Лабораторная работа №5

Научное программирование

Николаев Дмитрий Иванович, НПМмд-02-24

Содержание

# 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]

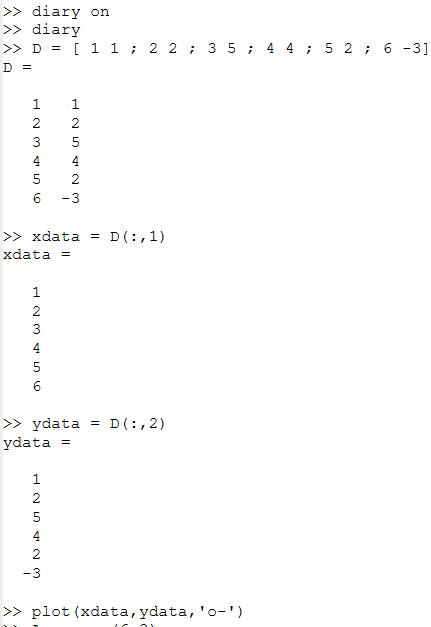
## 3.1 Подгонка полиномиальной кривой

Метод подгонки полиномиальной кривой заключается в нахождении полинома , который наилучшим образом описывает набор точек. Для этого необходимо решить систему линейных уравнений, в которой коэффициенты , и получаются через матричные операции.

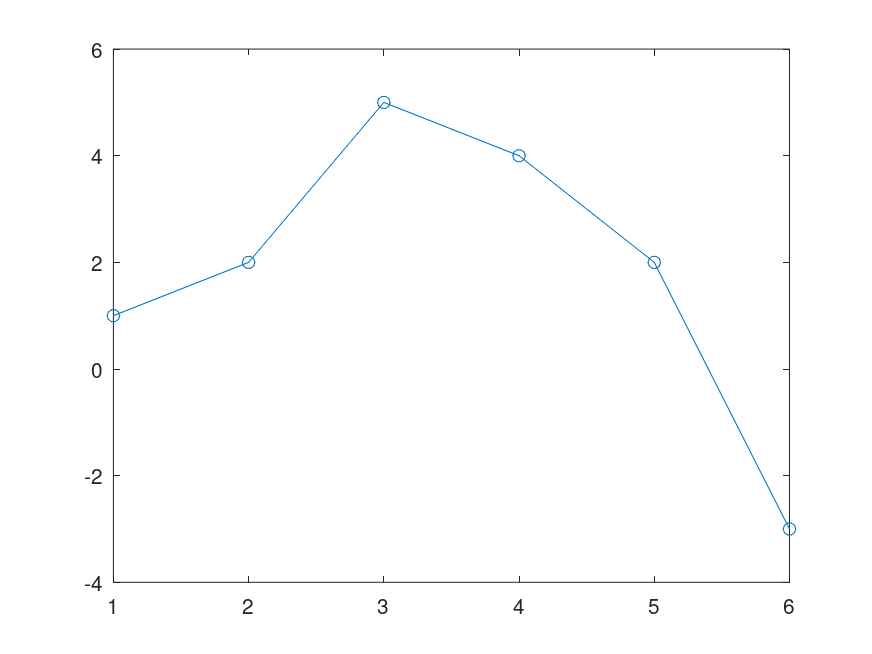
### 3.1.1 Реализация в Octave

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

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



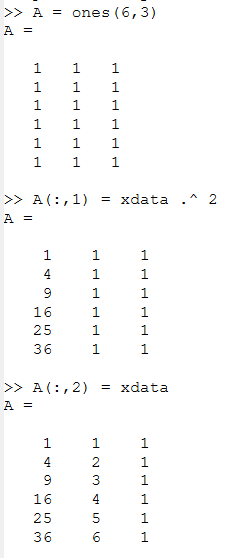
Построение графика по данным в Octave



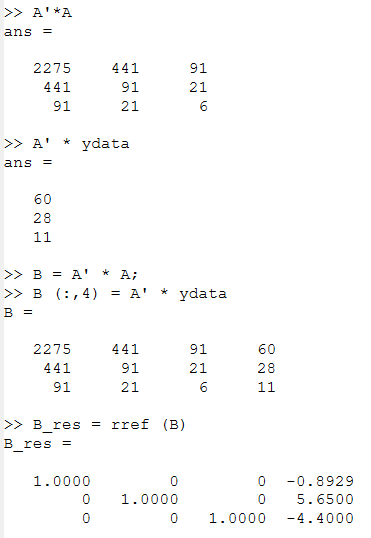
График, построенный по данным в Octave

1. Формируем систему уравнений и решаем методом наименьших квадратов ([??-??]):

