|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«*Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |  |
| --- | --- | --- |
| ФАКУЛЬТЕТ |  | ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ |
| КАФЕДРА |  | КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ |
| ГРУППА |  | ИУ6-61Б |

**ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА**

**Отчет по домашнему заданию №1**

**Прямые методы решения СЛАУ**

**Вариант № 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент: |  | 12.04.2025 |  | А.В. Аткин |
|  |  | (дата, подпись) |  | (И. О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Я. Ю. Павловский |
|  |  | (дата, подпись) |  | (И. О. Фамилия) |

Москва, 2025

**СОДЕРЖАНИЕ**

[Цель домашней работы 3](#_Toc194865886)

[Постановка задачи и исходные данные 3](#_Toc194865887)

[Краткое описание реализуемых методов 4](#_Toc194865888)

[Текст программы 4](#_Toc194865889)

[Результаты выполнения программы 8](#_Toc194865890)

[Анализ результатов 9](#_Toc194865891)

## Цель домашней работы

Изучения методов Гаусса, Хаусхолдера численного решения квадратной СЛАУ с невырожденной матрицей, оценка числа обусловленности матрицы и исследования его влияния на погрешность приближенного решения. Изучения метода прогонки решения СЛАУ с трехдиагональной матрицей.

## Постановка задачи и исходные данные

## Необходимо реализовать методы Гаусса, Хаусхолдера и прогонки для решения систем линейных алгебраических уравнений (СЛАУ). Выполнить расчёт двух квадратных СЛАУ размером 4×4, а также одной трёхдиагональной системы. Для каждого решения вычислить нормы невязок, определить абсолютные и относительные погрешности, найти обратные матрицы и оценить число обусловленности.

Ниже приведены расширенные матрицы систем для каждого варианта.  
Справа от символов @ даны компоненты векторов точных решений систем.

Листинг 1 – Исходные данные

|  |
| --- |
| 1. Хорошо обусловленная матрица  -52.4000 0.0000 -0.5700 4.7300 -1309.1700 @ 34  0.1200 32.4000 9.0500 0.4900 224.1300 @ 5  0.0000 5.8800 -175.0000 2.4300 97.4000 @ 1  -5.0100 -2.4300 1.8700 -76.2000 -7800.6200 @ 100  2. Плохо обусловленная матрица  -558.5500 2569.1600 12.6000 55.8600 127.3100 @ 1  -139.6500 642.3400 3.1500 13.9650 31.8150 @ 0  -19.9500 91.7700 0.4000 1.9950 2.0450 @ 50  838.3900 -3855.9940 -18.9000 -83.7890 -190.3990 @ 1 |

Ниже приведены коэффициенты трехдиагольных СЛАУ для каждого варианта

1-я строка: компоненты вектора a=(a\_2, a\_3, ..., a\_n) (подддиагональ заполняется со второго элемента)

2-я строка: компоненты вектора с=(с\_1, с\_2, ..., с\_n) (ддиагональ заполняется полностью)

3-я строка: компоненты вектора b=(b\_1, b\_2, ..., b\_(n-1) ) (наддиагональ заполняется без последнего элемента)

4-я строка: компоненты вектора d=(d\_1, d\_2, ..., d\_n) правых частей

Листинг 2 – Исходные данные

|  |
| --- |
| 1 1 -1 2 -1  60 80 130 -90 140 70  1 1 -2 1 -1  6 7 13 -8 15 9 |

## Краткое описание реализуемых методов

Метод Гаусса — это классический подход к решению систем линейных алгебраических уравнений, при котором последовательным исключением неизвестных матрица приводится к верхнетреугольному виду.

Метод Хаусхолдера — способ решения СЛАУ, использующий ортогональные преобразования посредством отражений Хаусхолдера для приведения исходной матрицы к треугольной форме.

Метод прогонки — специализированная процедура для решения систем с трёхдиагональными матрицами, состоящая из двух этапов: прямого и обратного проходов.

## Текст программы

Ниже в листингах приведены листинги кода проекта, реализованного на языке python версии 3.10

Метод Гаусса (файл gauss.py) представлен в листинге 3.

Листинг 3 – Функция gauss

|  |
| --- |
| def gauss(A, b):  A, b = A.astype(float), b.astype(float)  for i in range(len(b)):  max\_row = np.argmax(abs(A[i:, i])) + i  A[[i, max\_row]] = A[[max\_row, i]]  b[[i, max\_row]] = b[[max\_row, i]]  for j in range(i+1, n):  ratio = A[j][i] / A[i][i]  A[j, i:] -= ratio \* A[i, i:]  b[j] -= ratio \* b[i]  x = np.zeros(n)  for i in range(n-1, -1, -1):  x[i] = (b[i] - np.dot(A[i, i+1:], x[i+1:])) / A[i, i]  return x |

Метод Хаусхолдера (файл householder.py) представлен в листинге 4.

Листинг 4 – Функция householder

|  |
| --- |
| def householder(A, b):  A = A.astype(float)  b = b.astype(float)  m, n = A.shape  for k in range(n):  x = A[k:, k]  e = np.zeros\_like(x)  e[0] = np.linalg.norm(x) \* (-1 if x[0] < 0 else 1)  u = x + e  v = u / np.linalg.norm(u)  A[k:, k:] -= 2.0 \* np.outer(v, v @ A[k:, k:])  b[k:] -= 2.0 \* v \* (v @ b[k:])  x = np.zeros(n)  for i in range(n - 1, -1, -1):  x[i] = (b[i] - A[i, i + 1:] @ x[i + 1:]) / A[i, i]  return x |

Метод прогонки (файл progonka.py) представлен в листинге 5.

