МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А.И. ГЕРЦЕНА»



Направление подготовки

09.03.01 – Информатика и вычислительная техника

Профиль «Технологии разработки программного обеспечения»

**Лабораторная работа №1**

**«Численные методы решения систем линейных алгебраических уравнений. Метод Гаусса»**

|  | Работу выполнил:  Балаев Жамал,  Васильева Марина,  Иванов Никита,  Шардт Максим  Рожков Максим  очная форма обучения  курс: 2; группа:ИВТ-1.1 |
| --- | --- |
|  | Научный руководитель:  Профессор Власова Елена Зотиковна |

Санкт-Петербург

2023

# Содержание

Отчет [Балаева Ж. Б. 3](#_30j0zll)

Отчет [Васильевой М. А. 10](#_wl4bpvrj617c)

Отчет [Иванова Н. Р. 17](#_vr2u23c4g3jp)

Отчет [Рожкова М. В. 24](#_8eb1w5y4s5hj)

Отчет [Шардта М. А. 31](#_onwp4dzdf3kr)

Приложение 38

### Выполнил Балаев Ж. Б. ИВТ 1.1

# Задача 1. Разработать программу по решению СЛУ методом (алгоритм исключения неизвестных по столбцам)

## Математическая модель

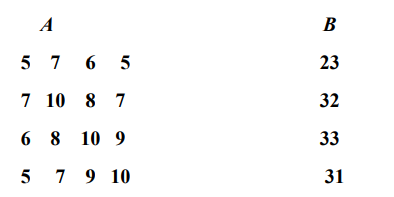
(1)

(2)

(3)

(4)

## Контрольный пример



## Моделирование

На языке Python был реализован этап прямого хода метода Гаусса.

В ходе алгоритма перебираются строки матрицы, и для каждой строки выбирается соответствующий диагональный элемент, а затем значения после делятся на этот элемент.

Во втором цикле из каждой строки вычитается кратное диагональному элементу количество первой строки, чтобы обнулить элементы под диагональю.

| def gauss\_direct(a):  n = len(a)  for i in range(n - 1):  p = a[i][i]  a[i][i] = 1  for j in range(i+1, n+1):  a[i][j] = round(a[i][j] / p, precision)  for k in range(i + 1, n):  for j in range(i + 1, n+1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Также был разработан этап обратного хода. Алгоритм вычисляет последнее значения вектора *Х*, и затем в ходе обратного обхода матрицы вычисляется следующее значение вектора решений путём вычисления суммы произведений коэффициентов матрицы на значения уже найденных переменных.

Полученный вектор *Х* является решением системы линейных уравнений.

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n] / a[n][n], precision)  for i in range(n, -1, -1):  s = 0  for j in range(i + 1, n + 1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Программный модуль позволяет выбрать отдельно выбрать точность вычислений. В том числе был разработан вспомогательный модуль для ввода и вывода матриц, который был использован во всех заданиях. Код данного модуля представлен в приложении.

# 

## Результат выполнения контрольного примера

# 

Рис. 1. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения системы линейных уравнений методом исключения неизвестных по столбцам. При этом было замечено, что при вводе параметров с плавающей точкой программа выдает некорректный результат.

# Задача 2. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

В ходе алгоритма вычисляется множитель по формуле (1) для каждой строки и затем с учетом множителя вычитаются из элементов нижестоящей строки по формуле (2). Таким образом матрица приводится к треугольному виду.

| def gauss\_direct(a):  n = len(a)  for i in range(n-1):  for k in range(i+1, n):  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i+1, n+1):  a[k][j] -= round(a[i][j]\*a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Сначала вычисляется значение *Х* вектора, затем для каждой следующей переменной вычисляется сумма произведений коэффициентов матрицы на значения соответствующих переменных вектора. Затем значения вектора вычисляется по формуле (4).

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n+1]/a[n][n], precision)  for i in range(n-1, -1, -1):  s = 0  for j in range(i+1, n+1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Также как и предыдущем модуле присутствует возможность выбора точности вычислений.

## Результат выполнения контрольного примера

# 

Рис. 2. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения СЛУ методом оптимального исключения неизвестных по столбцам.