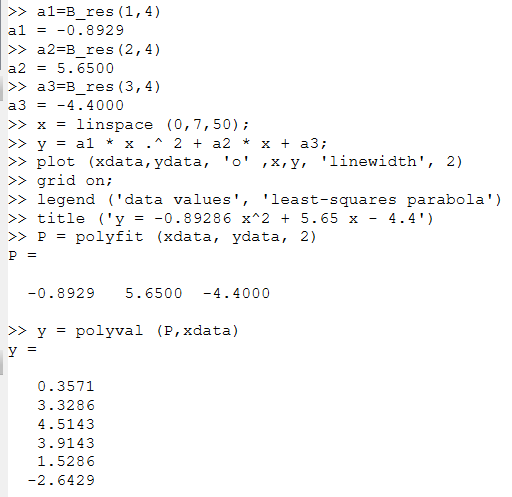
A = ones(6, 3);  
A(:, 1) = xdata.^2;  
A(:, 2) = xdata;  
B = A' \* A;  
B(:, 4) = A' \* ydata;  
B\_res = rref(B);  
a1 = B\_res(1,4);  
a2 = B\_res(2,4);  
a3 = B\_res(3,4);



Формирование системы уравнений в Octave (1/3)



Формирование системы уравнений в Octave (2/3)



Формирование системы уравнений в Octave (3/3)

1. Строим график параболы ([??,??]):

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');

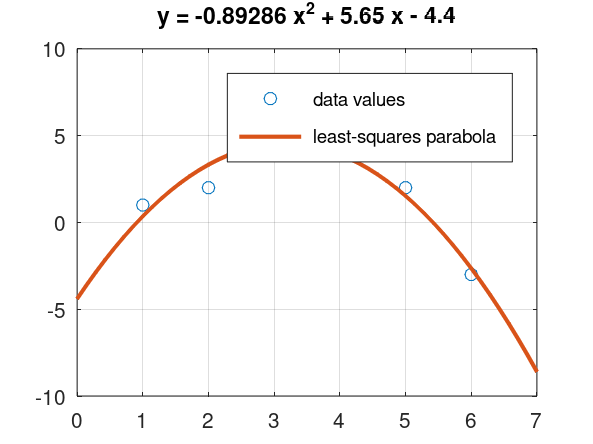


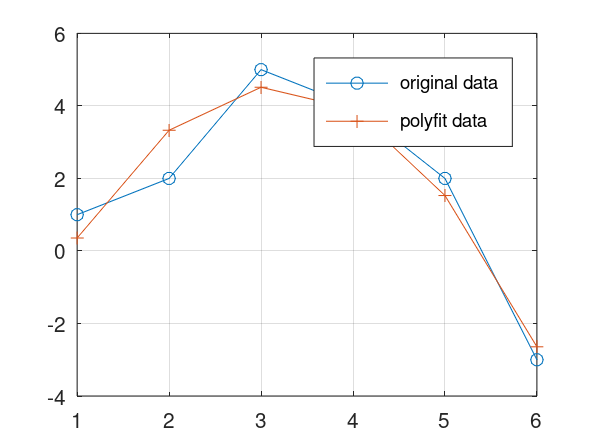
График параболы в Octave

1. Автоматизация с использованием встроенной функции ([??,??,??]):

P = polyfit(xdata, ydata, 2);  
y = polyval(P, xdata);  
plot(xdata, ydata, 'o-', xdata, y, 'r+-');  
grid on;  
legend('original data', 'polyfit data');

