Лабораторная работа №5

Научное программирование

Николаев Дмитрий Иванович, НПМмд-02-24

Содержание

1 Цель работы **5**

2 Теоретическое введение **6**

3 Выполнение лабораторной работы **7**

3.1 Подгонка полиномиальной кривой 7

3.1.1 Реализация в Octave 7

3.1.2 Реализация в Julia 14

3.2 Матричные преобразования 16

3.2.1 Реализация в Octave 17

3.2.2 Реализация в Julia 24

4 Выводы **27**

Список литературы **28**

Список иллюстраций

3.1 Построение графика по данным в Octave 8
3.2 График, построенный по данным в Octave 9
3.3 Формирование системы уравнений в Octave (1/3) 10
3.4 Формирование системы уравнений в Octave (2/3) 11
3.5 Формирование системы уравнений в Octave (3/3) 12
3.6 График параболы в Octave 13
3.7 Построение графика по точкам параболы в Octave 13
3.8 Графики, построенный по данным и по точкам параболы в Octave 14
3.9 Код подгонки полиномиальной кривой в Julia 15
3.10 Результат подгонки полиномиальной кривой в Julia 15
3.11 График подгонки полиномиальной кривой в Julia 16
3.12 Первоначальная фигура. Вращение в Octave (1/3) 18
3.13 Первоначальная фигура в Octave 18
3.14 Вращение в Octave (2/3) 19
3.15 Вращение в Octave (3/3) 19
3.16 Вращение фигуры на 90° и 225° в Octave 20
3.17 Отражение фигуры относительно прямой $x = y$ в Octave 21
3.18 График с первоначальной и отражённой фигурами в Octave 22
3.19 Двойная дилатация фигуры в Octave 23
3.20 График с первоначальной и увеличенной в два раза фигурами в Octave 23
3.21 Код вращения фигуры на 90° и 225° в Julia 24
3.22 График первоначальной фигуры и повёрнутых на 90° и 225° в Julia 25
3.23 Код отражения и дилатации фигуры в Julia 25
3.24 График с первоначальной и отражённой фигурами в Julia 26
3.25 График с первоначальной и увеличенной в два раза фигурами в Julia 26

Список таблиц

1 Цель работы

Освоение подгонки полиномиальной кривой, матричных преобразований, вращения, отражения и дилатации, и их реализации в Octave и Julia.

2 Теоретическое введение

Подгонка полиномиальной кривой — это один из методов аппроксимации данных, когда необходимо подобрать полином, который наилучшим образом соответствует заданным точкам. Наиболее распространённый способ для подгонки кривой — метод наименьших квадратов, который минимизирует сумму квадратов отклонений между фактическими и прогнозируемыми значениями.

Для решения такой задачи используется система линейных уравнений, где коэффициенты полинома определяются решением матричных уравнений.

Матричные преобразования включают операции вращения, отражения и дилатации. Эти преобразования широко применяются в компьютерной графике и позволяют изменять координаты объектов:

- **Вращение** — поворот объекта вокруг точки (обычно вокруг начала координат) на заданный угол.
- **Отражение** — зеркальное отображение объекта относительно выбранной оси или линии.
- **Дилатация** — увеличение или уменьшение объекта путём масштабирования.

3 Выполнение лабораторной работы

Следуем указаниям [1]

3.1 Подгонка полиномиальной кривой

Метод подгонки полиномиальной кривой заключается в нахождении полинома $y = ax^2 + bx + c$, который наилучшим образом описывает набор точек. Для этого необходимо решить систему линейных уравнений, в которой коэффициенты a, b и c получаются через матричные операции.