Листинг 5 – Функция progonka

|  |
| --- |
| def progonka(a, c, b, d):  n = lenI  alpha = np.zeros(n)  beta = np.zeros(n)  alpha[0] = -b[0] / c[0]  beta[0] = d[0] / c[0]  for I in range(1, n):  denom = c[i] + a[i-1] \* alpha[i-1]  if I < n – 1:  alpha[i] = -b[i] / denom  beta[i] = (d[i] – a[i-1] \* beta[i-1]) / denom  x = np.zeros(n)  x[-1] = beta[-1]  for I in reversed(range(n – 1)):  x[i] = alpha[i] \* x[I + 1] + beta[i]  return x |

После того, как были написаны основные функции, необходимые для обсчёта матриц 3 разными способами, необходимо было их проверить на заданных нам матрицам, на основе которых впоследствии мы проведём анализ используемых методов.

Главный файл с исходными данными (main.py) представлен в листинге 6.

Листинг 6 – Код основной программы

|  |
| --- |
| import numpy as np  from numpy.linalg import norm, inv, cond  from gauss import gauss  from householder import householder  from progonka import progonka  def run\_all\_methods(A, b, x\_exact, label):  print(f"\n===== {label} =====")    for method\_name, solver in [("Метод Гаусса", gauss), ("Метод Хаусхолдера", householder)]:  x = solver(A.copy(), b.copy())  r = A @ x - b  error = x - x\_exact  print(f"\n{method\_name}")  print("x =", x)  print("1-норма невязки:", norm(r, 1))  print("∞-норма невязки:", norm(r, np.inf))  print("1-норма погрешности:", norm(error, 1))  print("∞-норма погрешности:", norm(error, np.inf))  relative\_error\_1 = norm(error, 1) / norm(x\_exact, 1)  relative\_error\_inf = norm(error, np.inf) / norm(x\_exact, np.inf)  print("Относительная 1-норма погрешности:", relative\_error\_1)  print("Относительная ∞-норма погрешности:", relative\_error\_inf)  # Обратная матрица и проверка  A\_inv = inv(A)  I\_approx = A\_inv @ A  print("\nОбратная матрица A⁻¹:")  print(A\_inv)  print(f"Норма (E - A⁻¹A): {norm(np.eye(len(A)) - A\_inv @ A):.3e}")    # Обусловленность  cond\_1 = cond(A, 1)  cond\_inf = cond(A, np.inf)  print(f"cond\_1(A): {cond\_1:.3e}")  print(f"cond\_inf(A): {cond\_inf:.3e}")  if cond\_1 < 100:  print("➤ Матрица хорошо обусловлена.")  else:  print("➤ Матрица плохо обусловлена. Результаты могут быть неточными.")  # ============ Вариант 2 ============  # 1. Хорошо обусловленная система  A1 = np.array([  [-52.4, 0.0, -0.57, 4.73],  [0.12, 32.4, 9.05, 0.49],  [0.0, 5.88, -175.0, 2.43],  [-5.01, -2.43, 1.87, -76.2]  ])  b1 = np.array([-1309.17, 224.13, 97.4, -7800.62])  x1\_exact = np.array([34, 5, 1, 100])  run\_all\_methods(A1, b1, x1\_exact, "Система 1: Хорошо обусловленная")  # 2. Плохо обусловленная система  A2 = np.array([  [-558.5500, 2569.1600, 12.6000, 55.8600],  [-139.6500, 642.3400, 3.1500, 13.9650],  [-19.9500, 91.7700, 0.4000, 1.9950],  [838.3900, -3855.9940, -18.9000, -83.7890]  ])  b2 = np.array([127.31, 31.815, 2.045, -190.399])  x2\_exact = np.array([1, 0, 50, 1])  run\_all\_methods(A2, b2, x2\_exact, "Система 2: Плохо обусловленная")  # -----  print()  print()  # Данные варианта 2  a = np.array([1 , 1, -1 , 2 ,-1], dtype=float) # длина = n-1  c = np.array([60 , 80, 130 , -90 , 140 , 70], dtype=float) # длина = n  b = np.array([1 , 1 ,-2 , 1, -1], dtype=float) # длина = n-1  d = np.array([6 , 7 , 13, -8 , 15, 9], dtype=float) # длина = n  # Решение методом прогонки  x = progonka(a, c, b, d)  # Собираем A для проверки (только для проверки!)  n = len(c)  A = np.zeros((n, n))  for i in range(n):  A[i, i] = c[i]  if i > 0:  A[i, i - 1] = a[i - 1]  if i < n - 1:  A[i, i + 1] = b[i]  # Вычисление невязки  r = A @ x - d  print("Решение методом прогонки:")  print("x =", x)  print("1-норма невязки:", np.linalg.norm(r, 1))  print("∞-норма невязки:", np.linalg.norm(r, np.inf)) |

## **Результаты выполнения программы**

Ниже представлены результаты выполнения файла main.py

Рисунок 1 – Результаты выполнения программы

Рисунок 2 – Результаты выполнения программы

## Анализ результатов

Все исследуемые методы дали корректные результаты:

* **Методы Гаусса и Хаусхолдера** продемонстрировали высокую точность на хорошо обусловленных матрицах: значения невязок и погрешностей были близки к машинному нулю.
* Для **плохо обусловленной системы** отмечается значительное влияние ошибок округления, что подтверждается большим числом обусловленности матрицы.
* **Метод прогонки** отлично справился с решением трёхдиагональной системы, обеспечив высокую точность результатов.