# Задача 3. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

Алгоритм реализует приведение матрицы к треугольному виду метод Гаусса-Жордана. На каждом шаге цикла программы выполняет деление текущей строки на диагональный элемент, затем вычитает эту строку умноженную на коэффициент из всех остальных строк матрицы.

| def gauss\_jordan(a):  n = len(a)  for i in range(n):  for k in range(n):  if k == i:  continue  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i + 1, n + 1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Решением системы уравнений является вектор значений на главной диагонали. Данная функция собирает необходимые значения в один массив.

| def gauss\_jordan\_result(a):  x = [0] \* len(a)  for i in range(len(a)):  x[i] = round(a[i][-1] / a[i][i], 2)  return x |
| --- |

## Результат выполнения контрольного примера

# 

Рис. 3. Результат выполнения кода

## Вывод

Нами был реализован программный модуль для решения СЛУ методом Гаусса-Жордана.

### Выполнила Васильева М. А. ИВТ 1.1

# Задача 1. Разработать программу по решению СЛУ методом (алгоритм исключения неизвестных по столбцам)

## Математическая модель

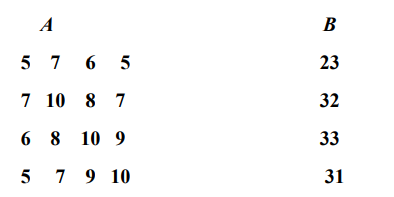
(1)

(2)

(3)

(4)

## Контрольный пример



## Моделирование

На языке Python был реализован этап прямого хода метода Гаусса.

В ходе алгоритма перебираются строки матрицы, и для каждой строки выбирается соответствующий диагональный элемент, а затем значения после делятся на этот элемент.

Во втором цикле из каждой строки вычитается кратное диагональному элементу количество первой строки, чтобы обнулить элементы под диагональю.

| def gauss\_direct(a):  n = len(a)  for i in range(n - 1):  p = a[i][i]  a[i][i] = 1  for j in range(i+1, n+1):  a[i][j] = round(a[i][j] / p, precision)  for k in range(i + 1, n):  for j in range(i + 1, n+1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Также был разработан этап обратного хода. Алгоритм вычисляет последнее значения вектора *Х*, и затем в ходе обратного обхода матрицы вычисляется следующее значение вектора решений путём вычисления суммы произведений коэффициентов матрицы на значения уже найденных переменных.

Полученный вектор *Х* является решением системы линейных уравнений.

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n] / a[n][n], precision)  for i in range(n, -1, -1):  s = 0  for j in range(i + 1, n + 1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Программный модуль позволяет выбрать отдельно выбрать точность вычислений. В том числе был разработан вспомогательный модуль для ввода и вывода матриц, который был использован во всех заданиях. Код данного модуля представлен в приложении.

# 

## Результат выполнения контрольного примера

# 

Рис. 1. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения системы линейных уравнений методом исключения неизвестных по столбцам. При этом было замечено, что при вводе параметров с плавающей точкой программа выдает некорректный результат.

# Задача 2. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

В ходе алгоритма вычисляется множитель по формуле (1) для каждой строки и затем с учетом множителя вычитаются из элементов нижестоящей строки по формуле (2). Таким образом матрица приводится к треугольному виду.

| def gauss\_direct(a):  n = len(a)  for i in range(n-1):  for k in range(i+1, n):  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i+1, n+1):  a[k][j] -= round(a[i][j]\*a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Сначала вычисляется значение *Х* вектора, затем для каждой следующей переменной вычисляется сумма произведений коэффициентов матрицы на значения соответствующих переменных вектора. Затем значения вектора вычисляется по формуле (4).

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n+1]/a[n][n], precision)  for i in range(n-1, -1, -1):  s = 0  for j in range(i+1, n+1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Также как и предыдущем модуле присутствует возможность выбора точности вычислений.

## Результат выполнения контрольного примера

# 

Рис. 2. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения СЛУ методом оптимального исключения неизвестных по столбцам.

# Задача 3. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

Алгоритм реализует приведение матрицы к треугольному виду метод Гаусса-Жордана. На каждом шаге цикла программы выполняет деление текущей строки на диагональный элемент, затем вычитает эту строку умноженную на коэффициент из всех остальных строк матрицы.

| def gauss\_jordan(a):  n = len(a)  for i in range(n):  for k in range(n):  if k == i:  continue  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i + 1, n + 1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Решением системы уравнений является вектор значений на главной диагонали. Данная функция собирает необходимые значения в один массив.

| def gauss\_jordan\_result(a):  x = [0] \* len(a)  for i in range(len(a)):  x[i] = round(a[i][-1] / a[i][i], 2)  return x |
| --- |

## Результат выполнения контрольного примера

# 

Рис. 3. Результат выполнения кода

## Вывод

Нами был реализован программный модуль для решения СЛУ методом Гаусса-Жордана.

