МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Московский государственный технический университет**

**имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**Московский техникум космического приборостроения**

**СПЕЦИАЛЬНОСТЬ 09.02.03 Программирование в компьютерных системах**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту по теме:**

РАЗРАБОТКА ПРОГРАММЫ РЕШЕНИЯ

СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ

Руководитель разработки

от техникума Н. А. Сидорова

(подпись, дата) (ФИО)

Разработчик А.В Симаньков

(подпись, дата) (ФИО)

Москва 2019

СОДЕРЖАНИЕ

Введение 4

1. Постановка задачи 6
   1. Метод Гаусса - Жордана 6
   2. Метод Гаусса - Зейделя 7
2. Структура программы 10
3. Схемы алгоритма программы 12
   1. Схема алгоритма основной программы 12
   2. Схема алгоритма функции jordana\_gauss\_method 13
   3. Схема алгоритма функции gauss\_seidel\_method 14
4. Отладка программы 15
5. Оптимизация программы 17
6. Тестирование программы 18
   1. Тестирование в нормальных условиях 18
   2. Тестирование в экстремальных условиях 19
   3. Тестирование в исключительных ситуациях 21
7. Руководство пользователя 23

Заключение 25

Список использованных источников 26

Приложение А Листинг программы 27

Приложение Б Результаты выполнения программы 36

# ВВЕДЕНИЕ

Система линейных алгебраических уравнений (линейная система, также употребляются аббревиатуры СЛАУ) — система уравнений, каждое уравнение в которой является линейным — алгебраическим уравнением первой степени.

Систему линейных алгебраических уравнений представляют в виде:

,

Где – коэффициенты при переменных;

– неизвестные, которые необходимо определить;

n – количество переменных;

– свободные члены.

m – количество уравнений;

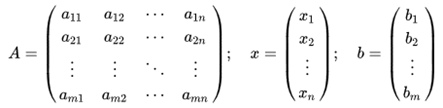
Индексы коэффициентов в СЛАУ формируются по следующему правилу:

* первый индекс (i) обозначает номер уравнения;
* второй (j) – номер переменной, при которой стоит этот коэффициент.

Матрица – математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых, действительных или комплексных чисел), которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Количество строк и столбцов задает размер матрицы. Хотя исторически рассматривались, например, треугольные матрицы, в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими.

Решение систем линейных алгебраических уравнений – одна из классических задач линейной алгебры, во многом определившая её объекты и методы. Кроме того, линейные алгебраические уравнения и методы их решения играют важную роль во многих прикладных направлениях, в том числе в линейном программировании, эконометрике.

Для решения СЛАУ исходную систему записывают в виде расширенной матрицы:



В результате решение систем линейных уравнений сводится к операциям над матрицами.

Методы решения СЛАУ:

* Метод Гаусса;
* Метод Жордана – Гаусса;
* Метод Зейделя;
* Метод итерации;
* Метод Крамера;
* Метод прогонки;
* Метод Якоби.

В данном курсовом проекте будут использованы методы решения систем линейных алгебраических уравнений Жордана – Гаусса, Зейделя.

# 1 Постановка задачи

Темой курсового проекта является «Разработка программы решения системы линейных уравнений».

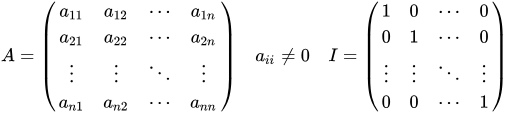
Разработка программы решения системы линейных уравнений:

* методом Жордана – Гаусса;
* методом Зейделя.

## 1.1 Метод Жордана - Гаусса

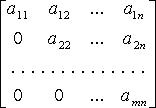
Метод Жордана - Гаусса – метод полного исключения неизвестных.

Пусть дано:



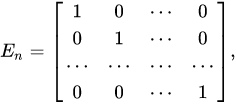
Алгоритм решения:

1. Выбирается первый слева столбец матрицы, в котором есть хоть одно отличное от нуля значение.
2. Если самое верхнее число в это столбце нуль, то меняется вся первая строка матрицы с другой строкой матрицы, где в этой колонке нет нуля.
3. Все элементы первой строки делят на верхний элемент выбранного столбца.
4. Из оставшихся строк вычитают первую строку, умноженную на первый элемент соответствующей строки, с целью получить первым элементом каждой строки (кроме первой) нуль.
5. Далее проводится аналогичная процедура с матрицей, получающейся из исходной матрицы после вычеркивания первой строки и первого столбца.
6. После повторения этой процедуры n – 1 раз получается верхняя треугольная матрица (где n – размерность матрицы).



1. Вычитается из последней строки последняя строка, умноженная на соответствующий коэффициент с тем, чтобы в предпоследней строке осталась только 1 на главной диагонали.
2. Предыдущий шаг повторяется для последующих строк.

В итоге получается единичная матрица:

.

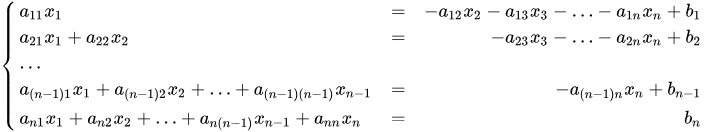
Решением является свободный вектор (весь алгоритм осуществляется с расширенной матрицей).

## 1.2 Метод Зейделя

Метод Зейделя – метод последовательных замещений. Он является модификацией метода Якоби.