3.1.1 Реализация в Octave

1. Вводим данные и строим график ([3.1,3.2]):

```
D = [1 1; 2 2; 3 5; 4 4; 5 2; 6 -3];
xdata = D(:,1);
ydata = D(:,2);
plot(xdata, ydata, 'o-');
```

```
>> diary on
>> diary
>> D = [ 1 1 ; 2 2 ; 3 5 ; 4 4 ; 5 2 ; 6 -3]
D =

   1   1
   2   2
   3   5
   4   4
   5   2
   6  -3

>> xdata = D(:,1)
xdata =

   1
   2
   3
   4
   5
   6

>> ydata = D(:,2)
ydata =

   1
   2
   5
   4
   2
  -3

>> plot(xdata,ydata,'o-')
```

Рис. 3.1: Построение графика по данным в Octave

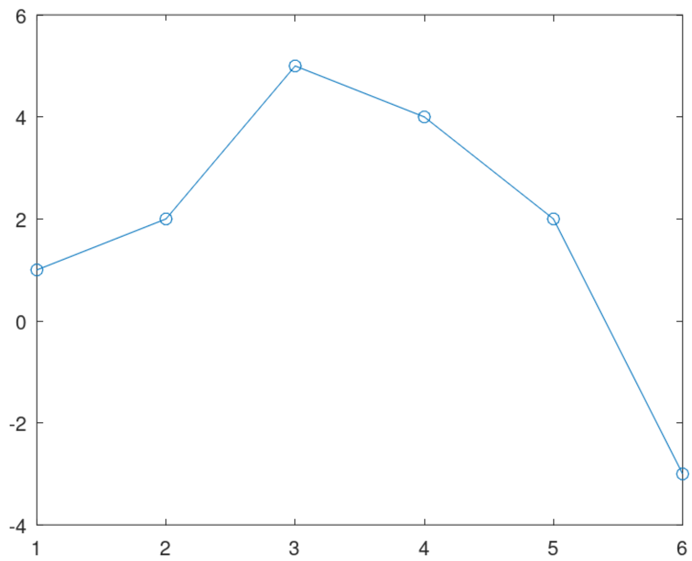

Рис. 3.2: График, построенный по данным в Octave

2. Формируем систему уравнений и решаем методом наименьших квадратов ([3.3-3.5]):

```
A = ones(6, 3);
A(:, 1) = xdata.^2;
A(:, 2) = xdata;
B = A' * A;
B(:, 4) = A' * ydata;
B_res = rref(B);
a1 = B_res(1,4);
a2 = B_res(2,4);
a3 = B_res(3,4);
```

```
>> A = ones(6,3)
A =

   1   1   1
   1   1   1
   1   1   1
   1   1   1
   1   1   1
   1   1   1

>> A(:,1) = xdata .^ 2
A =

    1    1    1
    4    1    1
    9    1    1
   16    1    1
   25    1    1
   36    1    1

>> A(:,2) = xdata
A =

    1    1    1
    4    2    1
    9    3    1
   16    4    1
   25    5    1
   36    6    1
```

Рис. 3.3: Формирование системы уравнений в Octave (1/3)

```
>> A'*A
ans =

   2275    441     91
    441     91     21
     91     21      6

>> A' * ydata
ans =

   60
   28
   11

>> B = A' * A;
>> B (:,4) = A' * ydata
B =

   2275    441     91     60
    441     91     21     28
     91     21      6     11

>> B_res = rref (B)
B_res =

   1.0000        0        0  -0.8929
        0   1.0000        0   5.6500
        0        0   1.0000  -4.4000
```

Рис. 3.4: Формирование системы уравнений в Octave (2/3)

```
>> a1=B_res(1,4)
a1 = -0.8929
>> a2=B_res(2,4)
a2 = 5.6500
>> a3=B_res(3,4)
a3 = -4.4000
>> x = linspace (0,7,50);
>> y = a1 * x .^ 2 + a2 * x + a3;
>> plot (xdata,ydata, 'o' ,x,y, 'linewidth', 2)
>> grid on;
>> legend ('data values', 'least-squares parabola')
>> title ('y = -0.89286 x^2 + 5.65 x - 4.4')
>> P = polyfit (xdata, ydata, 2)
P =

  -0.8929   5.6500  -4.4000

>> y = polyval (P,xdata)
y =

   0.3571
   3.3286
   4.5143
   3.9143
   1.5286
  -2.6429
```

Рис. 3.5: Формирование системы уравнений в Octave (3/3)

