МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ПЕТРОВ АНДРЕЙ АЛЕКСАНДРОВИЧ

Отчет по лабораторной работе № 1
вариант 4
(«Методы вычислений»)
студента 2-го курса 14-ой группы

Выполнил:
студент 2 курса 14 группы
Петров Андрей Александрович

Преподавать:
Бондарь И.В.

Минск
2021

УСЛОВИЕ

1. Используя библиотеку NumPy, написать программу, которая решает СЛАУ $Ax = b$ и вычисляет определитель матрицы A методом отражений. Применить программу к следующим ниже входным данным и вывести результат:

$$A = \begin{pmatrix} -1 & -4 & -5 & -1 & 0 & -1 & -1 \\ -1 & -4 & 0 & 4 & -4 & 0 & -4 \\ -2 & -8 & -5 & -1 & 0 & -4 & 0 \\ -4 & -16 & -10 & 2 & -4 & -2 & 0 \\ -8 & -32 & -20 & 4 & -8 & 0 & -2 \\ -16 & -64 & -40 & 8 & -16 & -7 & -4 \\ 5 & -5 & -5 & -2 & 0 & -2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} -12 \\ -8 \\ -18 \\ -30 \\ -58 \\ -123 \\ -15 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 & 512 \\ 1 & 3 & 9 & 27 & 81 & 243 & 729 & 2187 & 6561 & 19683 \\ 1 & 4 & 16 & 64 & 256 & 1024 & 4096 & 16384 & 65536 & 262144 \\ 1 & 5 & 25 & 125 & 625 & 3125 & 15625 & 78125 & 390625 & 1953125 \\ 1 & 6 & 36 & 216 & 1296 & 7776 & 46656 & 279936 & 1679616 & 10077696 \\ 1 & 7 & 49 & 343 & 2401 & 16807 & 117649 & 823543 & 5764801 & 40353607 \\ 1 & 8 & 64 & 512 & 4096 & 32768 & 262144 & 2097152 & 16777216 & 134217728 \\ 1 & 9 & 81 & 729 & 6561 & 59049 & 531441 & 4782969 & 43046721 & 387420489 \\ 1 & 10 & 100 & 1000 & 10000 & 100000 & 1000000 & 10000000 & 100000000 & 1000000000 \end{pmatrix} \quad b = \begin{pmatrix} 10 \\ 1023 \\ 29524 \\ 349525 \\ 2441406 \\ 12093235 \\ 47079208 \\ 153391689 \\ 435848050 \\ 1111111111 \end{pmatrix}$$

$$A = \begin{pmatrix} 5 & 3 & 3 & -4 & 5 & -5 & -4 & -5 & 0 & 2 \\ 5 & 3 & -1 & 2 & 3 & 0 & -4 & 1 & -4 & -5 \\ 10 & 6 & 2 & -3 & -2 & -2 & -1 & 0 & -3 & -5 \\ 20 & 12 & 4 & -5 & -1 & -4 & 4 & -2 & -2 & -4 \\ 40 & 24 & 8 & -10 & 5 & -1 & 5 & 2 & 0 & -3 \\ 80 & 48 & 16 & -20 & 10 & -12 & -3 & -3 & 3 & 2 \\ 160 & 96 & 32 & -40 & 20 & -24 & -3 & -4 & 1 & 4 \\ 320 & 192 & 64 & -80 & 40 & -48 & -6 & -11 & 0 & -3 \\ 640 & 384 & 128 & -160 & 80 & -96 & -12 & -22 & -5 & 2 \\ -3 & -3 & 0 & 0 & 5 & 3 & -2 & 2 & 5 & -2 \end{pmatrix} \quad b = \begin{pmatrix} -18 \\ 2 \\ -8 \\ -6 \\ 24 \\ 30 \\ 64 \\ 100 \\ 216 \\ 0 \end{pmatrix}$$

2. Найти точное решение указанных систем с использованием библиотеки SymPy или аналогичного программного обеспечения и сравнить результаты.