Пусть дано:

Тогда система

,

образуется по алгоритму: в j-ом уравнении переносится в правую часть все члены, содержащие xi, для i > j. Эта система может быть представлена как:

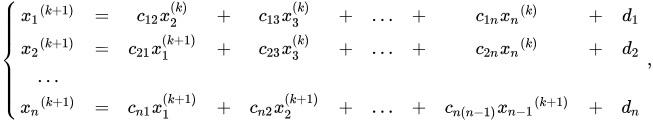
https://pp.userapi.com/c854528/v854528353/14b6f/Dd2Mixjos9M.jpg

где в принятых обозначениях D означает матрицу, у которой на главной диагонали стоят соответствующие элементы матрицы A, а все остальные нули; тогда как матрицы U и L содержат верхнюю и нижнюю треугольные матрицы A соответственно, на главной диагонали которых нули. Итерационный процесс в методе Гаусса – Зейделя строится по формуле:

https://pp.userapi.com/c854528/v854528353/14b7f/gWhVvRAD0Lk.jpg

После выбора соответствующего начального приближения https://pp.userapi.com/c854528/v854528353/14b86/UAzwde96m7g.jpg (чаще всего используют нули).

Модификация заключается в том, что новые значения https://pp.userapi.com/c854528/v854528353/14b8d/lvET8qxENUE.jpg используются сразу по мере получения, в то время как в методе Якоби они не используются до следующей итерации:



где

https://pp.userapi.com/c854528/v854528455/14bb1/gFFH8ZCRQAA.jpg

Таким образом, i-ая компонента (k + 1)-го приближения вычисляется по формуле:

https://pp.userapi.com/c854528/v854528455/14bb8/EIqFdFjibzg.jpg

Данная программа должна работать на ЭВМ типа PC и написана на языке Python, поскольку он имеет целый ряд преимуществ по сравнению с другими языками. К числу его особенностей можно отнести:

1) полностью автоматическое управление памятью. Данная функция позволяет программистам избежать волнений по поводу необходимости распределять или освобождать память;

2) выполнение операций осуществляется в более высоком уровне абстракций отчасти по причине архитектуры языка, отчасти благодаря расширенной библиотеке кодов, поставляемой вместе с Питон;

3) массив может включать объекты различных типов;

4) значение любого типа может быть назначено переменной;

5) язык легко объединяется с написанными на С и С++ модулями, что позволяет существенно увеличить скорость программ.

Благодаря этим особенностям развертка приложений может выполняться очень быстро.

# 2 Структура программы

Структура программы представлена на рисунке 2.1.



Рисунок 2.1 – Структура программы

Функции, используемые в программе, приведены в таблице 2.1.

Таблица 2.1 – Используемые функции

|  |  |
| --- | --- |
| Функция | Назначение |
| gauss\_seidel\_method | Функция решения СЛАУ методом Зейделя |
| Jordana\_gauss\_method | Функция решения СЛАУ методом Жордана - Гаусса |

Переменные, используемые в основной программе, представлены в таблице 2.2.

Таблица 2.2 – Переменные, используемые в основной программе

|  |  |
| --- | --- |
| Переменная | Назначение |
| matrix | Объект класса Matrix |
| done | Переменная, по которой идет цикл выбора метода |
| answer | Переменная для выбора метода решения СЛАУ |

# 3 Схемы алгоритма программы

## 3.1 Схема алгоритма основной программы



## 3.2 Схема алгоритма функции jordana\_gauss\_method



## 3.3 Схема алгоритма функции gauss\_seidel\_method



# 4 Отладка программы

Отладка представляет собой процесс поиска и устранения ошибок в программном проекте. Она занимает значительную часть рабочего времени программиста, нередко большую, чем составление программы. Практически любая программа перед началом отладки содержит хотя бы одну ошибку.

Виды ошибок программного обеспечения (ПО):

1. синтаксические ошибки (ошибки, обнаруживаемые компиляторы при выполнении синтаксического и частично семантического анализа);
2. ошибки компоновки (ошибки, обнаруживаемые компоновщиком при объединении модулей программы);
3. ошибки выполнения (ошибки, обнаруживаемые ОС, аппаратными средствами или пользователем при выполнении программы).

Во время отладки данной программы были обнаружены такие синтаксические ошибки:

1. неправильное имя оператора – имя оператора задано неверно и при использовании такого имени возникают ошибки. Например, в строке matrix.set\_values() есть ошибка, потому что метод set\_values() не существует; Правильная запись: set\_values\_user\_mode()
2. неверное объявление метода – методы, объявленные неверно, вызывают синтаксические ошибки, т.к. все методы имеют свои уникальные имена. Например: метод multiply\_line\_by\_number (self, line\_index, number).

Неверный пример использования функции: matrix.multiplication\_line\_by\_number(1,2).

Верный пример использования функции: double matrix.multiply\_line\_by\_number(1,2).

Так же была обнаружена ошибка выполнения – тип float не подлежит выполнению подскрипта. Это связано с тем, что при обновлении переменных списка, обращение к элементам происходило без индекса.

Код программы с исправленной ошибкой представлен ниже:

Current[line] = (self.answers[line] – sum1 – sum2) / self.coefficients[line][line]

Еще одна ошибка выполнения: converge = sqrt(sum((current[line] - previous[line]) \*\* 2 for line in range(n))) <= 1/(10 \*\* self.epsilon) OverflowError: (34, 'Result too large'), поэтому необходимо ввести ограничение на величину вводимых коэффициентов, либо видоизменить условие окончания.