3. Строим график параболы ([3.5,3.6]):

```
x = linspace(0, 7, 50);
y = a1 * x.^2 + a2 * x + a3;
plot(xdata, ydata, 'o', x, y, 'linewidth', 2);
grid on;
legend('data values', 'least-squares parabola');
title('y = -0.89286 x^2 + 5.65 x - 4.4');
```

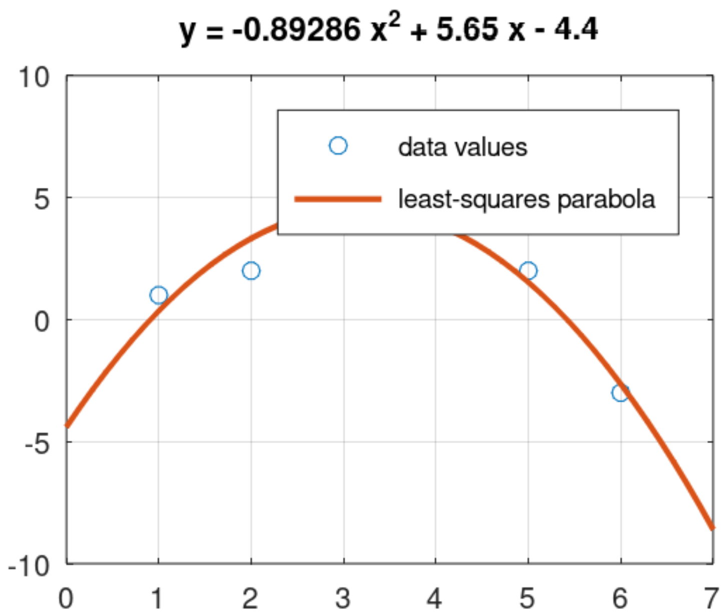

Рис. 3.6: График параболы в Octave

4. Автоматизация с использованием встроенной функции ([3.5,3.7,3.8]):

```
P = polyfit(xdata, ydata, 2);

y = polyval(P, xdata);

plot(xdata, ydata, 'o-', xdata, y, 'r+-');

grid on;

legend('original data', 'polyfit data');
```

```
>> plot(xdata,ydata,'o-',xdata,y,'+-')
>> grid on ;
>> legend ('original data' , 'polyfit data' );
```