Построение графика по точкам параболы в Octave

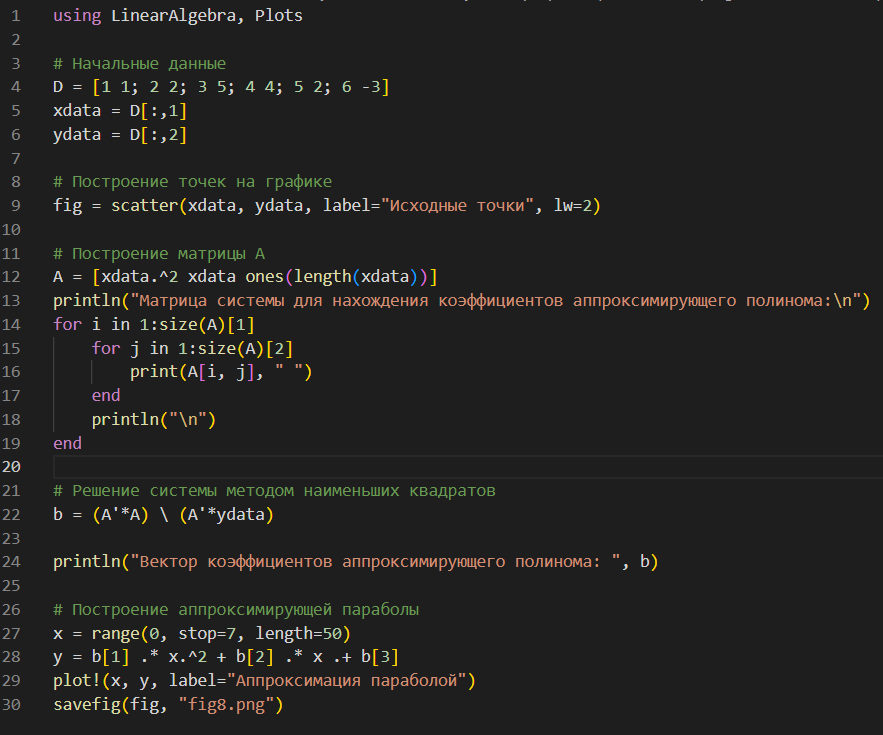
Построение графика по точкам параболы в Octave



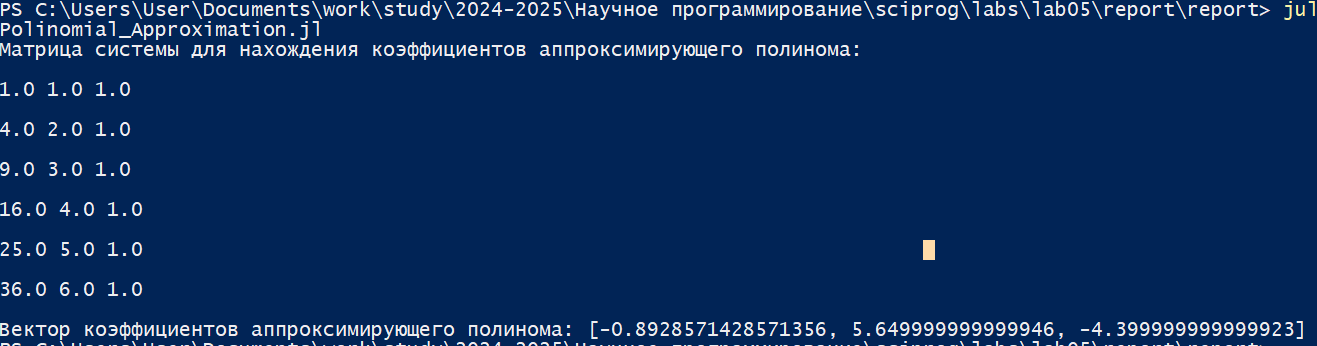
Графики, построенный по данным и по точкам параболы в Octave