### Выполнил Иванов Н. Р. ИВТ 1.1

# Задача 1. Разработать программу по решению СЛУ методом (алгоритм исключения неизвестных по столбцам)

## Математическая модель

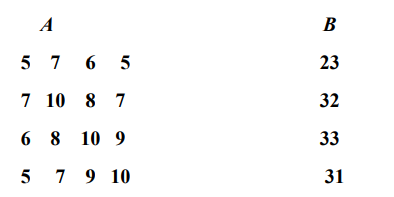
(1)

(2)

(3)

(4)

## Контрольный пример



## Моделирование

На языке Python был реализован этап прямого хода метода Гаусса.

В ходе алгоритма перебираются строки матрицы, и для каждой строки выбирается соответствующий диагональный элемент, а затем значения после делятся на этот элемент.

Во втором цикле из каждой строки вычитается кратное диагональному элементу количество первой строки, чтобы обнулить элементы под диагональю.

| def gauss\_direct(a):  n = len(a)  for i in range(n - 1):  p = a[i][i]  a[i][i] = 1  for j in range(i+1, n+1):  a[i][j] = round(a[i][j] / p, precision)  for k in range(i + 1, n):  for j in range(i + 1, n+1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Также был разработан этап обратного хода. Алгоритм вычисляет последнее значения вектора *Х*, и затем в ходе обратного обхода матрицы вычисляется следующее значение вектора решений путём вычисления суммы произведений коэффициентов матрицы на значения уже найденных переменных.

Полученный вектор *Х* является решением системы линейных уравнений.

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n] / a[n][n], precision)  for i in range(n, -1, -1):  s = 0  for j in range(i + 1, n + 1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Программный модуль позволяет выбрать отдельно выбрать точность вычислений. В том числе был разработан вспомогательный модуль для ввода и вывода матриц, который был использован во всех заданиях. Код данного модуля представлен в приложении.

# 

## Результат выполнения контрольного примера

# 

Рис. 1. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения системы линейных уравнений методом исключения неизвестных по столбцам. При этом было замечено, что при вводе параметров с плавающей точкой программа выдает некорректный результат.

# Задача 2. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

В ходе алгоритма вычисляется множитель по формуле (1) для каждой строки и затем с учетом множителя вычитаются из элементов нижестоящей строки по формуле (2). Таким образом матрица приводится к треугольному виду.

| def gauss\_direct(a):  n = len(a)  for i in range(n-1):  for k in range(i+1, n):  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i+1, n+1):  a[k][j] -= round(a[i][j]\*a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Сначала вычисляется значение *Х* вектора, затем для каждой следующей переменной вычисляется сумма произведений коэффициентов матрицы на значения соответствующих переменных вектора. Затем значения вектора вычисляется по формуле (4).

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n+1]/a[n][n], precision)  for i in range(n-1, -1, -1):  s = 0  for j in range(i+1, n+1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Также как и предыдущем модуле присутствует возможность выбора точности вычислений.

## Результат выполнения контрольного примера

# 

Рис. 2. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения СЛУ методом оптимального исключения неизвестных по столбцам.

# Задача 3. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

Алгоритм реализует приведение матрицы к треугольному виду метод Гаусса-Жордана. На каждом шаге цикла программы выполняет деление текущей строки на диагональный элемент, затем вычитает эту строку умноженную на коэффициент из всех остальных строк матрицы.

| def gauss\_jordan(a):  n = len(a)  for i in range(n):  for k in range(n):  if k == i:  continue  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i + 1, n + 1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Решением системы уравнений является вектор значений на главной диагонали. Данная функция собирает необходимые значения в один массив.

| def gauss\_jordan\_result(a):  x = [0] \* len(a)  for i in range(len(a)):  x[i] = round(a[i][-1] / a[i][i], 2)  return x |
| --- |

## Результат выполнения контрольного примера

# 

Рис. 3. Результат выполнения кода

## Вывод

Нами был реализован программный модуль для решения СЛУ методом Гаусса-Жордана.

### Выполнил Рожков М. В. ИВТ 1.1

# Задача 1. Разработать программу по решению СЛУ методом (алгоритм исключения неизвестных по столбцам)

## Математическая модель

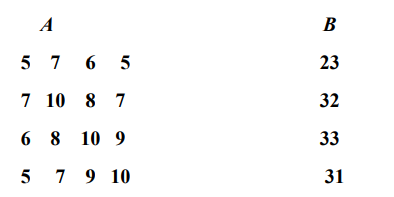
(1)

(2)

(3)

(4)

## Контрольный пример



## Моделирование

На языке Python был реализован этап прямого хода метода Гаусса.