Рис. 3.7: Построение графика по точкам параболы в Octave

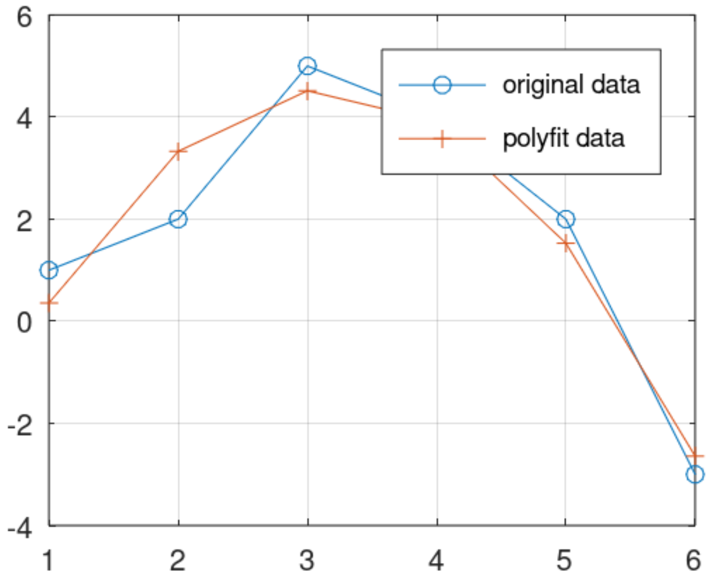

Рис. 3.8: Графики, построенный по данным и по точкам параболы в Octave

3.1.2 Реализация в Julia

Построение данных и их подгонка полиномиальной (второго порядка — параболой) кривой ([3.9,3.10,3.11]):

```julia
using LinearAlgebra, Plots

# Начальные данные
D = [1 1; 2 2; 3 5; 4 4; 5 2; 6 -3]
xdata = D[:,1]
ydata = D[:,2]

# Построение точек на графике
fig = scatter(xdata, ydata, label="Исходные точки", lw=2)

# Построение матрицы A
A = [xdata.^2 xdata ones(length(xdata))]
println("Матрица системы для нахождения коэффициентов аппроксимирующего полинома:\n")
for i in 1:size(A)[1]
    for j in 1:size(A)[2]
        print(A[i, j], " ")
    end
    println("\n")
end

# Решение системы методом наименьших квадратов
b = (A'*A) \ (A'*ydata)

println("Вектор коэффициентов аппроксимирующего полинома: ", b)

# Построение аппроксимирующей параболы
x = range(0, stop=7, length=50)
y = b[1] .* x.^2 + b[2] .* x .+ b[3]
plot!(x, y, label="Аппроксимация параболой")
savefig(fig, "fig8.png")
```

Рис. 3.9: Код подгонки полиномиальной кривой в Julia

```
PS C:\Users\User\Documents\work\study\2024-2025\Научное программирование\sciprog\labs\lab05\report\report> jul
Polinomial_Approximation.jl
Матрица системы для нахождения коэффициентов аппроксимирующего полинома:

1.0 1.0 1.0

4.0 2.0 1.0

9.0 3.0 1.0

16.0 4.0 1.0

25.0 5.0 1.0

36.0 6.0 1.0

Вектор коэффициентов аппроксимирующего полинома: [-0.8928571428571356, 5.649999999999946, -4.399999999999923]
```

Рис. 3.10: Результат подгонки полиномиальной кривой в Julia

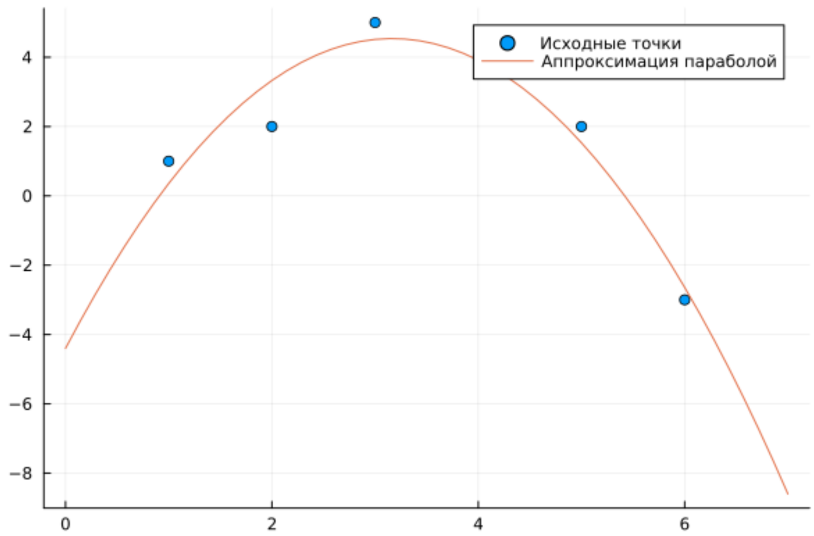

Рис. 3.11: График подгонки полиномиальной кривой в Julia

3.2 Матричные преобразования

Матричные преобразования позволяют изменять координаты объектов. Каждое преобразование можно описать с помощью матрицы:

- **Вращение** осуществляется с помощью матрицы поворота:

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix},$$

где θ — угол поворота.

- **Отражение** относительно оси $x = y$ задаётся матрицей:

$$R = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- **Дилатация** (масштабирование) осуществляется с помощью матрицы:

$$T = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$$

где k — коэффициент масштабирования.

