Министерство высшего образования и науки Российской Федерации

Национальный научно-исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по дисциплине

«Вычислительная математика»

Вариант 11

Работу выполнил:

Макеев Роман Ильич

Группа P3208

Преподаватель:

Машина Екатерина Алексеевна

Санкт-Петербург

2024

Цель работы:

Написать программу вычисляющую решение СЛАУ размерности до 20 методом Гаусса с выбором главного элемента

Описание метода и расчетные формулы:

Проверим что в матрице нет нулевых столбов

Прямой ход:

- Проходимся по каждой строке СЛАУ.
- Для i-той строки меняем ее местами с j-той строкой, у которой a_{ji} наибольший
- Исключим x_i из каждой строки ниже i-той, домножив i-тую строку на $(-\frac{a_{ji}}{a_{ii}})$ и сложив ее с j-той строкой
- Матрица приведена к треугольн $detA = (-1)^k \prod_{i=1}^{n} a_{ii}$

Можем вычислить определитель:

Если $det = 0$, решений бесконечно много или их нет

Обратный ход:

- Считаем $x_n = {b_n^{(n-1)}}/{a_{nn}^{(n-`1)}}$
- Подставляем x_n в строку $n - 1$, аналогично считаем x_{n-1} и тд

Вычисляем вектор невязок, подставив найденные решения в исходное уравнение $\boldsymbol{невязка\ r = Ax^* - b}$

Листинг программы:

```
# Приведение к треугольному виду
1 usage  gl4zis
def triangle(self) -> None:
    perm_count = 0
    triangled_eq: Equation = self.copy()
    for i in range(triangled_eq.size() - 1):
        # Выбор главного элемента
        max_row_idx: int = i
        for j in range(i, triangled_eq.size()):
            if abs(triangled_eq.matrix[j][i]) > abs(triangled_eq.matrix[max_row_idx][i]):
                max_row_idx = j

        # Перестановка строк
        if max_row_idx != i:
            triangled_eq.matrix.swap_rows(i, max_row_idx)
            triangled_eq.b_vector.swap_elems(i, max_row_idx)
            perm_count += 1

        # Убираем x[i][i] из каждой строки ниже i
        for j in range(i + 1, triangled_eq.size()):
            kf: float = - triangled_eq.matrix[j][i] / triangled_eq.matrix[i][i]

            adding_vec = triangled_eq.matrix[i] * kf
            triangled_eq.matrix[j] += adding_vec

            adding_b = kf * triangled_eq.b_vector[i]
            triangled_eq.b_vector[j] += adding_b

    self.triangled: Equation = triangled_eq
    self.set_det(perm_count)
```

```
# Считает детерминант треугольной матрицы
1 usage  gl4zis
def set_det(self, k: int) -> None:
    self.det: float = 1
    for i in range(self.size()):
        self.det *= self.triangled.matrix[i][i]

    self.det *= ((-1) ** k)

# Решает уравнение, считает вектор ответов
1 usage  gl4zis
def calc_answers(self) -> None:
    if not self.matrix.is_correct():
        return

    self.triangle()

    if self.det == 0:
        return

    answers: Vector = Vector(self.size())
    for i in range(self.size() - 1, -1, -1):
        answer = self.triangled.b_vector[i]
        for j in range(self.size() - 1, i, -1):
            answer -= self.triangled.matrix[i][j] * answers[j]
        answers[i] = answer / self.triangled.matrix[i][i]

    self.answers = answers
```

```python
# Считает вектор невязок
1 usage  gl4zis
def calc_residuals(self) -> None:
    residuals: Vector = Vector(self.size())

    if self.answers is None:
        return

    for i in range(self.size()):
        residuals[i] = (self.matrix[i] * self.answers).sum() - self.b_vector[i]

    self.residuals = residuals

1 usage  gl4zis
def solve(self) -> None:
    self.calc_answers()

    if self.answers is None:
        print('Invalid matrix! No answers or any answer')
        return

    self.calc_residuals()

    print(f'Determinant = {self.det:g}')
    print()
    print(f'Triangled equation:\n{self.triangled}')
    self.print_answers()
    print()
    self.print_residuals()
```

Примеры и результаты работы программы:

1)

```
10 -7 0 7
-3 2,099 6 3.901
5 -1 5 6
```

```
Determinant = -150.05

Triangled equation:
10  -7  0    =   7
0   2.5 5    =   2.5
0   0   6.002   =   6.002

Answers:
    x1 = 2.6645e-16
    x2 = -1
    x3 = 1

Residuals:
    r1 = 0
    r2 = -8.8818e-16
    r3 = 0
```

2)

```
Enter line by line 'a1 a2 ... an b' separated by a space:
1 2 3 4 5
1 2 3 4 5
11 12 13 14 15
21 22 23 24 25
Invalid matrix! No answers or any answer
```

Вывод:

Реализована программа, вычисляющая решение СЛАУ методом Гаусса с выбором главного элемента с большой точностью.

Преимущество этого метода перед обычным методом Гаусса заключаются в гораздо большей точности вычислений с малыми дробными коэффицентами.

В отличие от итерационных методов, в прямых можно заранее сказать сколько потребуется времени на вычисление результата, а также прямые методы могут давать большую точность при меньшем количестве операций.

Однако, при использовании итерационных методов можно вручную задать точность вычислений и не тратить много памяти на хранение всей СЛАУ.