МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по практической работе №11
по дисциплине «Вычислительная математика»
Тема: Решение системы линейных уравнений

Студент гр. 8383	____________________	Ларин А.
Преподаватель	____________________	Сучков А.И.

Санкт-Петербург
2019

Цель работы.

Исследование и реализация различных методов решения систем линейных алгебраических уравнений.

Основные теоретические положения.

Методы решения систем линейных алгебраических уравнений (СЛАУ) делятся на две группы. К первой группе принадлежат так называемые точные, или прямые, методы – алгоритмы, позволяющие получить решение системы за конечное число арифметических действий. Сюда относятся известное правило Крамера нахождения решения с помощью определителей, метод Гаусса (метод исключений) и метод прогонки. Правило Крамера при реализации на ЭВМ не применяется ввиду значительно большего по сравнению с методом Гаусса числа арифметических действий. Метод Гаусса используется при решении систем до порядка 10^3. Метод прогонки применяется для решения важного класса специальных систем линейных уравнений с трехдиагональной матрицей, часто возникающей в практических приложениях.

Решение систем линейных алгебраических уравнений методом Гаусса

Рассматривается СЛАУ n-го порядка

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases},$$

что в векторном виде записывается как $Ax = b$.

Суть метода исключения по главным элементам (метод Гаусса) заключается в следующем. Находится наибольший по абсолютной величине коэффициент a_{kj}. Для исключения x_j из i-го уравнения $(i \neq k)$ необходимо умножить k-е уравнение на a_{ij}/a_{kj} и вычесть его из i-го уравнения, после чего процесс повторяется для исключения другого неизвестного из оставшихся $n - 1$ уравнений и т.д. В результате система уравнений приводится к треугольному виду

$$\begin{cases} x_1 + \alpha_{12}x_2 + \cdots + \alpha_{1n}x_n = \beta_1, \\ x_2 + \cdots + \alpha_{2n}x_n = \beta_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots, \\ x_n = \beta_n \end{cases}$$

из которого легко находятся неизвестные $x_1, \dots, x_n$. Процесс приведения системы к треугольному виду называется прямым ходом, а нахождение неизвестных $x_1, \dots, x_n$– обратным ходом метода Гаусса.

Следует отметить, что если матрица заданной системы вырожденная, то перед исключением некоторой неизвестной главный элемент a_{kj} окажется равным нулю, что и будет свидетельствовать о равенстве нулю определителя системы. Мерой обусловленности матрицы A называют величину $\nu(A) = \parallel A \parallel \parallel A - 1 \parallel$, где $\parallel A \parallel$ – норма матрицы A. Мера обусловленности равна максимально возможному коэффициенту усиления относительной погрешности от правой части к решению СЛАУ. Если матрица A симметричная и выбрана вторая норма, то мера обусловленности может быть найдена как $\nu(A) = \frac{\max\limits_{1\leqslant i\leqslant n}\{\lambda i(A)\}}{\min\limits_{1\leqslant i\leqslant n}\{\lambda i(A)\}}$ где $\lambda_i(A)$– i-е собственное число матрицы A. Если $\nu(A)$ большая, то матрица A называется плохо обусловленной, в противном случае – хорошо обусловленной.

Решение систем линейных алгебраических уравнений методом простой итерации. Рассматривается система уравнений вида $x = Ax + b$, где A – заданная числовая квадратная матрица n-го порядка, а b – заданный вектор (свободный член). Метод простой итерации состоит в следующем. Выбирается произвольный вектор x (начальное приближение), и строится итерационная последовательность векторов по формуле $x(k) = Ax(k-1) + b$, где$k = 1,2, \dots$

Доказана теорема, что если норма $\parallel A \parallel < 1$, то система уравнений имеет единственное решение x^* и итерации сходятся к решению со скоростью геометрической прогрессии. Для оценки погрешности k-го приближения широко применяется неравенство $\parallel x^* - x^{(k)} \parallel \leqslant \frac{\|A\|}{1-\|A\|} \parallel x^{(k)} - x^{(k-1)} \parallel$, которое может быть использовано для принятия решения об останове итерационного процесса при выполнении условия $\frac{\|A\|}{1-\|A\|}\parallel x^{(k)}-x^{(k-1)}\parallel\leqslant\varepsilon$, где ε – некоторая заданная погрешность вычислений.

Постановка задачи.

