Лабораторная работа №3

Арцыбашева Григория, КН-301

Оглавление

[Введение 2](#_Toc125908452)

[Отбрасывание знаков 3](#_Toc125908453)

[Обзор и реализация методов решения 4](#_Toc125908454)

[1) Компактная схема Гаусса 4](#_Toc125908455)

[2) Метод Гаусса с выбором главного элемента 6](#_Toc125908456)

[Результаты вычислений 8](#_Toc125908457)

[Вывод 10](#_Toc125908458)

# Введение

Дано матричное уравнение Ему соответствует система:

Где

Тогдаи система уравнений имеет вид

Точное решение системы:

Необходимо решить систему, используя:

1. компактную схему Гаусса;
2. метод Гаусса с выбором главного элемента,

проводя все вычисления с 2, 4, 6 знаками после запятой.

# Отбрасывание знаков

При каждой арифметической операции вызывается функция которая отбрасывает все цифры после запятой у числа , начиная   
с -ой.

Программный код (здесь и далее используется язык программирования Python 3):

def cut(num: float, k: int) -> float:  
 s = pow(10, k)  
 return floor(num \* s) / s

# Обзор и реализация методов решения

## Компактная схема Гаусса

Первый шаг процедуры – решение задачи факторизации исходной матрицы системы, то есть представление матрицы в виде произведения треугольных матриц и , причём  
 – нижнетреугольная матрица, а – верхнетреугольная матрица   
с единицами на главной диагонали. Элементы матриц вычисляются   
по следующим формулам (здесь n – размерность матрицы исходной системы):

С помощью решения задачи факторизации исходную задачу можно свести к решению двух систем с треугольными матрицами:

Из первой системы находятся все (формула та же, что и для ):

где – элемент -ой строки матрицы-столбца .

Далее решается система. Матрица системы верхнетреугольная, вычисление производится по формуле:

Программный код, вычисляющий , и :

def calc\_b\_ij(  
 A: list[list[float]],  
 B: list[list[float]],  
 C: list[list[float]],  
 ij: tuple[int, int],  
 k: int) \  
 -> float:  
 i, j = ij  
 res = A[i][j]  
 for t in range(j):  
 res = cut(res - cut(B[i][t] \* C[t][j], k), k)  
 return cut(res, k)

def calc\_c\_ij(  
 A: list[list[float]],  
 B: list[list[float]],  
 C: list[list[float]],  
 ij: tuple[int, int],  
 k: int) \  
 -> float:  
 i, j = ij  
 res = A[i][j]  
 for t in range(i):  
 res = cut(res - cut(B[i][t] \* C[t][j], k), k)   
 return cut(res / B[i][i], k)

def calc\_x(  
 Y: list[float],  
 C: list[list[float]],  
 X: list[float],  
 i: int,  
 n: int,  
 k: int) \  
 -> float:  
 res = Y[i]  
 for t in range(i, n - 1):  
 res = cut(res - cut(C[i][t + 1] \* X[t + 1], k), k)  
 return cut(res, k)

Программный код компактной схемы Гаусса:

def compact\_scheme\_gauss(ext\_A: list[list[float]], k: int) -> list[float]:  
 *"""Компактная схема Гаусса"""* # Размерность матрицы A  
 n = len(ext\_A)  
  
 # Вычисление матриц B и C  
 B = [[0.0] \* n for \_ in range(n)]  
 C = [[0.0] \* i + [1.0] + [0.0] \* (n - i - 1) for i in range(n)]  
 for i in range(n):  
 for j in range(n):  
 if i >= j:  
 B[i][j] = calc\_b\_ij(ext\_A, B, C, (i, j), k)  
 else:  
 C[i][j] = calc\_c\_ij(ext\_A, B, C, (i, j), k)  
  
 # Вычисление матрицы-столбца Y  
 raw\_Y = [[0.0] \* (n + 1) for \_ in range(n)]  
 for i in range(n):  
 raw\_Y[i][n] = calc\_c\_ij(ext\_A, B, raw\_Y, (i, n), k)  
 Y = list(map(lambda y: y[n], raw\_Y))  
  
 # Вычисление матрицы-столбца X  
 X = [0.0 for \_ in range(n)]  
 for i in range(n - 1, -1, -1):  
 X[i] = calc\_x(Y, C, X, i, n, k)  
 return X

## Метод Гаусса с выбором главного элемента

На -ом шаге метода сначала выбирается главный элемент среди -го столбца расширенной матрицы системы, затем -ая строка нормируется и происходит исключение элементов -го столбца. В результате получается треугольная матрица, в последней строке которой находится уравнение . Далее найденные на предыдущем шаге последовательно подставляются в уравнения получившейся матрицы, тем самым находится решение исходной системы.

Программный код, реализующий метод Гаусса с выбором главного элемента:

def method\_gauss(ext\_A: list[list[float]], k: int) -> list[float]:  
 *"""Метод Гаусса с выбором главного элемента"""* # Размерность матрицы A  
 n = len(ext\_A)  
  
 for i in range(n):  
 # Ищем главный элемент в i-ом столбце  
 line\_num, main\_it = max(  
 [(line, abs(ext\_A[line][i])) for line in range(i, n)],  
 key=lambda z: z[-1]  
 )  
  
 # Переставляем i-ую и line\_num-ую строки местами  
 tmp = ext\_A[line\_num]  
 ext\_A[line\_num] = ext\_A[i]  
 ext\_A[i] = tmp  
  
 # Нормировка  
 ext\_A[i] = list(map(lambda z: cut(z / main\_it, k), ext\_A[i]))  
 # Исключение элементов i-го столбца  
 for j in range(i + 1, n):  
 a\_ji = ext\_A[j][i]  
 for u in range(n + 1):  
 ext\_A[j][u] = cut(  
 ext\_A[j][u] - cut(ext\_A[i][u] \* a\_ji, k),  
 k)  
 # Вычисление матрицы-столбца X  
 X = [0.0 for \_ in range(n)]  
 for i in range(n - 1, -1, -1):  
 X[i] = ext\_A[i][n]  
 for j in range(n - 1, i, -1):  
 X[i] = cut(X[i] - cut(ext\_A[i][j] \* X[j], k), k)  
 return X

# Результаты вычислений

– решения системы, полученные с помощью компактной схемы Гаусса и метода Гаусса с выбором главного элемента соответственно, в которых все вычисления проводились с знаками после запятой ().

|  |  |  |
| --- | --- | --- |
| Компактная схема Гаусса | | |
|  |  |  |
| 2 |  | 2.33072949953442 |
| 4 |  | 2.33820734110557 |
| 6 |  | 1.48258741263373 |
| Метод Гаусса с выбором главного элемента | | |
|  |  |  |
| 2 |  | 5.74456264653803 |
| 4 |  | 4.58257569495582 |
| 6 |  |  |

Для вычисления нормы использовался следующий программный код:

def calc\_norm(vector: list[float]) -> float:  
 *"""Вычисляет норму"""* return sqrt(sum(map(lambda x: pow(x, 2), vector)))

# Вывод

Округление чисел до цифр после запятой является возмущением параметров системы. Метод Гаусса с выбором главного элемента при разных дал примерно одинаковый результат, близкий к точному решению.   
В этом случае малые возмущения параметров привели к малому возмущению решения, что даёт основания для предположения, что система хорошо обусловлена.

В компактной схеме Гаусса при разных решения сильно отличаются. Это происходит из-за потери точности при выполнении арифметических действий, таких, как деление на число, близкое к нулю.

Число обусловленности матрицы . Это подтверждает предположение, что исходная система хорошо обусловлена.