Для исправления ошибки была ограничена вводимая точность до 3-ех знаков после запятой.

Код программы с исправленной ошибкой представлен ниже:

while True:

try:

self.epsilon = int(input('Set accuracy (number of digits after point. It should be >= 0: '))

if self.epsilon < 0 or self.epsilon >= 3:

continue

except ValueError as error:

print(error)

else:

break

Все ошибки были исправлены. Для проверки правильности работы программы, необходимо провести тестирование.

# 5 Оптимизация программы

Оптимизация представляет собой процесс модификации программы для улучшения ее эффективности. Бывает двух видов:

1. оптимизация по времени выполнения программы;
2. оптимизация по используемой программной памяти.

Оптимизация одного параметра чаще всего происходит за счет другого. Кроме того, зачастую оптимизация и вовсе неэффективна, так как на нее тратится слишком много ресурсов и времени.

Данная программа была частично оптимизирована по обоим пунктам.

Для оптимизации памяти, занимаемой программой, использовались локальные переменные и методы класса. Также при генерации матриц и массивов использовались генераторы списков, которые позволяют получать элементы списка по одному, что избавляет от надобности хранить временные переменные с матрицами.

Для оптимизации по времени было использовано явное преобразование типов, исключение инвариантных выражений, а также сжатие циклов.

# 6 Тестирование программы

Во время тестирования программное обеспечение было проверено в нормальных и экстремальных условиях, а также в исключительных ситуациях.

## 6.1 Тестирование в нормальных условиях

При вводе элементов матрицы, входящих в диапазон вещественного типа, то есть, в нормальных условиях, программа работает корректно и без ошибок.

На рисунке 6.1 показан ввод начальных значений.

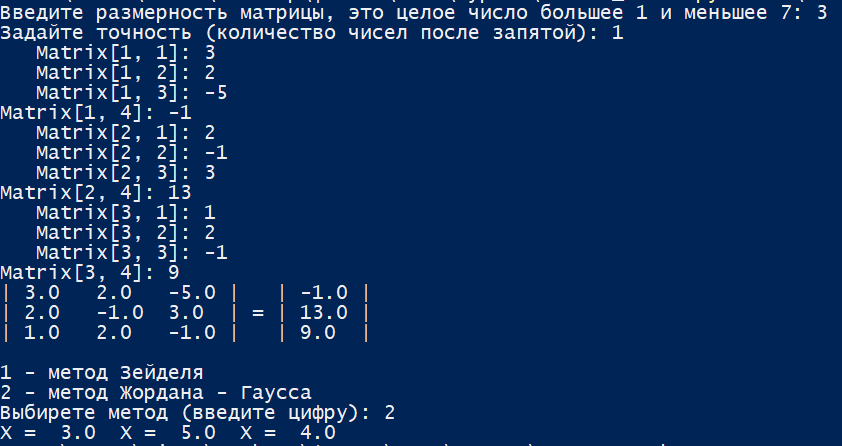


Рисунок 6.1 – Ввод начальных значений

На рисунках 6.2, 6.3 показаны выводы результатов работы программы.

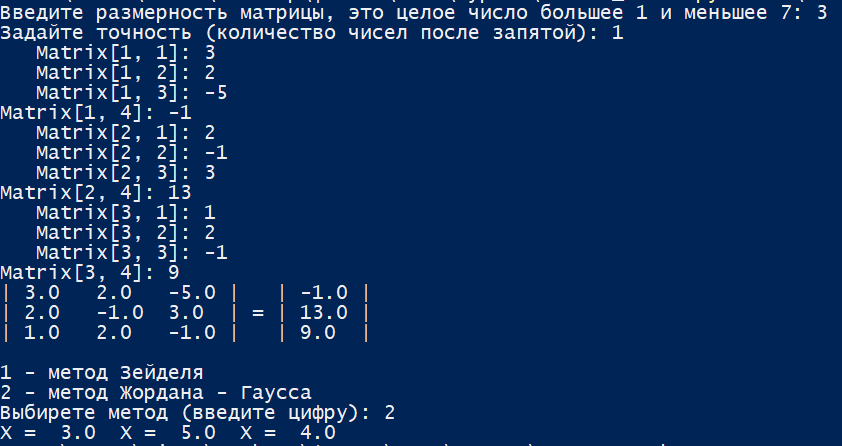


Рисунок 6.2 – Вывод результатов методом Зейделя

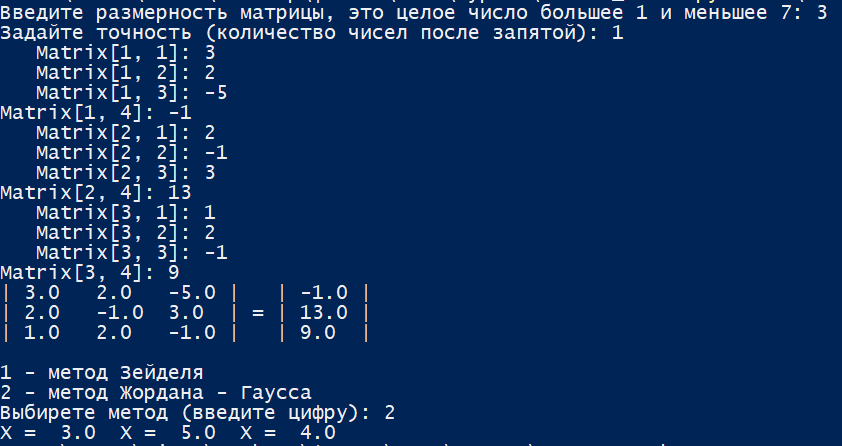
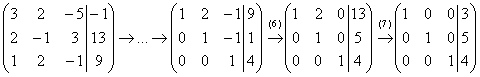