3.2.1 Реализация в Octave

1. Построение графика первоначальной фигуры ([3.12,3.13]), её вращение на 90° и 225° ([3.12-3.15,3.16]):

```
theta1 = 90 * pi / 180;
R1 = [cos(theta1), -sin(theta1); sin(theta1), cos(theta1)];
RD1 = R1 * D';
plot(RD1(1,:), RD1(2,:), 'ro-');
theta2 = 225 * pi / 180;
R2 = [cos(theta2), -sin(theta2); sin(theta2), cos(theta2)];
RD2 = R2 * D';
plot(x, y, 'bo-', RD1(1,:), RD1(2,:), 'ro-', RD2(1,:), RD2(2,:), 'go-');
grid on;
```

```
>> D = [ 1 1 3 3 2 1 3 ; 2 0 0 2 3 2 2 ]
D =

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

>> x = D(1,:)
x =

   1   1   3   3   2   1   3

>> y = D(2,:)
y =

   2   0   0   2   3   2   2

>> plot (x,y)
>> theta1 = 90*pi/180
theta1 = 1.5708
>> R1 = [cos(theta1) -sin(theta1); sin(theta1) cos(theta1)]
R1 =

   6.1230e-17  -1.0000e+00
   1.0000e+00   6.1230e-17

>> RD1 = R1*D
RD1 =

 Columns 1 through 6:

  -2.0000e+00   6.1230e-17   1.8369e-16  -2.0000e+00  -3.0000e+00  -2.0000e+00
   1.0000e+00   1.0000e+00   3.0000e+00   3.0000e+00   2.0000e+00   1.0000e+00

 Column 7:

  -2.0000e+00
   3.0000e+00
```

Рис. 3.12: Первоначальная фигура. Вращение в Octave (1/3)

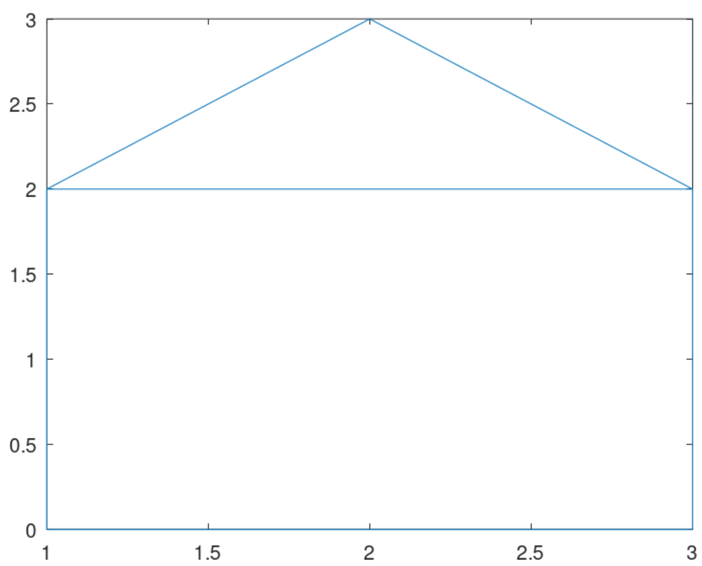

Рис. 3.13: Первоначальная фигура в Octave

```
>> x1 = RD1(1,:)
x1 =

 Columns 1 through 6:

  -2.0000e+00   6.1230e-17   1.8369e-16  -2.0000e+00  -3.0000e+00  -2.0000e+00

 Column 7:

  -2.0000e+00

>> y1 = RD1(2,:)
y1 =

   1   1   3   3   2   1   3

>> theta2 = 225*pi/180
theta2 = 3.9270
>> R2 = [cos(theta2) -sin(theta2); sin(theta2) cos(theta2)]
R2 =

  -0.7071   0.7071
  -0.7071  -0.7071

>> RD2 = R2*D
RD2 =

   0.7071  -0.7071  -2.1213  -0.7071   0.7071   0.7071  -0.7071
  -2.1213  -0.7071  -2.1213  -3.5355  -3.5355  -2.1213  -3.5355

>> x2 = RD2(1,:)
x2 =

   0.7071  -0.7071  -2.1213  -0.7071   0.7071   0.7071  -0.7071
```

Рис. 3.14: Вращение в Octave (2/3)

```
>> y2 = RD2(2,:)
y2 =

  -2.1213  -0.7071  -2.1213  -3.5355  -3.5355  -2.1213  -3.5355

>> plot (x,y, 'bo-' , x1 , y1 , 'ro-' , x2 , y2 , 'go-' )
>> axis ([-4 4 -4 4] , 'equal' ) ;
>> grid on ;
>> legend ('original' , 'rotated 90 deg' , 'rotated 225 deg' ) ;
```