3. Вычислить число обусловленности в максимум-норме матрицы A из второго тестового задания. Что это означает на практике? Путем решения нескольких СЛАУ с возмущенным вектором b подтвердите связь между числом обусловленности и относительными погрешностями начальных данных и решения.

4. Проведите экспериментальное исследование скорости решения СЛАУ в зависимости от размерности системы, используя для тестов матрицу A и вектор b со случайными числами. Постройте график зависимости времени работы от размерности. Систему какой размерности ваша программа на вашем компьютере может решить за одну минуту?

ХОД ВЫПОЛНЕНИЯ

ЗАДАНИЕ 1.

1. Программа написана на языке C#. Для задания матриц и векторов использовалась библиотека MathNet.Numerics.

Реализация программы:

Считываем данные из файла и инициализируем Matrix A и Vector b

```
var A :Matrix<double> = Matrix<double>.Build.DenseOfArray(FileReader.ReadMatrix( path: "../../../resources/matrix1.txt"));
var b :Vector<double> = Vector<double>.Build.DenseOfArray(FileReader.ReadVector( path: "../../../resources/vector1.txt"));
```

Для решения СЛАУ Ax=b методом отражений реализуем метод Reflection():

Присоединяем к матрице A вектор b.

```
var m :Matrix<double> = Matrix<double>.Build.Dense( rows: a.RowCount, columns: a.ColumnCount + 1);
var x :Vector<double> = Vector<double>.Build.Dense(a.RowCount);
for (var i = 0; i < m.RowCount; i++)
{
    for (var j = 0; j < m.ColumnCount - 1; j++)
        m[i, j] = a[i, j];
    m[i, m.ColumnCount - 1] = b[i];
}
```

Приводим матрицу A к треугольной и находим матрицу R методом отражений.

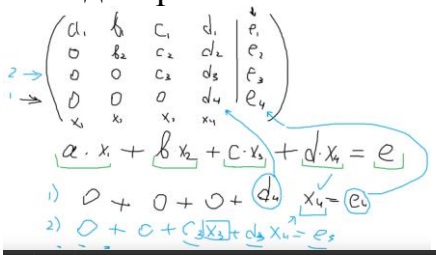
```
// w - верхний элемент вектора "омега", root - верхний элемент a~
double w, root;
for (var k = 0; k < m.ColumnCount - 2; k++)
{
    root = m.FindLength( row: k,  col: k);
    w = m[k, k] - root;
    m[k, k] = root;
    // a = a - 2*|aw|*w, где w - вектор "Омега", |aw| - скалярное произведение (scalarMultiply)
    // также нет необходимости непосредственно делить вектор "омега" на его длину,
    // в ходе вычислений длина выносится за скобки и возводится в квадрат (lengthIn2),
    // это позволяет увеличить точность, так как нам не надо считать квадратные корни и делить вектор на них
    var lengthIn2 :double = m.FindLengthIn2( row: k + 1,  col: k) + w * w;
```

```
    for (var j:int = k + 1; j < m.ColumnCount; j++)
    {
        var scalarMultiply:double = m[k, j] * w + m.ScalarMultiplyColumns( rowStart: k + 1, rowEnd: m.RowCount, col1: j, col2: k);
        m[k, j] = m[k, j] - 2 * scalarMultiply / lengthIn2 * w;
        for (var i:int = k + 1; i < m.RowCount; i++)
            m[i, j] = m[i, j] - 2 * scalarMultiply / lengthIn2 * m[i, k];
    }
    for (var i:int = k + 1; i < m.RowCount; i++)
        m[i, k] = 0; // зануление нижних элементов a~
}
```

Выделяем вектор b из матрицы A в вектор x.

```
// выделяем вектор b из матрицы в вектор x
for (var i = 0; i < b.Count; i++)
    x[i] = m[i, m.ColumnCount - 1];
```

Находим решение линейного уравнения:



```
        for (var i = 1; i <= m.RowCount; i++)
        {
            double temp = 0;
            temp += x[^i];
            for (var j = 1; j < i; j++)
                temp += m[m.RowCount - i, m.ColumnCount - j - 1] * -x[m.RowCount - j];

            x[^i] = temp / m[m.RowCount - i, m.ColumnCount - i - 1];
        }
        return x;
```