В ходе алгоритма перебираются строки матрицы, и для каждой строки выбирается соответствующий диагональный элемент, а затем значения после делятся на этот элемент.

Во втором цикле из каждой строки вычитается кратное диагональному элементу количество первой строки, чтобы обнулить элементы под диагональю.

| def gauss\_direct(a):  n = len(a)  for i in range(n - 1):  p = a[i][i]  a[i][i] = 1  for j in range(i+1, n+1):  a[i][j] = round(a[i][j] / p, precision)  for k in range(i + 1, n):  for j in range(i + 1, n+1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Также был разработан этап обратного хода. Алгоритм вычисляет последнее значения вектора *Х*, и затем в ходе обратного обхода матрицы вычисляется следующее значение вектора решений путём вычисления суммы произведений коэффициентов матрицы на значения уже найденных переменных.

Полученный вектор *Х* является решением системы линейных уравнений.

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n] / a[n][n], precision)  for i in range(n, -1, -1):  s = 0  for j in range(i + 1, n + 1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Программный модуль позволяет выбрать отдельно выбрать точность вычислений. В том числе был разработан вспомогательный модуль для ввода и вывода матриц, который был использован во всех заданиях. Код данного модуля представлен в приложении.

# 

## Результат выполнения контрольного примера

# 

Рис. 1. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения системы линейных уравнений методом исключения неизвестных по столбцам. При этом было замечено, что при вводе параметров с плавающей точкой программа выдает некорректный результат.

# Задача 2. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

В ходе алгоритма вычисляется множитель по формуле (1) для каждой строки и затем с учетом множителя вычитаются из элементов нижестоящей строки по формуле (2). Таким образом матрица приводится к треугольному виду.

| def gauss\_direct(a):  n = len(a)  for i in range(n-1):  for k in range(i+1, n):  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i+1, n+1):  a[k][j] -= round(a[i][j]\*a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Сначала вычисляется значение *Х* вектора, затем для каждой следующей переменной вычисляется сумма произведений коэффициентов матрицы на значения соответствующих переменных вектора. Затем значения вектора вычисляется по формуле (4).

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n+1]/a[n][n], precision)  for i in range(n-1, -1, -1):  s = 0  for j in range(i+1, n+1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Также как и предыдущем модуле присутствует возможность выбора точности вычислений.

## Результат выполнения контрольного примера

# 

Рис. 2. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения СЛУ методом оптимального исключения неизвестных по столбцам.

# Задача 3. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

Алгоритм реализует приведение матрицы к треугольному виду метод Гаусса-Жордана. На каждом шаге цикла программы выполняет деление текущей строки на диагональный элемент, затем вычитает эту строку умноженную на коэффициент из всех остальных строк матрицы.

| def gauss\_jordan(a):  n = len(a)  for i in range(n):  for k in range(n):  if k == i:  continue  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i + 1, n + 1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Решением системы уравнений является вектор значений на главной диагонали. Данная функция собирает необходимые значения в один массив.

| def gauss\_jordan\_result(a):  x = [0] \* len(a)  for i in range(len(a)):  x[i] = round(a[i][-1] / a[i][i], 2)  return x |
| --- |

## Результат выполнения контрольного примера

# 

Рис. 3. Результат выполнения кода

## Вывод

Нами был реализован программный модуль для решения СЛУ методом Гаусса-Жордана.

### Выполнил Шардт М. А. ИВТ 1.1

# Задача 1. Разработать программу по решению СЛУ методом (алгоритм исключения неизвестных по столбцам)

## Математическая модель

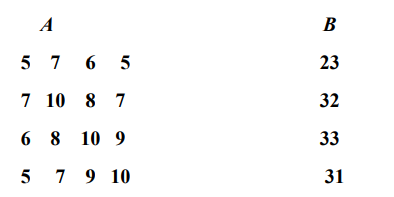
(1)

(2)

(3)

(4)

## Контрольный пример



## Моделирование

На языке Python был реализован этап прямого хода метода Гаусса.

В ходе алгоритма перебираются строки матрицы, и для каждой строки выбирается соответствующий диагональный элемент, а затем значения после делятся на этот элемент.

Во втором цикле из каждой строки вычитается кратное диагональному элементу количество первой строки, чтобы обнулить элементы под диагональю.

| def gauss\_direct(a):  n = len(a)  for i in range(n - 1):  p = a[i][i]  a[i][i] = 1  for j in range(i+1, n+1):  a[i][j] = round(a[i][j] / p, precision)  for k in range(i + 1, n):  for j in range(i + 1, n+1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Также был разработан этап обратного хода. Алгоритм вычисляет последнее значения вектора *Х*, и затем в ходе обратного обхода матрицы вычисляется следующее значение вектора решений путём вычисления суммы произведений коэффициентов матрицы на значения уже найденных переменных.