### 3.1.2 Реализация в Julia

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



Код подгонки полиномиальной кривой в Julia



Результат подгонки полиномиальной кривой в Julia

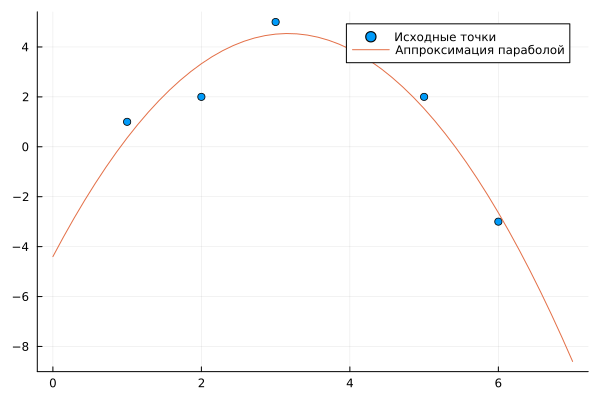


График подгонки полиномиальной кривой в Julia

## 3.2 Матричные преобразования

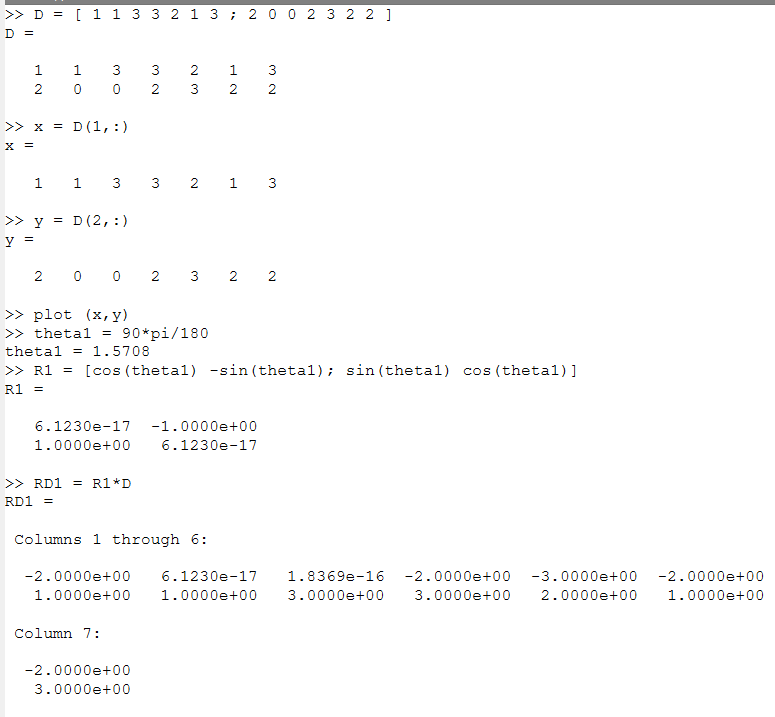
Матричные преобразования позволяют изменять координаты объектов. Каждое преобразование можно описать с помощью матрицы:

* **Вращение** осуществляется с помощью матрицы поворота:
* где — угол поворота.
* **Отражение** относительно оси задаётся матрицей:
* **Дилатация** (масштабирование) осуществляется с помощью матрицы:
* где — коэффициент масштабирования.