Нахождение определителя матрицы методом отражений:

```
Console.WriteLine("log>\tDeterminant: {0}", A.QR().Determinant);
```

Выполнение программы:

1) Входные данные:

$$A = \begin{pmatrix} -1 & -4 & -5 & -1 & 0 & -1 & -1 \\ -1 & -4 & 0 & 4 & -4 & 0 & -4 \\ -2 & -8 & -5 & -1 & 0 & -4 & 0 \\ -4 & -16 & -10 & 2 & -4 & -2 & 0 \\ -8 & -32 & -20 & 4 & -8 & 0 & -2 \\ -16 & -64 & -40 & 8 & -16 & -7 & -4 \\ 5 & -5 & -5 & -2 & 0 & -2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} -12 \\ -8 \\ -18 \\ -30 \\ -58 \\ -123 \\ -15 \end{pmatrix}$$

Результат выполнения:

```
log>    Resultat:
        DenseVector 7-Double
-0,64
 1,16
  1,8
   -3
   -3
    1
    1

log>    Determinant: 0
log>    Time: 2 Milliseconds
```

2) Входные данные:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 & 512 \\ 1 & 3 & 9 & 27 & 81 & 243 & 729 & 2187 & 6561 & 19683 \\ 1 & 4 & 16 & 64 & 256 & 1024 & 4096 & 16384 & 65536 & 262144 \\ 1 & 5 & 25 & 125 & 625 & 3125 & 15625 & 78125 & 390625 & 1953125 \\ 1 & 6 & 36 & 216 & 1296 & 7776 & 46656 & 279936 & 1679616 & 10077696 \\ 1 & 7 & 49 & 343 & 2401 & 16807 & 117649 & 823543 & 5764801 & 40353607 \\ 1 & 8 & 64 & 512 & 4096 & 32768 & 262144 & 2097152 & 16777216 & 134217728 \\ 1 & 9 & 81 & 729 & 6561 & 59049 & 531441 & 4782969 & 43046721 & 387420489 \\ 1 & 10 & 100 & 1000 & 10000 & 100000 & 1000000 & 10000000 & 100000000 & 1000000000 \end{pmatrix} \quad b = \begin{pmatrix} 10 \\ 1023 \\ 29524 \\ 349525 \\ 2441406 \\ 12093235 \\ 47079208 \\ 153391689 \\ 435848050 \\ 1111111111 \end{pmatrix}$$

Результат выполнения:

```
log>    Resultat:
        DenseVector 10-Double
 1,00002
0,999958
 1,00004
0,999976
 1,00001
0,999998
       1
       1
       1
       1

log>    Determinant: 1,834933472955732E+21
log>    Time: 2 Milliseconds
```

3) Входные данные:

$$A = \begin{pmatrix} -1 & -4 & -5 & -1 & 0 & -1 & -1 \\ -1 & -4 & 0 & 4 & -4 & 0 & -4 \\ -2 & -8 & -5 & -1 & 0 & -4 & 0 \\ -4 & -16 & -10 & 2 & -4 & -2 & 0 \\ -8 & -32 & -20 & 4 & -8 & 0 & -2 \\ -16 & -64 & -40 & 8 & -16 & -7 & -4 \\ 5 & -5 & -5 & -2 & 0 & -2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} -12 \\ -8 \\ -18 \\ -30 \\ -58 \\ -123 \\ -15 \end{pmatrix}$$

Результат выполнения:

```
log>    Resultat:
        DenseVector 10-Double
-3,90737E-14
           2
 5,73398E-13
           2
-3,08811E-14
           2
-2,32819E-14
           2
-6,39488E-15
           2

log>    Determinant: 1058399,9999995849
```

ЗАДАНИЕ 2.

Найдем точные решения СЛАУ с помощью библиотеки NumPy и MathNet.Numerics.