Рисунок 6.3 – Вывод результатов методом Жордана - Гаусса

Для проверки результатов работы программы был проведен ручной просчет для метода Жордана - Гаусса, который показан ниже.



Ответ: x1 = 3 x2 = 5 x3 = 4

**Вывод:** Тестирование в нормальных условиях прошло корректно. Так как результаты программы обоих методов совпали, а они совпали с результатами ручного просчета, то все методы вычисляют верные значения. Следовательно, программа работает правильно в нормальных условиях.

## 6.2 Тестирование в экстремальных условиях

Результаты программы при тестировании крайних значений ограничений ввода показано на рисунках 6.4. 6.5.

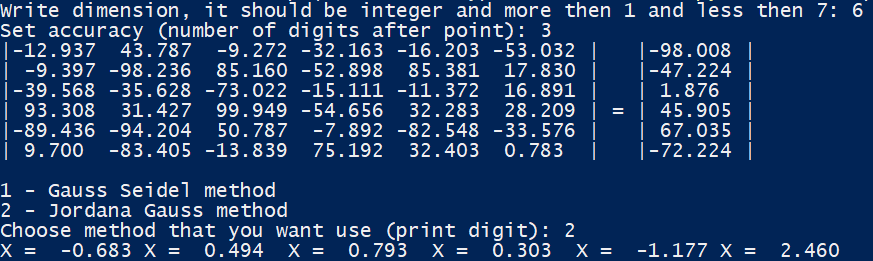


Рисунок 6.4 – Вывод результатов методом Жордана – Гаусса

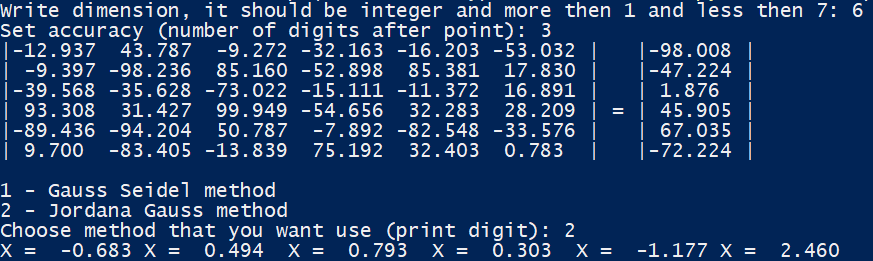


Рисунок 6.5 – Вывод результатов методом Зейделя

Для проверки результатов работы программы был проведен ручной просчет методом Жордана - Гаусса, который представлен ниже:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| -12.937 | 43.787 | -9.272 |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Первую строку делим на -12.937 и к 2 строке добавляем 1 строку, умноженную на 9.397; к 3 строке добавляем 1 строку, умноженную на 39.568; от 4 строки отнимаем 1 строку, умноженную на 93.308; к 5 строке добавляем 1 строку, умноженную на 89.436; от 6 строки отнимаем 1 строку, умноженную на 9.7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 43.787 | -9.272 |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Далее продолжаем приводить матрицу A(n)(n-1) к единичной и в результате получаем:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

x1 ≈ -0.683 x2 ≈ 0.494 x3 ≈ 0.793 x4 ≈ 0.0303 x5 ≈ -1.177 x6 ≈ 2.460

**Вывод:** Тестирование в экстремальных условиях прошло корректно. Так как результаты программы обоих методов совпали, а они совпали с результатами ручного просчета, то все методы вычисляют верные значения. Следовательно, программа работает правильно в экстремальных условиях.

6.3 Тестирование в исключительных ситуациях

На рисунке 6.6 изображена защита пользователя от ошибок.

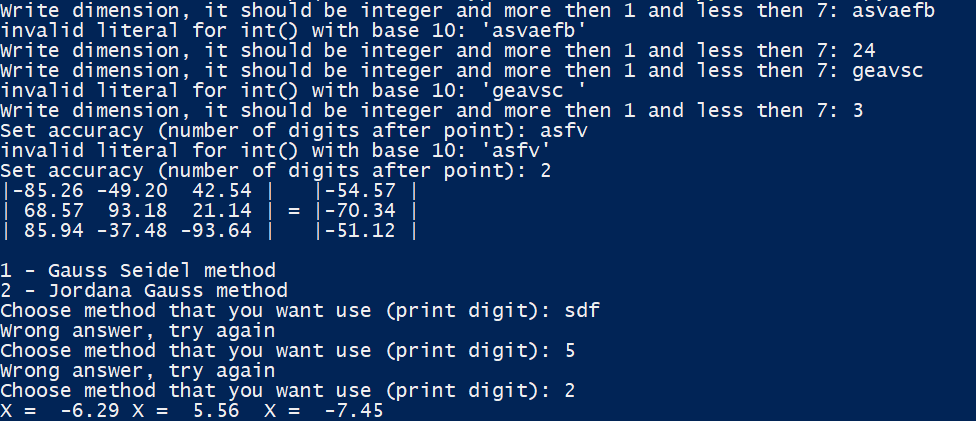


Рисунок 6.6 – Защита от ошибок

**Вывод:** При вводе некорректных значений программа сообщает об этом пользователю с просьбой повторить ввод данных. Следовательно, программа работает правильно в исключительных условиях.

