Презентация по лабораторной работе №4

Дисциплина “Научное программирование”

Живцова А.А.

11 октября 2024

Кафедра теории вероятностей и кибербезопасности, Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

Информация

Докладчик

- Живцова Анна Александровна
- студент кафедры теории вероятностей и кибербезопасности
- Российский университет дружбы народов имени Патриса Лумумбы
- zhivtsova_aa@pfur.ru
- https://github.com/AnnaZhiv

Вводная часть

Актуальность

Системы линейных уравнений возникают во многих научных задачах. Языки Octave и Julia содержат сложные алгоритмы, встроенные для решения систем линейных уравнений. Наиболее популярные методы: метод Гаусса, обращение матрицы и LU разложение мы рассмотрим сегодня.

Объект и предмет исследования

- Системы линейных алгебраических уравнений
- Метод Гаусса
- Метод обращения матрицы
- LU разложение

Цели

- Изучить методы решения систем линейных уравнений
- Реализовать методы программно на языках Octave и Julia

Задачи

- Изучить и реализовать метод Гаусса для решения систем линейных уравнений
- Решить систему линейных уравнений обращением матрицы
- Изучить и реализовать метод решения систем линейных уравнений, основанный на LU разложении

Материалы и методы

- Язык научного программирования Octave
- Среда программирования GNUoctave
- Язык научного программирования Julia
- Среда программирования Jupyter notebook

Выполнение работы

Исходные данные

Будем решать систему линейных уравнений, в матричном виде имеющую запись $Ax = b$. Зададим матрицу A и вектор b.

```
>> A = [ 1 2 3; 0 -2 -4; 1 -1 0]
A =

   1   2   3
   0  -2  -4
   1  -1   0

>> b = [4; 6; 0]
b =

   4
   6
   0
```

Рис. 1: Матрица A и вектор b

Решение методом Гаусса. Octave

Проведем прямой и обратный ход вручную и сравним результаты с встроенной функцией `rref`.

```
>> B = [ 1 2 3 4 ; 0 -2 -4 6 ; 1 -1 0 0 ];
>> B(3,:) = (-1) * B(1,:) + B(3,:);
>> B(3,:) = -1.5 * B(2,:) + B(3,:);
>> B
B =

    1    2    3    4
    0   -2   -4    6
    0    0    3  -13

>> x1 = [4 + 13 + 6 - 4*13/3;(-6 + 4*13/3)/2;-13/3]
x1 =

   5.6667
   5.6667
  -4.3333

>> rref(B)
ans =
```

Решение методом Гаусса. Julia

```
B = [ 1 2 3 4 ; 0 -2 -4 6 ; 1 -1 0 0 ];
A = B[:,1:3];
b = B[:,4];
B[3,:] = (-1) * B[1,:] + B[3,:];
B[3,:] = -1.5 * B[2,:] + B[3,:];
B
```

```
3×4 Matrix{Int64}:
 1   2   3    4
 0  -2  -4    6
 0   0   3  -13
```

```
x = [4 + 13 + 6 - 4*13/3; (-6 + 4*13/3)/2; -13/3]
```

```
3-element Vector{Float64}:
  5.666666666666668
  5.666666666666666
 -4.333333333333333
```

Найдем вектор x с помощью операции левого деления A\b.

```
>> A\b
ans =

   5.6667
   5.6667
  -4.3333
```

Рис. 4: Решение обращением матрицы. Octave

```
A\b
```

```
3-element Vector{Float64}:
  5.666666666666666
  5.666666666666667
 -4.333333333333333
```

Рис. 5: Решение обращением матрицы. Julia

Решение с помощью LU разложения. Octave

Найдем матрицы L и U с помощью функции `lu` и решим исходную систему уравнений дважды реализовав подстановку.

```
>> [L U P] = lu(A)
L =

   1.0000         0         0
   1.0000    1.0000         0
        0    0.6667    1.0000

U =

   1    2    3
   0   -3   -3
   0    0   -2

P =

Permutation Matrix

   1    0    0
   0    0    1
   0    1    0

>> b1 = [b(1); b(3); b(2)]
b1 =

   4
   0
   6

>> y = [4; -4; 26/3];
>> x = [17 -34/3; 17/3; -13/3]
```

Решение с помощью LU разложения. Julia

```
using LinearAlgebra
lu(A)
```

```
LU{Float64, Matrix{Float64}, Vector{Int64}}
L factor:
3×3 Matrix{Float64}:
 1.0  0.0       0.0
 1.0  1.0       0.0
 0.0  0.666667  1.0
U factor:
3×3 Matrix{Float64}:
 1.0   2.0   3.0
 0.0  -3.0  -3.0
 0.0   0.0  -2.0
```

```
b = [b[1] ; b[3]; b[2]]
```

```
3-element Vector{Int64}:
 4
 0
 6
```

```
y = [4; -4; 26/3]
x = [17/3;17/3;-13/3]
```

```
3-element Vector{Float64}:
  5.666666666666667
  5.666666666666667
 -4.333333333333333
```

Результаты

Проверка полученных результатов. Octave

Вектор x, найденный в этой работе тремя способами, подставим в исходное уравнение и удостоверимся в правильности решения.

```
>> A*x
ans =

   4.0000e+00
   6.0000e+00
  -8.8818e-16

>> b
b =

   4
   6
   0
```


Рис. 8: Проверка полученных результатов. Octave

```
A*x, b
```

```
([4.0, 5.999999999999998, 0.0], [4, 0, 6])
```

Рис. 9: Проверка полученных результатов. Octave

Выводы

Выводы

В данной работе я познакомилась с методами решения систем линейных уравнений и реализовала их программно на языках Octave и Julia. Конкретно я использовала метод Гаусса для решения систем линейных уравнений, метод обращения матрицы и метод, основанный на LU разложении.