Затем, исходя из значения первого коэффициента, вычисляется второй и далее – третий и четвертый. После вычисления всех коэффициентов можно вычислить значение функции.

После определения значений функции и аргумента проверяется номер узловой точки, и если он окажется равным 1001, 2001 и так далее, то данные значения будут выведены на экран компьютера. Таким образом, общее количество выведенных на экран узловых точек будет в 1000 раз меньше числа повторений цикла и будет соответствовать указанному пользователем. Многократное же повторение процесса вычислений обеспечит удовлетворительную точность результата.

Экран программы с решением дифференциального уравнения приведен на рисунке 5.1.

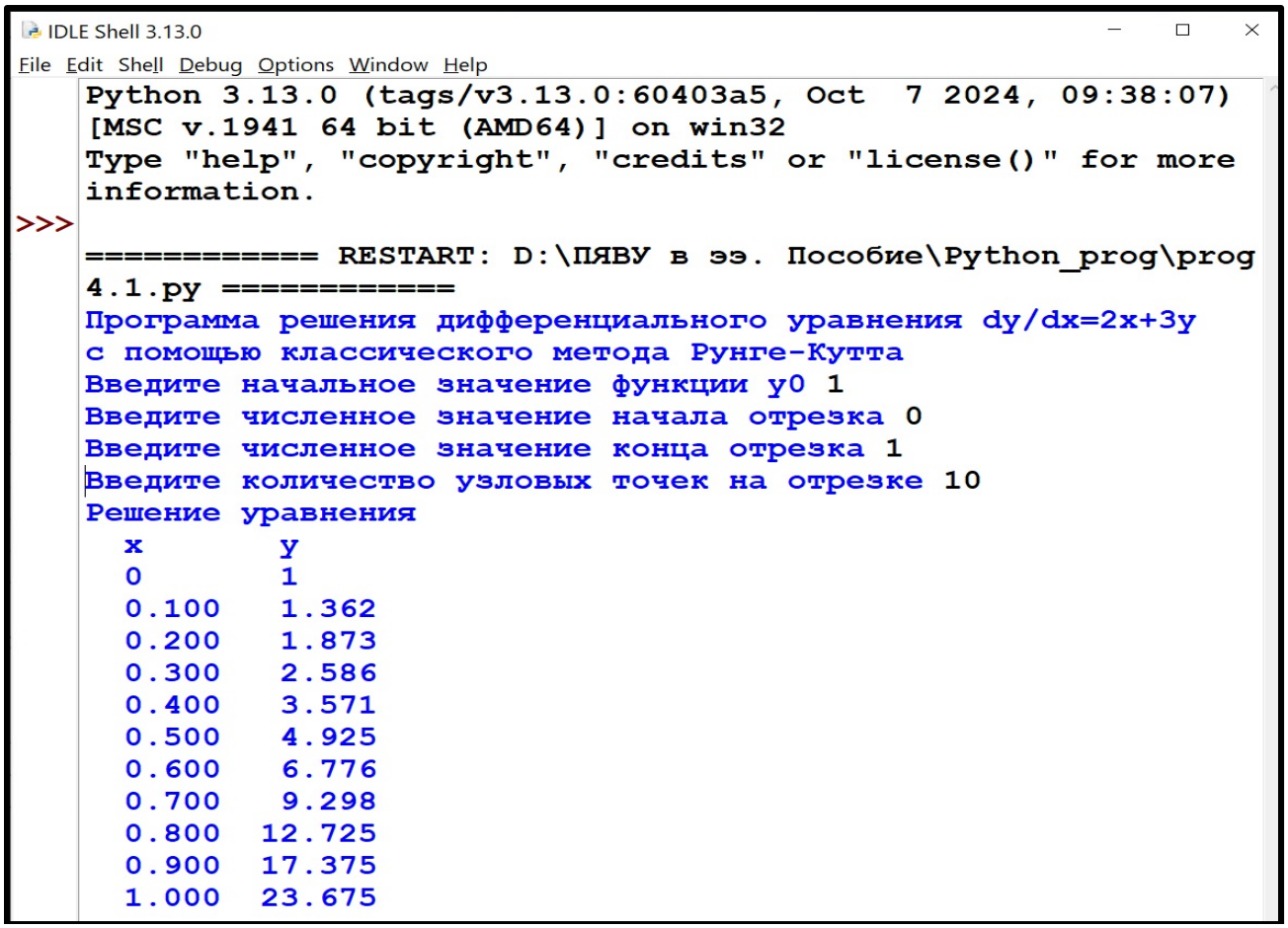

Рисунок 5.1 – Решение дифференциального уравнения методом Рунге-Кутта

Задание

Методом Рунге-Кутта найти приближенное решение задачи $dy/dx = f(x,y)$, удовлетворяющее начальному условию $y(x_0) = y_0$. Зависимость $y(x)$ получить с шагом $h = 0,1$ для интервала $(x0, b)$ изменения аргумента x. Исходные данные указаны в таблице 6.