**Вывод по тестированию:** Программа прошла тестирование в нормальных условиях, в экстремальных условиях и в исключительных ситуациях. Программа аварийно не завершается, следовательно, она работает корректно.

Листинг программы представлен в приложении А, результаты выполнения – в приложении Б.

# 7 Руководство пользователя

Минимальные системные требования:

– операционная система: Windows. Рекомендуемая ОС: Windows7 и выше;

– процессор: 1.8 ГГц;

– ОЗУ: 128 МБ;

– свободное место на жестком диске: 77КБ;

– наличие IBM – совместимых клавиатуры и мыши.

– интерпретатор Python версии 3.7

Программное обеспечение предназначено для вычисления корней системы линейных алгебраических уравнений. Для выполнения программы необходимо запустить файл main.py.

При запуске программы появляется окно для ввода начальных значений системы и выбора метода решения, которое показано на рисунке 7.1.

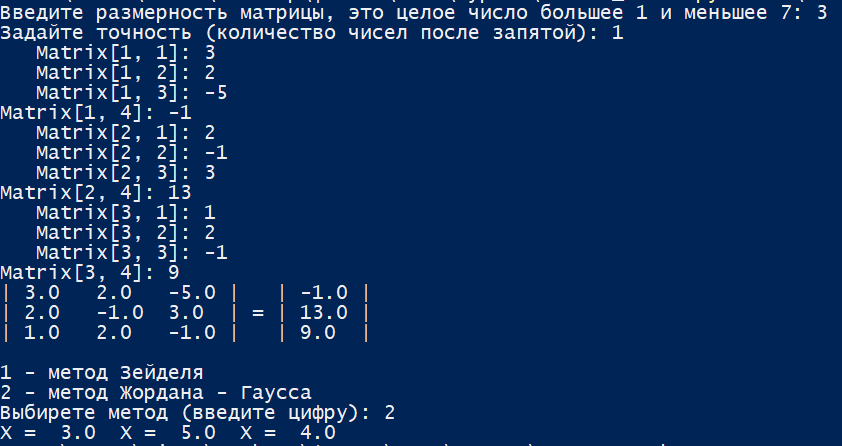


Рисунок 7.1 – Ввод начальных значений

На рисунках 7.2, 7.3 показаны результаты решения системы линейных уравнений.

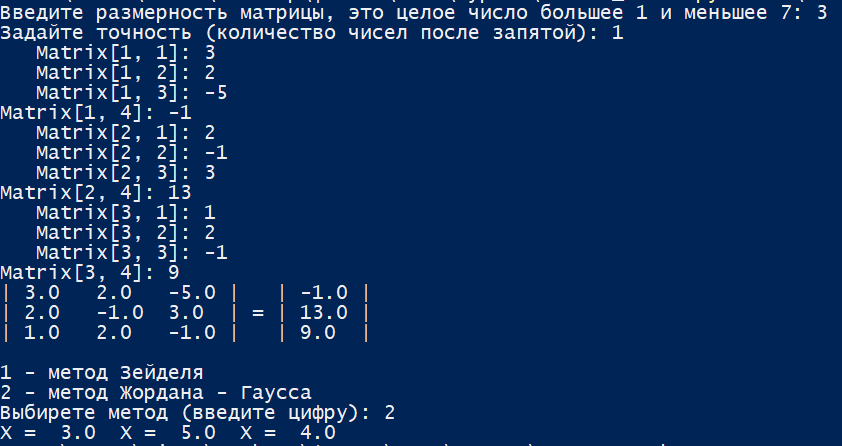


Рисунок 7.2 – Вывод результатов методом Жордана - Гаусса

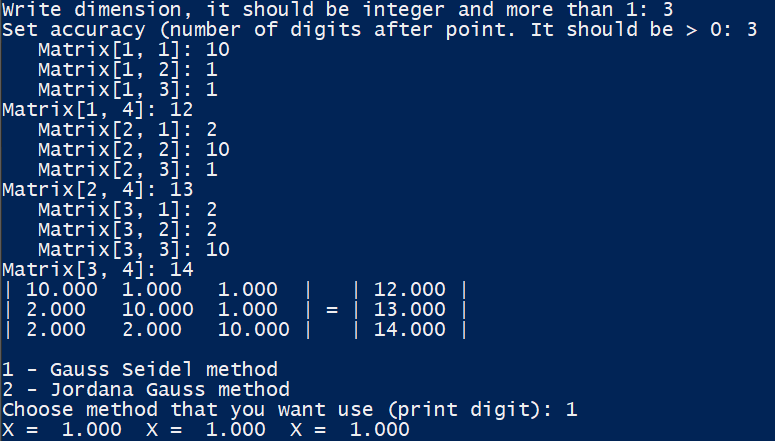


Рисунок 7.3 – Вывод результатов методом Зейделя

После этого можно закрыть окно программы нажатием на крестик окна в правом верхнем углу.

