Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Лабораторная работа №3

студента 5 курса 531 группы
специальности 10.05.01 «Компьютерная безопасность»
факультета компьютерных наук и информационных технологий
Енца Михаила Владимировича

Преподаватель
профессор _____ В. А. Молчанов
подпись, дата

Заведующий кафедрой
д.ф.-м.н., доцент _____ М. Б. Абросимов
подпись, дата

Саратов 2018

1. Постановка задачи.

Изучение основных методов решения систем линейных уравнений над конечными полями и их программная реализация.

2. Теоретические сведения по рассмотренным темам с их обоснованием.

Рассматриваем системы из p линейных алгебраических уравнений с n неизвестными переменными вида:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{p1}x_1 + a_{p2}x_2 + \cdots + a_{pn}x_n = b_p \end{cases}$$

где $x_1, x_2, \dots, x_n$ – неизвестные переменные, a_{ij}, $i = 1,2, \dots, p$, $j = 1,2, \dots, n$ – коэффициенты, $b_1, b_2, \dots, b_p$ – свободные члены. Такую форму записи СЛАУ называют координатной.

В матричной форме записи эта система уравнений имеет следующий вид $A * X = B$, где $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & \dots & a_{pn} \end{pmatrix}$ – основная матрица системы, $X = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$ – матрица столбец неизвестных переменных, $B = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_p \end{pmatrix}$ – матрица столбец свободных членов.

Если к матрице A добавить в качестве $n + 1$ ого столбца матрицу столбец свободных членов, то получим так называемую расширенную матрицу системы линейных уравнений.

Решением системы линейных алгебраических уравнений называют набор значений неизвестных переменных, обращающий все уравнения системы в тождества. Матричное уравнение при данных значениях неизвестных переменных так же обращается в тождество.

Метод Гаусса решения систем линейных

Классический метод решения системы линейных алгебраических уравнений (СЛАУ).

Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к системе диагонального вида.

Псевдокод:

Вход. n – количество уравнений системы, m – модуль, M – расширенная матрица системы линейных уравнений.

Выход. Общее решение системы линейных уравнений.

1. Положить $i \leftarrow 0$.

2. Пока i не равно n:

2.1. Выбираем i-ю строку матрицы M и элемент $M[i][i]$.

2.2. Проверяем, что выбранный элемент $M[i][i] \neq 0$, иначе производим такую модификацию матрицы, чтобы на данном месте оказался ненулевой элемент.

2.3. Умножаем выбранную строку на обратный элемент в поле для элемента, выбранного на предыдущем шаге.

2.4. Рассматриваем оставшиеся сверху и снизу строки матрицы M. И выполняем следующие действия.

2.4.1. Выбираем i-ый элементы k-ой строки.

2.4.2. Далее умножаем i-ую строку на выбранный на предыдущем шаге элемент и складываем с k-ой строкой.

2.5. Проверяем, что среди строк матрицы нет нулевых, если есть, то удаляем их.

2.6. Проверяем матрицу на наличие противоречий.

2.7. Возвращаем общее решение.

Алгоритм Ланцоша.

Пусть K – поле, A – матрица размера $n * n$ над K, b – n-мерный вектор, $b \neq 0$. Мы хотим решить систему линейных уравнений $Ax = b$.

Предположим дополнительно, что матрица A – симметричная, а вектор b удовлетворяет условию $b^T ab \neq 0$. Рассмотрим последовательность Крылова S, состоящую из n-мерных векторов s_0, s_1 ..., где $s_0 = b, s_i = A^i b = As_{i-1}, i = 1,2, \ldots$

Лемма. Пусть $m \in N$, $s_0 \ldots s_{m-1}$ – линейно независимы над K, а $s_0, \ldots, s_m$ – линейно зависимы. Тогда любой элемент из множества S линейно выражается через $s_0, \ldots, s_{m-1}$.

Лемма. Пусть процесс ортогонализации закончился вектором w_k. Если $w_k = 0$, то $k = m$; если же $w_k \neq 0$, то $k < m$.

Лемма. Пусть процесс ортогонализации завершился при $k = m$, при этом были построены линейно независимые векторы $w_0, \ldots, w_{m-1}$, а $w_m = 0$. Тогда $x = \sum_{i=0}^{r} \frac{w_i, b}{w_i, w_i} w_i$ при $r = m - 1$ дает решение $Ax = b$.

Алгоритм Ланцоша работает следующим образом. Мы вычисляем последовательность векторов $w_0 = b, w_1, w_2$, ..., пользуясь формулами

$$\begin{cases} w_0' = s_0 = b \\ w_1' = Aw_0' - \alpha_{10} w_0{}' \\ w_i' = Aw_{i-1}' - \sum_{j=0}^{i-1} \alpha_{ij} w_j' \end{cases}$$

$$\alpha_{ij} = \frac{\left(Aw_{i-1}', w_j'\right)_A}{\left(w_j', w_j'\right)_A}$$

$$w_i = Aw_{i-1} - \frac{(w_{i-1}, Aw_{i-1})_A}{(w_{i-1}, w_{i-1})_A} w_{i-1} - \frac{(w_{i-1}, Aw_{i-2})_A}{(w_{i-2}, w_{i-2})_A} w_{i-2}$$

Если на некотором шаге будет построен вектор $w_i \neq 0$ такой, что $(w_i, w_i)_A = 0$, то мы не сможем найти решение $Ax = b$ этим методом. Если же будет построен вектор $w_m = 0$, то решение x мы находим по формуле $x = \sum_{i=0}^{r} \frac{w_i, b}{w_i, w_i} w_i$ при $r = m - 1$. При этом следует сделать проверку, поскольку мы предполагаем выполнение некоторых условий.