Таблица 6 – Исходные данные для решения дифференциального уравнения

Вариант	$f(x,y)$	x_0	y_0	b
1,7,13,19	x^2+y^2+1	0	0	0,6
2,8,14,20	x^2-2y^2	-1	1	-0,6
3,9,15	x^2-2y	-1	1	-0,6
4, 10,16	x^2+y^2	0	0	0,4
5, 11,17	$x+y^2$	0	1	0,5
6,12,18	e^x-y/x	1	1	1,5

ЗАКЛЮЧЕНИЕ

В данных методических указаниях был сформулирован порядок выполнения контрольной работы по дисциплине «Программирование на языках высокого уровня в электроэнергетике».

В методических указаниях была определена цель выполнения контрольной работы, рассмотрен порядок ее выполнения, указаны основные требования к ее содержанию и оформлению. Приведены варианты заданий для всех практических работ, входящих в контрольную работу, и представлены необходимые пояснения о том, каким образом нужно решать поставленные перед обучающимися задачи.

Теоретические знания и практические навыки, приобретенные при выполнении данной контрольной работы, должны стать необходимой частью подготовки будущих специалистов в области электроэнергетики и электротехники.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Грунин, О. М. Математические задачи энергетики : учебное пособие /О. М. Грунин, Л. В. Савицкий. – Чита : ЗабГУ, 2014. – 260 с. – Текст : непосредственный.
2. Дроботун, Н. В. Алгоритмизация и программирование. Язык Python: учебное пособие / Н. В. Дроботун, Е. О. Рудков, Н. А. Баев. – СПб. : СПбГУПТД, 2020. – 119 с. – URL: http://publish.sutd.ru/tp_ext_inf_publish.php?id=202064 (дата обращения: 12.08.2025). – Текст : электронный.
3. Костин, В. Н. Оптимизационные задачи электроэнергетики : учебное пособие / В. Н. Костин. – СПб. : СЗТУ, 2003. – 120с. – Текст : непосредственный.
4. Маслобоев, А. Н. Языки и методы программирования. Основы программирования на языке Python : учебное пособие / А. Н. Маслобоев. – СПб. : ВШТЭ СПбГУПТД, 2024. – 62 с. – Текст : непосредственный.
5. Митрофанов, С. В. Применение библиотек языка Python для расчета режимов электроэнергетических систем: учебное пособие / С. В. Митрофанов, П. В. Матренин. – Новосибирск: Изд-во НГТУ, 2024. – 132 с. – Текст : непосредственный.
6. Пестриков, В. М. Решение математических задач в Turbo Pascal : учебное пособие / В. М. Пестриков, А. Н. Маслобоев. – СПб. : СПбГТУРП, 2009. – 110с. – Текст : непосредственный.
7. Сузи, Р. А. Язык программирования Python : учебное пособие Р. А. Сузи. – 4-е изд. – М. : ИНТУИТ ; Ай Пи Ар Медиа, 2024. – 350 с. – URL: https://www.iprbookshop.ru/142310.html (дата обращения: 12.08.2025). – Текст : электронный.

ПРИЛОЖЕНИЕ 1. ОБРАЗЕЦ ТИТУЛЬНОГО ЛИСТА КОНТРОЛЬНОЙ РАБОТЫ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ И ДИЗАЙНА»

ВЫСШАЯ ШКОЛА ТЕХНОЛОГИИ И ЭНЕРГЕТИКИ

Институт энергетики и автоматизации

Контрольная работа по дисциплине «Программирование на языках высокого уровня в электроэнергетике»

Вариант №

Выполнил

студент группы 7-038 _____П. С. Иванов

(подпись, дата)

Проверил

преподаватель кафедры ПМИ _____ А. И. Сидоров

(подпись, дата)

Санкт-Петербург

2025