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 линейных алгебраических уравнений с неизвестными переменными вида:

где – неизвестные переменные, , – коэффициенты, – свободные члены. Такую форму записи СЛАУ называют координатной.

В матричной форме записи эта система уравнений имеет следующий вид , где – основная матрица системы, – матрица столбец неизвестных переменных, – матрица столбец свободных членов.

Если к матрице добавить в качестве 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*. – количество уравнений системы, – модуль, – расширенная матрица системы линейных уравнений.

*Выход*. Общее решение системы линейных уравнений.

1. Положить .

2. Пока не равно :

2.1. Выбираем -ю строку матрицы и элемент .

2.2. Проверяем, что выбранный элемент , 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 . И выполняем следующие действия.

2.4.1. Выбираем -ый элементы -ой строки.

2.4.2. Далее умножаем -ую строку на выбранный на предыдущем шаге элемент и складываем с -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 – поле, – матрица размера над , – -мерный вектор, . Мы хотим решить систему линейных уравнений .

Предположим дополнительно, что матрица – симметричная, а вектор удовлетворяет условию Рассмотрим последовательность Крылова , состоящую из -мерных векторов , где

*Лемма*. Пусть , – линейно независимы над а – линейно зависимы. Тогда любой элемент из множества линейно выражается через .

*Лемма*. Пусть процесс ортогонализации закончился вектором . Если , то если же то

*Лемма*. Пусть процесс ортогонализации завершился при , при этом были построены линейно независимые векторы . Тогда при дает решение .

Алгоритм Ланцоша работает следующим образом. Мы вычисляем последовательность векторов пользуясь формулами

Если на некотором шаге будет построен вектор такой, что то мы не сможем найти решение этим методом. Если же будет построен вектор то решение мы находим по формуле при . При этом следует сделать проверку, поскольку мы предполагаем выполнение некоторых условий.

В алгоритмах факторизации и дискретного логарифмирования матрица системы не является симметричной. В этом случае предполагается случайным образом выбрать диагональную матрицу и рассмотреть систему уравнений .

Ее матрица будет квадратной и симметричной, и к системе мы затем применяем алгоритм Ланцоша. Если алгоритм закончится неудачей либо найденный вектор не будет решением , то следует выбрать другую матрицу

Однородная система может быть преобразована в неоднородную в предположении, что в ее решении последняя координата . Положив тогда , мы сможем перенести столбец соответствующих коэффициентов в правую часть и затем применить алгоритм Ланцоша.

Алгоритм Видемана

Алгоритм 1.

1. Присвоить .

2. Если то решение равно и алгоритм завершает работу.

3. Выбрать случайный вектор

4. Вычислить первые членов последовательности

5. С помощью алгоритма Берлекэмпа-Месси вычислить минимальный многочлен последовательности шага 4, нормализованный так, чтобы его свободный член равнялся единице.

6. Присвоить

7. Присвоить 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 .

**Результаты тестирования программы**

Рассмотрим несколько СЛАУ. Решим каждую с помощью метода Гаусса.

1)

Результат работы программы:

|  |
| --- |
|  |

2)

Результат работы программы:

|  |
| --- |
|  |

Результат тестирования программ по времени:

|  |  |
| --- | --- |
| Матрица | Время работы |
|  | 2.595с. |
|  | 4.5с. |
|  | 8.155с. |

Время работы прямо пропорционально зависит от сложности СЛАУ.