ЗАКЛЮЧЕНИЕ

Задача, поставленная в курсовом проекте, выполнена полностью. Созданная в курсовом проекте программа позволяет решить СЛАУ методом Жордана – Гаусса и методом Зейделя.

Программа понятна и проста в использовании, имеет удобный интерфейс, подсказки к действиям и выполняет все необходимые функции.

При разработке данной программы использовались все известные работы с консольными приложениями на языке Python. При использовании данного языка программирования намного проще работать с функциями, что является несомненным преимуществом языка Python перед аналогами.

Преимуществом данной программы является ее маленький размер и практичность. Использование консоли позволяет обеспечить наиболее быструю работу программы. Также вывод результатов нагляден и удобен для просмотра.

Листинг хорошо структурирован и имеет множество комментариев, что помогает легко понимать программу.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Березин И.С., Жидков Н.П. Методы вычислений, Т.2. М.: ГИФМЛ, 1959, 620 с.
2. Вабищевич П. Н. Численные методы. Вычислительный практикум. СПб.: Символ-плюс, 2010, 320 c.
3. Доусон М. Программируем на Python, СПб.: Питер, 2014, 416 с.
4. Лутц М. Изучаем Python, 4-е издание, СПб.: Символ-Плюс, 2011, 1280 с.
5. Лутц М. Программирование на Python, том I, 4-е издание, СПб.: Символ-Плюс, 2011, 992 с.
6. Лутц М. Программирование на Python, том II, 4-е издание, СПб.: Символ-Плюс, 2011, 992 с.
7. Пилгрим Марк. Погружение в Python 3, СПб.: Символ-плюс, 2012, 428 с.
8. Прохоренок Н.А. Python 3. Разработка приложений, СПб.: БХВ-Петербург, 2012, 704 с.
9. Прохоренок Н.А. Самое необходимое, СПб.: БХВ-Петербург, 2011, 416 с.
10. Саммерфилд, М. Программирование на Python 3, СПб.: Символ-плюс, 2015, 608 c.
11. Хахаев И. Практикум по алгоритмизации и программированию на Python, М.: Альт Линукс, 2010, 126 с.
12. Чаплыгин А.Н. Учимся программировать вместе с Python, СПб.: Символ-плюс, 2014, 450 с.
13. Шапошникова С. Основы программирования на Python. Вводный курс, СПб.: Символ-плюс, 2014, 300 c.
14. Downey Allen, ThinkPython and Kart, СПб.: Символ-плюс, 2011, 230 с.
15. Briggs J. R, Python for Kids, СПб.: Символ-плюс, 2012, 120 с.

# ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программы

"""

Курсовой проект по предмету МДК 03.01 Технология разработки программного обеспечения

по теме “Разработка программы решения системы линейный уравнений.”:

- методом Гаусса – Жордана;

- методом Гаусса – Зейделя.

Язык: Python 3.7

Среда: PyCharm

Название программы: main.py

Разработал: Симаньков А.В.

Дата: 31.03.2019

Версия: v 1.0

Задание:

Разработать программу, решающую систему линейных алгебраических уравнений

методами Жордана - Гаусса и Зейделя

Описание алгоритма программы:

а) Создание объекта класса Matrix

б) Запрос исходных данных от пользователя

1) Ввод пользователем размерности матрицы

2) Ввод пользователем точности вычислений

3) Ввод пользователем значений

в) Вывод построенной расширенной матрицы

г) Запрос выбора пользователем методом, которым будет решаться СЛАУ

д) Вывод результатов выполнения программы

Класс:

class\_matrix.py - класс матрица, хранящая в себе все необходимые данные и

весь необходимый функционал.

Использованные переменные:

matrix - объект класса Matrix;

answer - решение пользователя, какой метод использовать.

"""

from class\_matrix import Matrix

if \_\_name\_\_ == '\_\_main\_\_':

matrix = Matrix()

matrix.set\_size\_user\_mode()

matrix.set\_epsilon\_user\_mode()

matrix.set\_values\_user\_mode()

matrix.print\_extended\_matrix()

done = False

print("1 - Gauss Seidel method")

print("2 - Jordana Gauss method")

while not done:

answer = input("Choose method that you want use (print digit): ")

if answer == '1':

for elem in matrix.gauss\_seidel\_method():

print(f'X = {elem:^{matrix.epsilon + 5}.{matrix.epsilon}f}', end='')

done = True

elif answer == '2':

for elem in matrix.jordana\_gauss\_method():

print(f'X = {elem:^{matrix.epsilon + 5}.{matrix.epsilon}f}', end='')

done = True

else:

print('Wrong answer, try again')

"""

Course project

This program should find a solution of linear algebraic equations system

using methods of Jordana-Gauss and Gauss-Seidel

"""

from random import randrange, uniform

from math import sqrt

class Matrix(list):

#Этот класс используется для хранения данных и методов матрицы

def \_\_init\_\_(self, size=0):

#инициализация объекта

self.\_\_dimension = size

self.coefficients = [[0 for \_ in range(size)] for \_ in range(size)]

self.answers = [0 for \_ in range(size)]

self.epsilon = 1

def get\_coefficients(self) -> list:

# возвращает коэффициенты

return self.coefficients

def get\_answers(self) -> list:

#возвращает свободные члены

return self.answers

def get\_extended\_matrix(self) -> list:

#возвращает расширенную матрицу

out = self.coefficients

out.append(self.answers)

return out

def set\_size\_user\_mode(self):

#выводит на экран приглашение пользователю на ввод размерности матрицы

while True:

try:

self.\_\_dimension = int(input("Write dimension, it should be integer and more then 1 and less then 7: "))

if self.\_\_dimension <= 1 or self.\_\_dimension >= 7:

continue

except ValueError as error:

print(error)

continue

else:

self.coefficients = [[0 for \_ in range(self.\_\_dimension)] for \_ in range(self.\_\_dimension)]

self.answers = [0 for \_ in range(self.\_\_dimension)]

break

def set\_values\_user\_mode(self):

#выводит на экран приглашение пользователю на ввод значений

for line in range(self.\_\_dimension):

for column in range(self.\_\_dimension): #ввод данныхх

while True:

try:

self.coefficients[line][column] = float(input(f' Matrix[{line+1}, {column+1}]: '))

except ValueError as error:

print(error)

continue

break

while True:

try:

self.answers[line] = float(input(f'Matrix[{line+1}, {self.\_\_dimension+1}]: '))

except ValueError as error:

print(error)

continue

break

def set\_epsilon\_user\_mode(self):

#Выводит на экран приглашение пользователю на ввод точности вычислений (количество знаков после запятой)

while True:

try:

self.epsilon = int(input('Set accuracy (number of digits after point): '))

except ValueError as error:

print(error)

else:

break

def set\_coefficients(self, coefs: list):

#задать коэффициенты матрицы

self.coefficients = coefs

def set\_answers(self, ans: list):

#задать свободные члены матрицы

self.answers = ans

def generate\_rand\_integer\_matrix(self, low=0, high=10):

#генерация матрицы целочисленными значениями

for line in range(self.\_\_dimension):

for column in range(self.\_\_dimension):

self.coefficients[line][column] = randrange(low, high)

self.answers = [randrange(low, high) for \_ in range(self.\_\_dimension)]

def generate\_rand\_float\_matrix(self, low=0, high=10):

#генерация матрицы вещественными значениями

for line in range(self.\_\_dimension):

for column in range(self.\_\_dimension):

self.coefficients[line][column] = round(uniform(low, high), self.epsilon)

self.answers = [round(uniform(low, high), self.epsilon) for \_ in range(self.\_\_dimension)]

def print\_matrix(self):

#простой вывод матрицы

for line in self.coefficients: #вывод матрицы

for elem in line:

print(f'{elem:<{self.epsilon + 2}.{self.epsilon}f}', end='')

print()

print()

def print\_extended\_matrix(self):

#простой вывод расширенной матрицы

for line\_index in range(self.\_\_dimension):

print('|', end='')

for column\_index in range(self.\_\_dimension):

print(f'{self.coefficients[line\_index][column\_index]:^{self.epsilon + 5}.{self.epsilon}f}', end='')

if self.\_\_dimension // 2 is line\_index:

print(f'| = |{self.answers[line\_index]:^{self.epsilon + 5}.{self.epsilon}f}|')

else:

print(f'| |{self.answers[line\_index]:^{self.epsilon + 5}.{self.epsilon}f}|')

print()

def swap\_lines(self, index\_1=0, index\_2=0):

#меняет местами линии в матрице

temp = self.coefficients[index\_1]

self.coefficients[index\_1] = self.coefficients[index\_2]

self.coefficients[index\_2] = temp

temp = self.answers[index\_1]

self.answers[index\_1] = self.answers[index\_2]

self.answers[index\_2] = temp

def swap\_columns(self, index\_1=0, index\_2=0, ans\_swap=False):

#меняет местами колонки в матрице

if ans\_swap:

for i in range(self.\_\_dimension):

temp = self.coefficients[i][index\_1]

self.coefficients[i][index\_1] = self.answers[i]

self.answers[i] = temp

else:

for i in range(self.\_\_dimension):

temp = self.coefficients[i][index\_1]

self.coefficients[i][index\_1] = self.coefficients[i][index\_2]

self.coefficients[i][index\_2] = temp

def adding\_lines(self, addend\_1, addend\_2, number):

#складывает строку 1 со строкой 2, умноженной на число

for elem\_index in range(self.\_\_dimension):

self.coefficients[addend\_1][elem\_index] += self.coefficients[addend\_2][elem\_index] \* number

self.answers[addend\_1] += self.answers[addend\_2] \* number

def multiply\_line\_by\_number(self, line\_index, number):

#умножает строку на число

if number != 0:

for elem\_index in range(self.\_\_dimension):

self.coefficients[line\_index][elem\_index] \*= number

self.answers[line\_index] \*= number

else:

print("number should not be zero")

def set\_not\_zero\_diagonal\_element(self, index=0):

#переставляет линии в матрице так, чтобы на главной диагонали не было нулей

if self.coefficients[index][index]: # check is elem != 0

return

for line\_index in range(index + 1, len(self.coefficients)):

if self.coefficients[line\_index][index]:

self.swap\_lines(index, line\_index)

break

else:

continue

def jordana\_gauss\_method(self):