1) Невозможно найти единственное решение СЛАУ как с помощью библиотеки NumPy и MathNet.Numerics, так как определитель матрицы равен 0.

```
log>    Result:
        DenseVector 7-Double
не число
не число
не число
не число
не число
       1
       1

log>    Time: 6 Milliseconds
```

2) Решение с помощью библиотеки NumPy:

```
%%time
np.linalg.solve(M, b)

Wall time: 0 ns
array([1.0000089 , 0.99997703, 1.00002328, 0.9999875 , 1.000004  ,
       0.9999992 , 1.0000001 , 0.99999999, 1.        , 1.        ])
```

Решение с помощью библиотеки MathNet.Numerics:

```
log>    Result:
        DenseVector 10-Double
      1
0,999997
      1
0,999998
      1
      1
      1
      1
      1
      1

log>    Time: 5 Milliseconds
```

3) Решение с помощью библиотеки NumPy:

```
%%time
np.linalg.solve(M, b)

Wall time: 0 ns
array([ 0.,  2.,  0.,  2., -0.,  2., -0.,  2.,  0.,  2.])
```

Решение с помощью библиотеки MathNet.Numerics:

```
log>    Result:
        DenseVector 10-Double
3,55271E-16
         2
         0
         2
        -0
         2
        -0
         2
         0
         2

log>    Time: 7 Milliseconds
```

Вывод:

Сравнив результаты выполнения моей программы и результатов полученных с помощью библиотек NumPy и MathNet.Numerics, можно сделать вывод, что результаты идентичны с учетом погрешности вычислений.

ЗАДАНИЕ 3.

На практике это означает, что нахождение числа обусловленности в максимум-норме матрицы A, находится путем умножения максимума нормы матрицы A на максимум нормы матрицы обратной матрице A.

Возмутим вектор b путем минимального изменения его значений (vector_b +epsilon*vector_Sdviga(rand(1, -1))

```
log>    Resultat:
        DenseVector 10-Double
-0,0229701
   3,76188
  -1,97571
   2,71613
  0,407482
   1,12838
    0,9824
   1,00148
   0,99993
         1

log>    Determinant: 1,834933472955732E+21
log>    Time: 2 Milliseconds
```

Результат отличен от результата до возмущения вектора b.

Вывод:

При изменении вектора b решение меняется. Чем больше будет изменено значение вектора, тем больше будет изменено решение. Следовательно, можно сделать вывод, что чем больше мы будем совершать погрешности, тем больше будет искажаться результат в ходе решения.

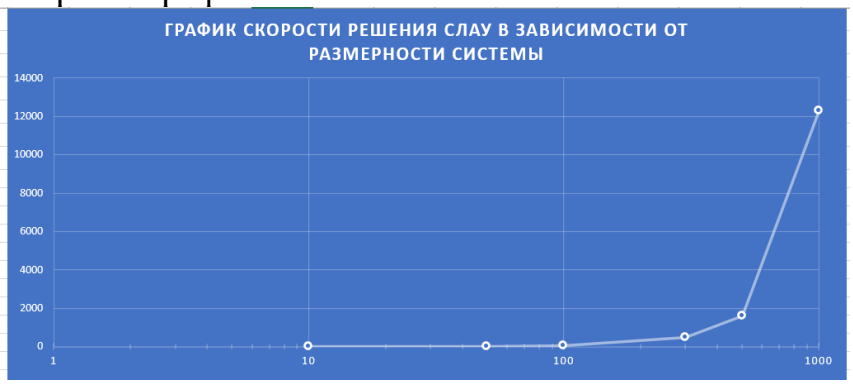
ЗАДАНИЕ 4.

Зададим рандомные значения матрице и вектору нужной нам длины

```
var A :Matrix<double> = Matrix<double>.Build.Random( rows: 50,  columns: 50);
var b :Vector<double> = Vector<double>.Build.Random( length: 50);
```

Проведем несколько выполнений программы с разной размерностью матрицы и вектора. Построим график зависимости.

N	t, ms
10	3
50	12
100	40
300	463
500	1618
1000	12314



Мой компьютер смог посчитать СЛАУ для матрицы максимальной размерности в 2000x2000 и соответствующего вектора за 72171 милисекунд.