Полученный вектор *Х* является решением системы линейных уравнений.

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n] / a[n][n], precision)  for i in range(n, -1, -1):  s = 0  for j in range(i + 1, n + 1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Программный модуль позволяет выбрать отдельно выбрать точность вычислений. В том числе был разработан вспомогательный модуль для ввода и вывода матриц, который был использован во всех заданиях. Код данного модуля представлен в приложении.

# 

## Результат выполнения контрольного примера

# 

Рис. 1. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения системы линейных уравнений методом исключения неизвестных по столбцам. При этом было замечено, что при вводе параметров с плавающей точкой программа выдает некорректный результат.

# Задача 2. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

В ходе алгоритма вычисляется множитель по формуле (1) для каждой строки и затем с учетом множителя вычитаются из элементов нижестоящей строки по формуле (2). Таким образом матрица приводится к треугольному виду.

| def gauss\_direct(a):  n = len(a)  for i in range(n-1):  for k in range(i+1, n):  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i+1, n+1):  a[k][j] -= round(a[i][j]\*a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Сначала вычисляется значение *Х* вектора, затем для каждой следующей переменной вычисляется сумма произведений коэффициентов матрицы на значения соответствующих переменных вектора. Затем значения вектора вычисляется по формуле (4).

| def gauss\_reverse(a):  n = len(a) - 1  x = [0] \* len(a)  x[n] = round(a[n][n+1]/a[n][n], precision)  for i in range(n-1, -1, -1):  s = 0  for j in range(i+1, n+1):  s += a[i][j] \* x[j]  x[i] = round((a[i][-1] - s) / a[i][i], precision)  return x |
| --- |

Также как и предыдущем модуле присутствует возможность выбора точности вычислений.

## Результат выполнения контрольного примера

# 

Рис. 2. Результаты вычислений

## Вывод

Нами был реализован программный модуль для решения СЛУ методом оптимального исключения неизвестных по столбцам.

# Задача 3. Модифицировать программу по решению СЛУ методом Гаусса для реализации алгоритма оптимального исключения неизвестных

## Математическая модель

(1)

(2)

(3)

(4)

## Моделирование

Алгоритм реализует приведение матрицы к треугольному виду метод Гаусса-Жордана. На каждом шаге цикла программы выполняет деление текущей строки на диагональный элемент, затем вычитает эту строку умноженную на коэффициент из всех остальных строк матрицы.

| def gauss\_jordan(a):  n = len(a)  for i in range(n):  for k in range(n):  if k == i:  continue  a[k][i] = round(a[k][i] / a[i][i], precision)  for j in range(i + 1, n + 1):  a[k][j] -= round(a[i][j] \* a[k][i], precision)  a[k][i] = 0  return a |
| --- |

Решением системы уравнений является вектор значений на главной диагонали. Данная функция собирает необходимые значения в один массив.

| def gauss\_jordan\_result(a):  x = [0] \* len(a)  for i in range(len(a)):  x[i] = round(a[i][-1] / a[i][i], 2)  return x |
| --- |

## Результат выполнения контрольного примера

# 

Рис. 3. Результат выполнения кода

## Вывод

Нами был реализован программный модуль для решения СЛУ методом Гаусса-Жордана.

# 

# *Приложение*

## Вспомогательные функции

| def matrix\_print(a):  col\_indent = {}  for row in a:  for (i, col) in enumerate(row):  prev = col\_indent.get(i, 0)  cur = len(f'{col:g}')  col\_indent[i] = max(cur, prev)   for row in a:  for (i, col) in enumerate(row):  if i == len(a[0]) - 1:  print(" |", end='')  indent = (col\_indent[i] - len(f'{col:g}') + 1) \* ' '  print(f'{indent}{col:g}', end='')   print()   def matrix\_from\_values(height, width, values):  a = []  for \_ in range(height):  t = []  for \_ in range(width):  t.append(values.pop(0))  a.append(t)  return a   def matrix\_input():  n = int(input("Введите высоту матрицы: "))  print("Введите матрицу: ")  result = []  while len(result) < n:  values = input().strip()  line = list(map(lambda x: float(x), values.split(' ')))  result.append(line)  return result |
| --- |

## Исходный код

<https://replit.com/@MaximSchardt/TKM1>