#Метод Гаусса - Жордана

for diagonal\_index in range(self.\_\_dimension):

self.set\_not\_zero\_diagonal\_element(diagonal\_index)

self.multiply\_line\_by\_number(diagonal\_index, 1/self.coefficients[diagonal\_index][diagonal\_index])

for under\_index in range(diagonal\_index + 1, self.\_\_dimension):

self.adding\_lines(under\_index, diagonal\_index, -self.coefficients[under\_index][diagonal\_index])

for upper\_index in range(diagonal\_index - 1, -1, -1):

self.adding\_lines(upper\_index, diagonal\_index, -self.coefficients[upper\_index][diagonal\_index])

return self.answers

def gauss\_seidel\_method(self):

#Метод Гаусса - Зейделя

n = self.\_\_dimension

previous = [.0 for \_ in range(n)]

converge = False

while not converge:

current = previous[:]

for line in range(n):

sum1 = sum(self.coefficients[line][column] \* current[column] for column in range(line))

sum2 = sum(self.coefficients[line][column] \* previous[column] for column in range(line+1, n))

current[line] = (self.answers[line] - sum1 - sum2) / self.coefficients[line][line]

converge = sqrt(sum((current[line] - previous[line]) \*\* 2 for line in range(n))) <= 1/(10 \*\* self.epsilon)

previous = current[:]

return previous

# ПРИЛОЖЕНИЕ Б

(обязательное)

Результаты выполнения программы

На рисунке Б.1 представлен ввод начальных значений программы и выбор метода.

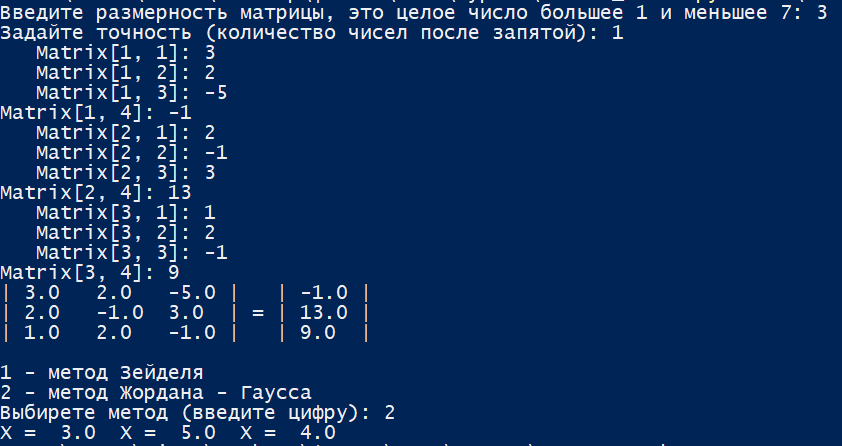


Рисунок Б.1 – Ввод начальных значений программы и выбор метода

На рисунках Б.2, Б.3 представлены результаты выполнения программы.

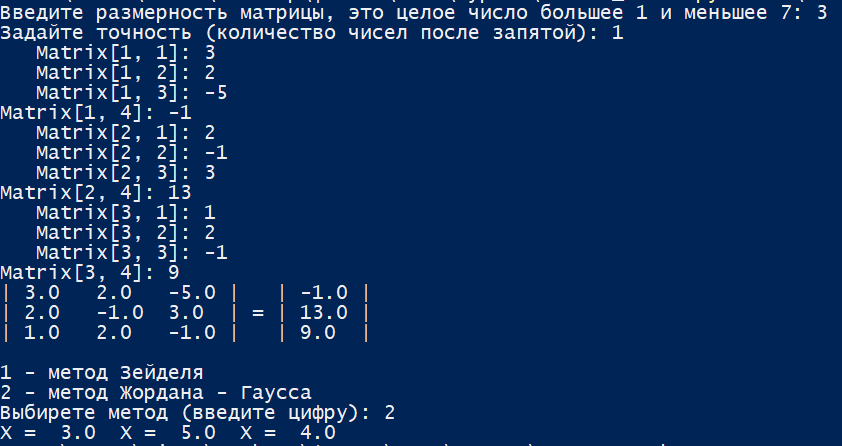


Рисунок Б.2 – Вывод результатов методом Жордана - Гаусса

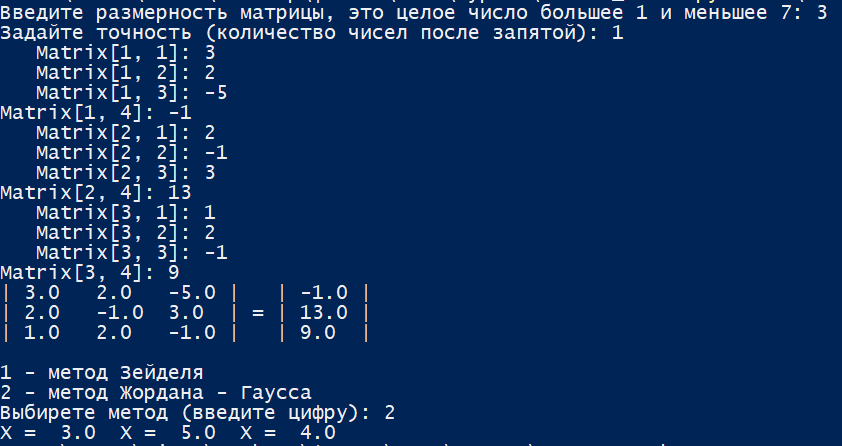


Рисунок Б.3 – Вывод результатов методом Зейделя