Рис. 3.15: Вращение в Octave (3/3)

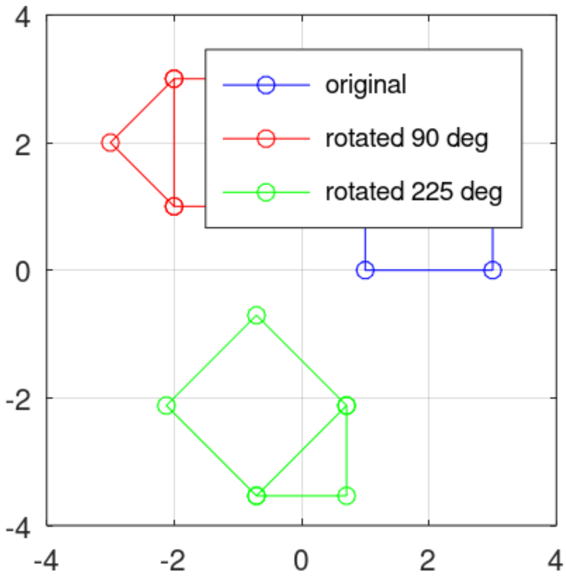

Рис. 3.16: Вращение фигуры на 90° и 225° в Octave

2. Отражение относительно прямой $x = y$ ([3.17,3.18]):

```
R = [0 1; 1 0];
RD = R * D';
plot(x, y, 'o-', RD(1,:), RD(2,:), 'o-');
grid on;
```

```
>> R = [0 1; 1 0]
R =

   0   1
   1   0

>> RD = R * D
RD =

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

>> x1 = RD(1,:)
x1 =

   2   0   0   2   3   2   2

>> y1 = RD(2,:)
y1 =

   1   1   3   3   2   1   3

>> plot (x,y,'o-',x1,y1,'o-')
>> axis([-1 4 -1 4], 'equal');
>> axis([-1 5 -1 5], 'equal');
>> grid on ;
>> legend ( 'original' , 'reflected' );
```

Рис. 3.17: Отражение фигуры относительно прямой $x = y$ в Octave

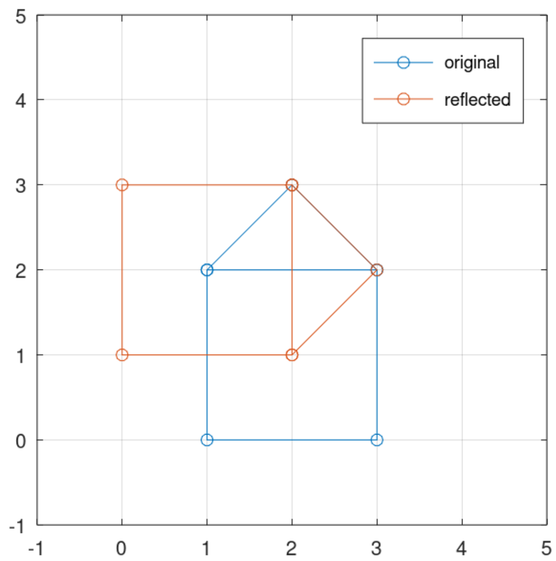

Рис. 3.18: График с первоначальной и отражённой фигурами в Octave

3. Дилатация (увеличение) в 2 раза ([3.19,3.20]):

```
T = [2 0; 0 2];
TD = T * D';
plot(x, y, 'o-', TD(1,:), TD(2,:), 'o-');
grid on;
```

```
>> T = [2 0; 0 2]
T =

   2   0
   0   2

>> TD = T*D;
>> x1 = TD(1,:); y1 = TD(2,:);
>> plot (x, y, 'o-', x1, y1,'o-')
>> axis ([-1 7 -1 7], 'equal');
>> grid on;
>> legend ('original', 'expanded')
>> diary off
```