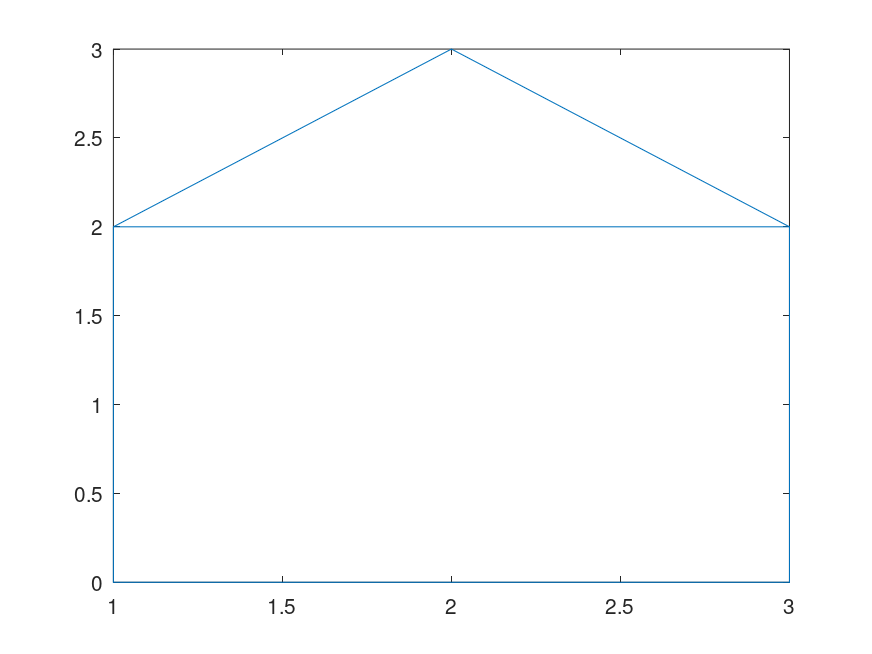
### 3.2.1 Реализация в Octave

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

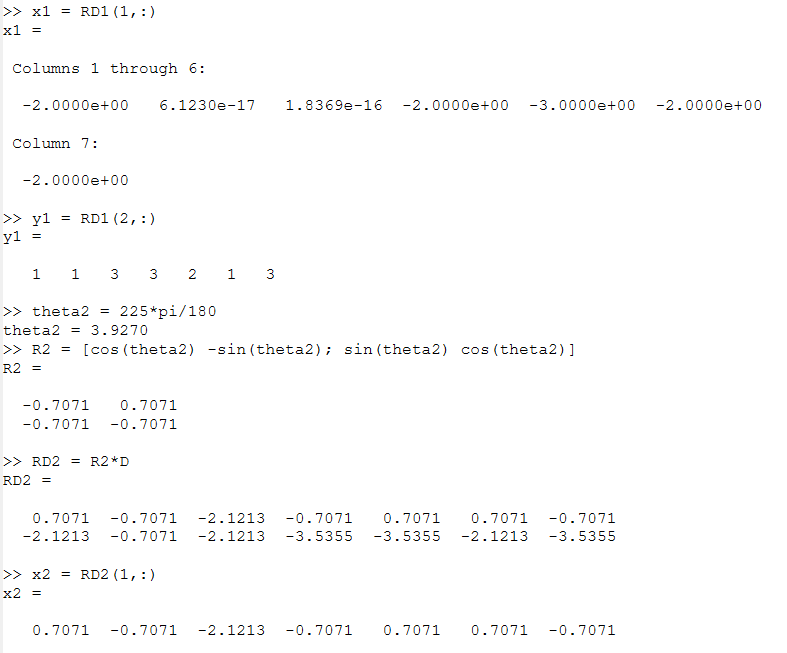
theta1 = 90 \* pi / 180;  
R1 = [cos(theta1), -sin(theta1); sin(theta1), cos(theta1)];  
RD1 = R1 \* D';  
plot(RD1(1,:), RD1(2,:), 'ro-');  
theta2 = 225 \* pi / 180;  
R2 = [cos(theta2), -sin(theta2); sin(theta2), cos(theta2)];  
RD2 = R2 \* D';  
plot(x, y, 'bo-', RD1(1,:), RD1(2,:), 'ro-', RD2(1,:),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



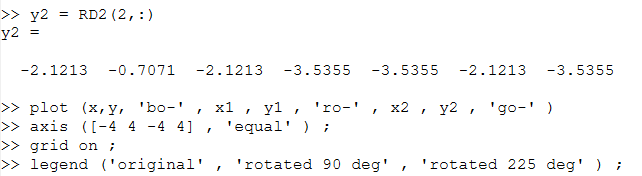
Первоначальная фигура. Вращение в Octave (1/3)



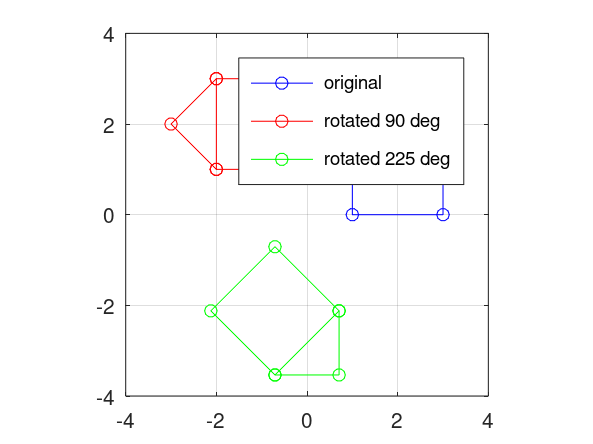
Первоначальная фигура в Octave



Вращение в Octave (2/3)