В ходе выполнения работы студенты должны найти решение системы линейных уравнений с n неизвестными, заданной матрицей коэффициентов A и вектором свободных членов b, методом Гаусса и/или методом простых итераций. Порядок выполнения работы следующий:

1. С помощью преподавателя определить систему уравнений, которую нужно решить.
2. Для решения системы уравнений разработать программу на любом языке программирования, выполняющую решение методом Гаусса и/или методом простых итераций.
3. Для метода Гаусса: провести вычисления с использованием разработанной программы и исследовать обусловленность задачи с использованием пакета MATLAB, при этом для определения числа обусловленности матрицы A рекомендуется использовать функцию `cond(A)`. Кроме того, для проверки получаемых результатов можно провести вычисления с помощью пакета MATLAB.
4. Для метода простых итераций: произвести вычисления с использованием разработанной программы и построить график зависимости числа итераций от задаваемой точности.

Выполнение работы.

Возьмем в качестве примера систему

$$\begin{pmatrix} 0.8143 & 0.1966 & 0.3517 & 0.9172 & 0.3804 \\ 0.2435 & 0.2511 & 0.8308 & 0.2858 & 0.5678 \\ 0.9293 & 0.6160 & 0.5853 & 0.7572 & 0.0759 \\ 0.3500 & 0.4700 & 0.5497 & 0.7537 & 0.0540 \end{pmatrix}$$

Программа разработана на языке Python, с live скриптом matlab для удобства и проверки.

Для решения СЛАУ методом Гаусса разработана ф-я gauss, принимающая на вход матрицу A вместе со столбцом свободных членов.

Была разработана программа на языке m, обеспечивающая ввод-вывод данных, служащая оберткой над кодом на языке Python, и помогающая контролировать правильность работы основной программы путем решения идентичной СЛАУ методами среды matlab. Данная программа состоит из нескольких модулей. Среди них:

- Ввод данных из файла
- Ручной ввод данных
- Случайная генерация данных
- Решение СЛАУ методами matlab
- Решение СЛАУ при помощи программы на Python и сравнение результатов.

Также методами matlab осуществляется расчет числа обусловленности матрицы A.

Получаем вектор $x = \begin{pmatrix} 0.3064 \\ -1.3346 \\ 0.9785 \\ 0.0537 \end{pmatrix}$. Число обусловленности $\nu = 8.5669$

Код программы на Python представлен а приложении А, на m в приложении Б

Было проведено исследование обусловленности метода Гаусса. В таб. 1. приведены погрешность входных значений Delta и порядок погрешности результата Eps. Более подробные результаты можно получить при помощи программы приведенной в приложении Б.

Таблица 1 – Зависимость порядка точности результата от точности входных данных.

Количество знаков	Значение Delta	Порядок Eps
8	0.00000001	10^{-15}
6	0.000001	10^{-6}

Окончание таблицы 1

5	0.00001	10^{-5}
4	0.0001	10^{-4}
3	0.001	10^{-3}
2	0.01	10^{-3}
1	0.1	10^{-1}

Выводы.