Рис. 3.19: Двойная дилатация фигуры в Octave

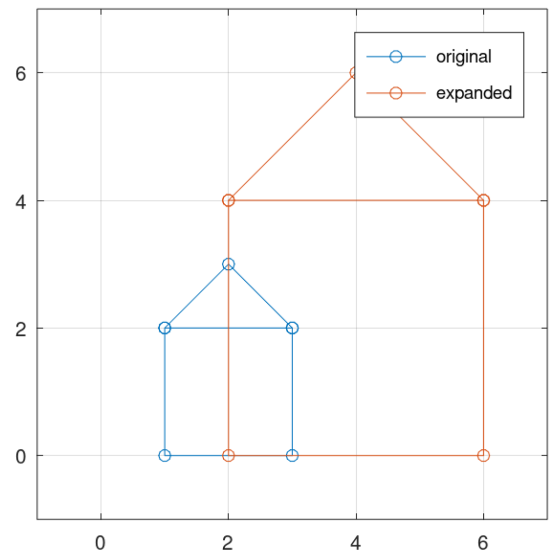

Рис. 3.20: График с первоначальной и увеличенной в два раза фигурами в Octave

3.2.2 Реализация в Julia

Реализация вращения ([3.21,3.22]), отражения и дилатации в два раза ([3.23,3.24,3.25]):

```julia
using LinearAlgebra, Plots

# Начальные данные
D = [ 1 1 3 3 2 1 3 ; 2 0 0 2 3 2 2 ]'
x = D[:,1]
y = D[:,2]
fig1 = plot(x, y, label="Исходный граф")

# Повороты на 90 и 225 градусов
"""Матрица поворота против часовой стрелки на некоторый угол"""
function Rotation_Matrix(theta)
    return [cos(theta) -sin(theta); sin(theta) cos(theta)]
end
theta1 = 90 * pi / 180
R1 = Rotation_Matrix(theta1)
RD1 = R1 * D'

x1 = RD1[1,:]
y1 = RD1[2,:]
plot!(x1, y1, label="Повёрнутый на 90° граф")

theta2 = 225 * pi / 180
R2 = Rotation_Matrix(theta2)
RD2 = R2 * D'

x2 = RD2[1,:]
y2 = RD2[2,:]
plot!(x2, y2, label="Повёрнутый на 225° граф")
savefig(fig1, "fig9.png")
```

Рис. 3.21: Код вращения фигуры на 90° и 225° в Julia

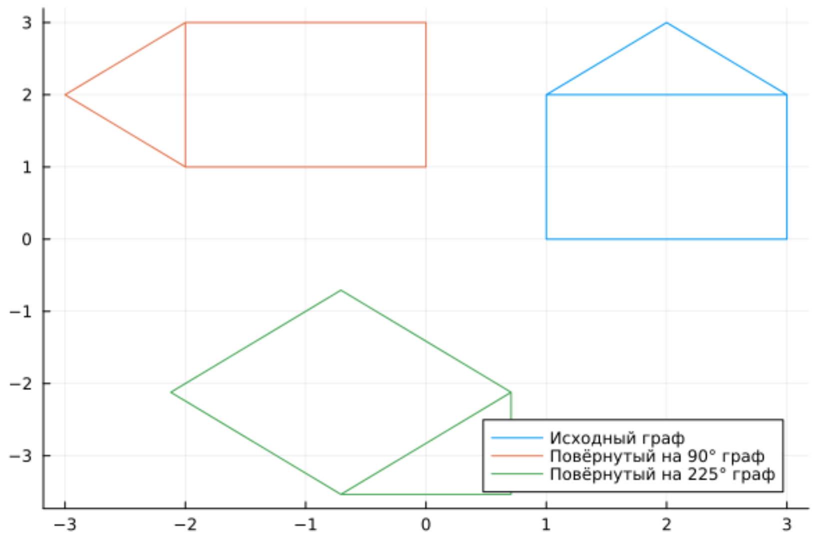

Рис. 3.22: График первоначальной фигуры и повёрнутых на 90° и 225° в Julia

```
# Отражение относительно прямой y = x
fig2 = plot(x, y, label="Исходный граф")

R = [0 1; 1 0]
RD = R * D'
plot!(RD[1,:], RD[2,:], label="Отражённый граф")
savefig(fig2, "fig10.png")

# Дилатация (растяжение) с коэффициентом 2
fig3 = plot(x, y, label="Исходный граф")

T = [2 0; 0 2]
TD = T * D'
plot!(TD[1,:], TD[2,:], label="Растяжённый граф")
savefig(fig3, "fig11.png")
```