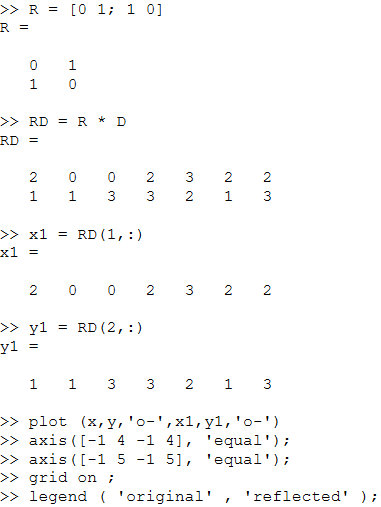
Вращение в Octave (3/3)



Вращение фигуры на 90° и 225° в Octave

1. Отражение относительно прямой ([??,??]):

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



Отражение фигуры относительно прямой в Octave

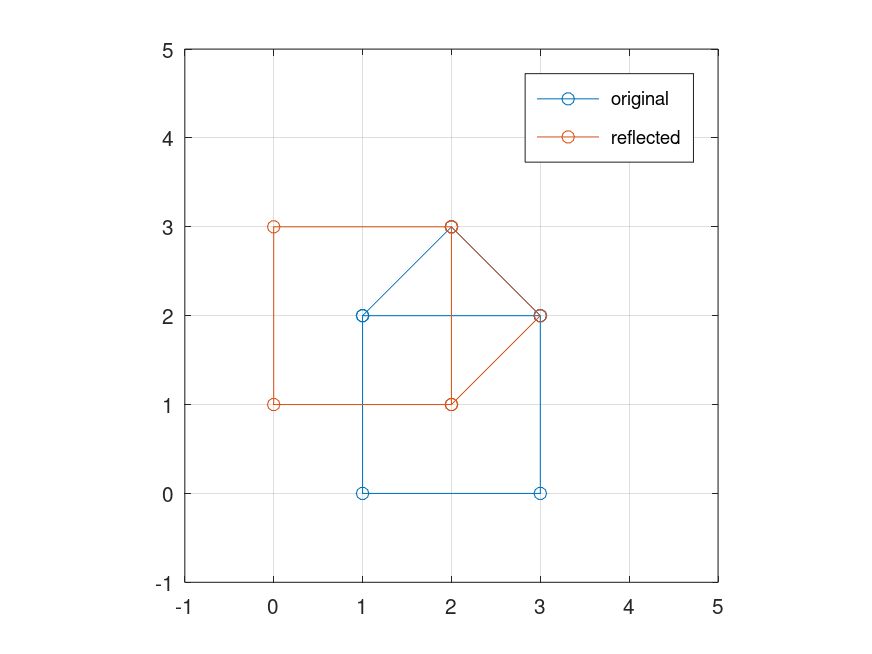
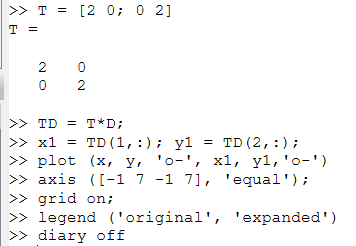


График с первоначальной и отражённой фигурами в Octave

1. Дилатация (увеличение) в 2 раза ([??,??]):