Проанализировав результаты работы можно сделать выводы, что метод Гаусса имеет в значительной степени более простую реализацию, что метод простой итерации. А также при $det(A) \neq 0$ метод Гаусса гарантированно сходится, т.е. дает вектор x. В результате можно сделать вывод, что метод Гаусса является самым простым и надежным методом решения СЛАУ, однако имеет довольно низкую скорость работы. По результатам оценки обусловленности метода Гаусса можно сделать вывод, что значение обусловленности примерно равно 1.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ PYTHON

```
import math
import numpy
import sys

def gauss(A):

    n=len(A)
    lc = False#lin comb
    for brow in range(n-1):
        A[brow::1]  =  sorted(A[brow::1],  key=lambda  x:  -
abs(x[brow]))
        for crow in range(brow+1,n):
            if A[brow][brow] ==0:
                print("lc")
                lc=True
                continue
            mul = A[crow][brow] / A[brow][brow]
            for ccol in range(n+1):
                A[crow][ccol]-=A[brow][ccol] * mul

    ic=False           #insolvable
    for brow in range(n-1,-1,-1):
        #A[brow::1]  =  sorted(A[brow::1],  key=lambda  x:  -
abs(x[brow]))
        for crow in range(brow-1,-1,-1):
            if A[brow][brow] ==0:
                print("lc")
                ic=True
                continue
            mul = A[crow][brow] / A[brow][brow]
            for ccol in range(n+1):
                A[crow][ccol]-=A[brow][ccol] * mul

    if ic: return False

    x=[]
    for i in range(n):
        x.append(A[i][-1]/A[i][i])
    return x

def iter(A,i):
    its=i
    b = [i[-1] for i in A]
```

```
    A = [i[:-1] for i in A]
    flag = True
    n=len(A)

    C = [[0 for i in range(n)]for j in range(n)]
    d = [0 for i in range(n)]

    for i in range(n):
        for j in range(n):
            if i==j:
                C[i][j] = 0
            else:
                C[i][j] = -A[i][j] / A[i][i]
                d[i] = b[i] / A[i][i]

    #print(C)
    #print(d)


    A = numpy.array(A)
    b = numpy.array(b)
    C = numpy.array(C)
    d = numpy.array(d)

    #print("N", numpy.linalg.norm(C))
    #print("N", numpy.linalg.norm(A))
    if numpy.linalg.norm(C) > 1:
        flag = False
        print("Iter not applyable")


    x0 = numpy.array([0 for i in range(n)])
    x = x0
    for i in range(its):
        x = C.dot(x)+d




    #print("Foo",x)
    return x


if __name__ == '__main__':#(i/g pres [its])
    A=[]
    #print(sys.argv[0])
    n = int(input())
```

```
        [A.append([round(float(j),int(sys.argv[2]))   for   j   in input().strip(' ').split(' ')]) for i in range(n)]
        #[A.append([float(j)  for  j  in  input().strip('  ').split(' ')]) for i in range(n)]
        B=[i[:-1] for i in A]
        #iter(A)
        x = gauss(A)
        #while(len(sys.argv)<3):
        #    sys.argv.append('10')

        if(0 or sys.argv[1]=='g'):
            [print(i) for i in x]
        if(sys.argv[1]=='i'):
            x = iter(A,1)
            [print(i) for i in x]




        # print([i for i in [str(j) for j in A]])
```

ПРИЛОЖЕНИЕ Б
КОД ПРОГРАММЫ M

```
%PREPARATIONS
cd
/media/anton/E6D8B24FD8B21E2D/Git/txcloud/Labs/CM/Larin_Anton_8383_CM_21_11/Solution
%cd D:\Git\TxCloud\Labs\CM\Larin_Anton_8383_CM_21_11\Solution

filename = "inp"

%MATRIX BY HANDS

%2
%0.5308 0.9304 0.5688
%0.7792 0.1299 0.4694

n=2;
M=[1 2 3;
```

```
4 5 6];

file = fopen(filename,'w');
fprintf(file,'%d\n',n);
for j = 1:size(M,1)
    for i = 1:size(M,2)
        fprintf(file,"%d ",M(j,i));
    end
    fprintf(file,"\n");
end
fclose(file);
A = M(:,1:1:end-1)
b=M(:,end)

%RANDOM
n=randi([2,10],1,1)
M=rand(n,n+1)



file = fopen(filename,'w');
fprintf(file,'%d\n',n);
for j = 1:size(M,1)
    for i = 1:size(M,2)
        fprintf(file,"%d ",M(j,i));
    end
    fprintf(file,"\n");
end
fclose(file);
A = M(:,1:1:end-1)
b=M(:,end)

%MATRIX FROM FILE
file = fopen(filename,'r');
raw=fscanf(file,"%f");
n=raw(1)
```

```
raw=raw(2:end);
M=vec2mat(raw,n+1);
A = M(1:1:n,1:1:end-1)
b=M(1:1:n,end)


%PROCESS



ethalonRoots = A\b
%gauss
[~,out]=system("python3 main.py g 16 <"+filename)
%out="42"
pyRoots = str2num(out);
vpa(pyRoots,16)
%iter roots
[~,out]=system("python3 main.py i 16 10 <"+filename)
%out="42"
pyRoots = str2num(out);

%cond res

ethalonRoots = A\b

pr=8

res=[]
eps=[]

while pr>0
    [~,out]=system("python3 main.py g "+pr+" <"+filename);
    pyRoots = str2num(out);
    res=[res,pyRoots];
    eps=[eps,ethalonRoots-pyRoots]
    pr=pr-1;
end
```

```
res
for i=1:size(eps,2)
    vpa(eps(:,i),10)
end
vpa(eps,16)



cond(A)
eig(A)
ethalonRoots
pyRoots
```