Рис. 3.23: Код отражения и дилатации фигуры в Julia

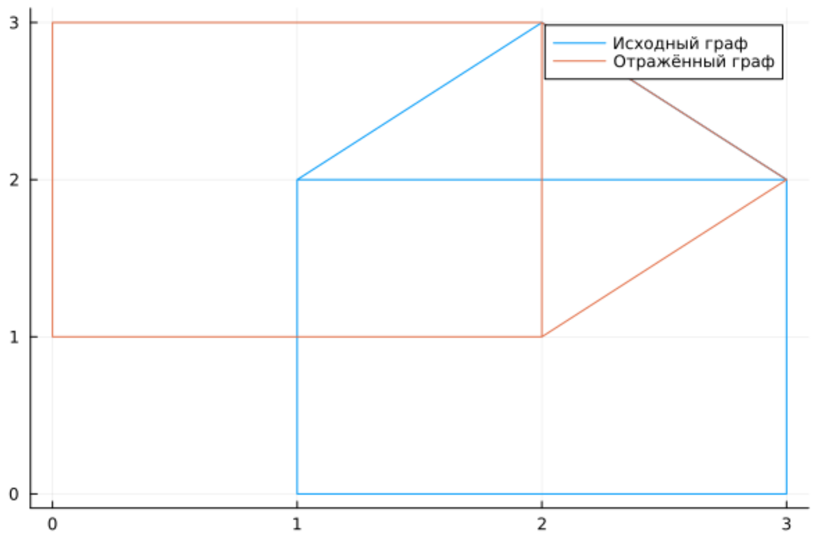

Рис. 3.24: График с первоначальной и отражённой фигурами в Julia

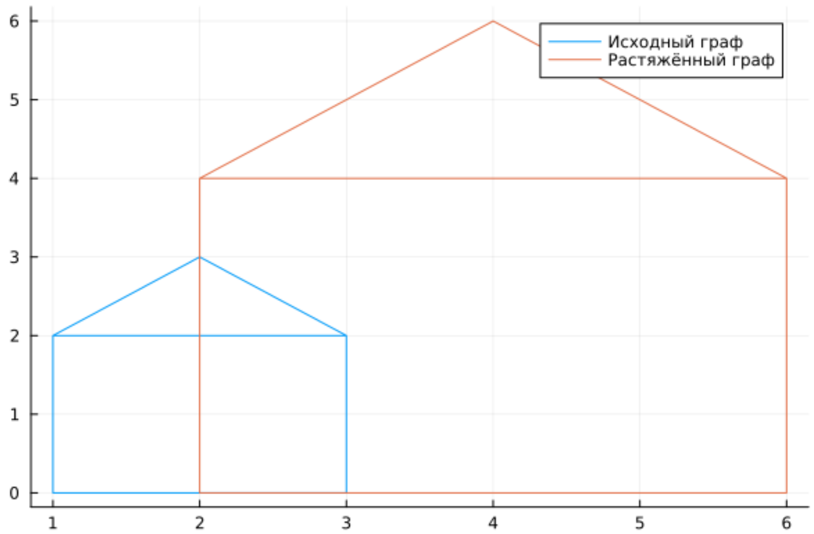

Рис. 3.25: График с первоначальной и увеличенной в два раза фигурами в Julia

4 Выводы

В ходе выполнения лабораторной работы я освоил методы подгонки полиномиальной кривой, матричных преобразований, вращения, отражения и дилатации, и их реализации в Octave и Julia.

Список литературы

1. Кулябов Д. С. Лабораторная работа №5 [Электронный ресурс]. RUDN, 2024. URL: https://esystem.rudn.ru/pluginfile.php/2372906/mod_resource/content/2/README.pdf.