В алгоритмах факторизации и дискретного логарифмирования матрица A системы $Ax = b$ не является симметричной. В этом случае предполагается

случайным образом выбрать диагональную матрицу D и рассмотреть систему уравнений $A^T D^2 Ax = A^T D^2 b$.

Ее матрица $\mathrm{A}^{\mathrm{T}} D^2 A = (DA)^T DA$ будет квадратной и симметричной, и к системе $A^T D^2 Ax = A^T D^2 b$ мы затем применяем алгоритм Ланцоша. Если алгоритм закончится неудачей либо найденный вектор x не будет решением $Ax = b$, то следует выбрать другую матрицу D.

Однородная система $Ax = 0$ может быть преобразована в неоднородную в предположении, что в ее решении $(x_1, \dots, x_n)$ последняя координата $x_n \neq 0$. Положив тогда $x_n = 1$, мы сможем перенести столбец соответствующих x_n коэффициентов в правую часть и затем применить алгоритм Ланцоша.

Алгоритм Видемана

Алгоритм 1.

1. Присвоить $b_0 = b, k = 0, y_0 = 0, d_0 = 0$.

2. Если $b_k = 0$, то решение $Ax = b, b \neq 0$ равно $x = -y_k$ и алгоритм завершает работу.

3. Выбрать случайный вектор $u_{k+1} \in K^n, u_{k+1} \neq 0$.

4. Вычислить первые $2(n - d_k)$ членов последовательности $\{(u_{k+1}, A^i b_k)\} i = 0,1,2, \dots$

5. С помощью алгоритма Берлекэмпа-Месси вычислить минимальный многочлен $f_{k+1}(z)$ последовательности шага 4, нормализованный так, чтобы его свободный член равнялся единице.

6. Присвоить

$$y_{k+1} = y_k + f_{k+1}(A) b_k,$$

$$b_{k+1} = b_0 + A y_{k+1},$$

$$d_{k+1} = d_k + \deg f_{k+1}(z).$$

7. Присвоить $k = k + 1$ и вернуться на 2 шаг.

Коды программ, реализующей рассмотренные алгоритмы

Программа реализована на языке Python (версия интерпретатора 3.6).

```
def change_matrix(system, changable, id_column, id_string):
    for i in range(len(system)):
        if system[i][id_column] != 0 and i in changable:
            system[i], system[id_string] = system[id_string], system[i]
            changable.remove(id_string)
            break


def not_solved(system):
    for i in range(len(system)):
        c = collections.Counter(system[i][:-1])
        if c[0] == len(system[i]) - 1 and system[i][-1] != 0:
            print("Нет решений")
            exit(0)


# System of linear equations
def gauss(n, SOLE, m, changable):
    for i in range(n):
        if i < n:
            if SOLE[i][i] == 0:
                change_matrix(SOLE, changable, i, i)
            else:
                changable.remove(i)
            elem = SOLE[i][i]
            if elem != 1:
                obr_elem = utils.inverse(elem, m)
                for j in range(len(SOLE[i])):
                    SOLE[i][j] *= obr_elem
                    SOLE[i][j] %= m
            for k in range(i + 1, n):
                mult = SOLE[k][i]
                mult = -mult
                for a in range(len(SOLE[i])):
                    SOLE[k][a] += mult * SOLE[i][a]
                    SOLE[k][a] %= m
            if i != 0:
                for k in range(0, i):
                    mult = SOLE[k][i]
                    mult = -mult
                    for a in range(len(SOLE[i])):
                        SOLE[k][a] += mult * SOLE[i][a]
                        SOLE[k][a] %= m
            for j in range(n):
                c = collections.Counter(SOLE[j])
                if c[0] == len(SOLE[j]):
                    SOLE.remove(SOLE[j])
                    n -= 1
        not_solved(SOLE)
    return SOLE
```

Оценки сложности рассмотренных алгоритмов

Сложность метода Гаусса равна $O(n^3)$.

Результаты тестирования программы

Рассмотрим несколько СЛАУ. Решим каждую с помощью метода Гаусса.

1) $\begin{pmatrix} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{pmatrix}$ в поле $\mathbb{Z}_5$

Результат работы программы:

$$x1 = 2$$
$$x2 = 3$$
$$x3 = 4$$

2) $\begin{pmatrix} 3 & 2 & -5 & 4 & 1 \\ 2 & -1 & 3 & 1 & 13 \\ 1 & 2 & -1 & 8 & 9 \end{pmatrix}$ в поле $\mathbb{Z}_5$

Результат работы программы:

$$x1 + 3 * x3 + 3 * x4 = 1$$
$$x2 + 3 * x3 = 4$$

Результат тестирования программ по времени:

Матрица	Время работы
$\begin{pmatrix} 3 & 2 & -5 & 4 & 1 \\ 2 & -1 & 3 & 1 & 13 \\ 1 & 2 & -1 & 8 & 9 \end{pmatrix}$ в поле $\mathbb{Z}_5$	2.595с.
$\begin{pmatrix} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{pmatrix}$ в поле $\mathbb{Z}_5$	4.5с.
$\begin{pmatrix} 1 & 2 & 3 & 4 & 7 & 8 \\ -3 & 2 & 4 & 5 & 3 & 12 \\ 6 & 8 & 3 & 8 & 9 & 2 \\ 6 & 3 & 9 & 2 & 3 & 11 \end{pmatrix}$ в поле $\mathbb{Z}_5$	8.155с.

Время работы прямо пропорционально зависит от сложности СЛАУ.