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



Двойная дилатация фигуры в Octave

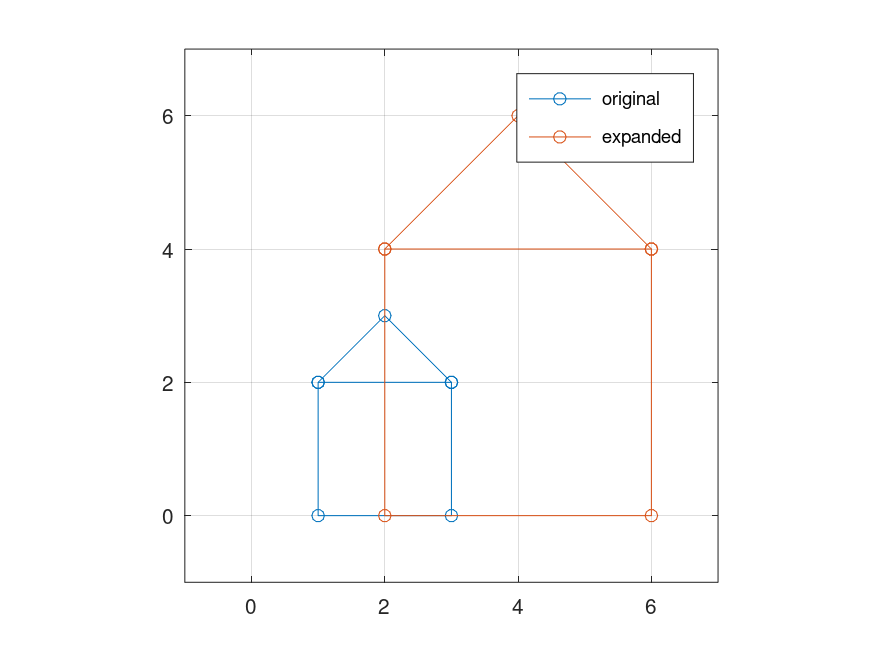
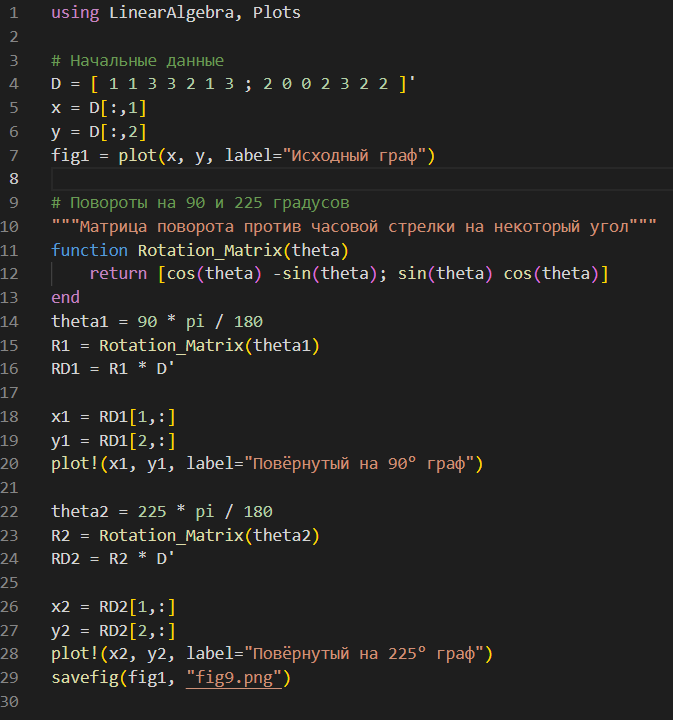


График с первоначальной и увеличенной в два раза фигурами в Octave

### 3.2.2 Реализация в Julia

Реализация вращения ([??,??]), отражения и дилатации в два раза ([??,??,??]):



Код вращения фигуры на 90° и 225° в Julia

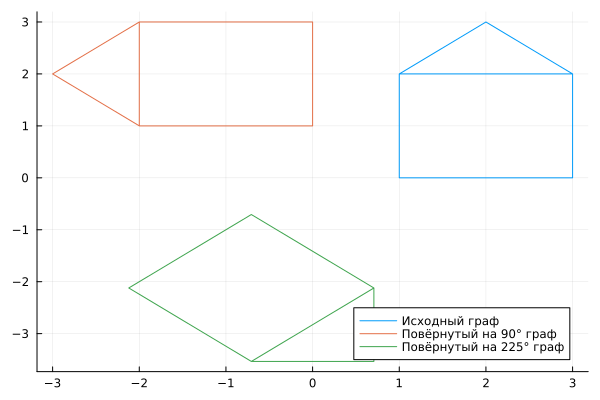
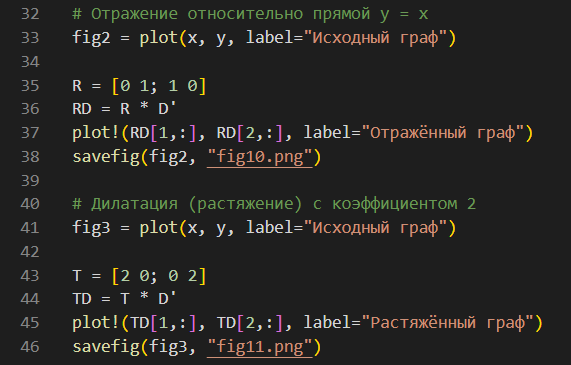


График первоначальной фигуры и повёрнутых на 90° и 225° в Julia



Код отражения и дилатации фигуры в Julia

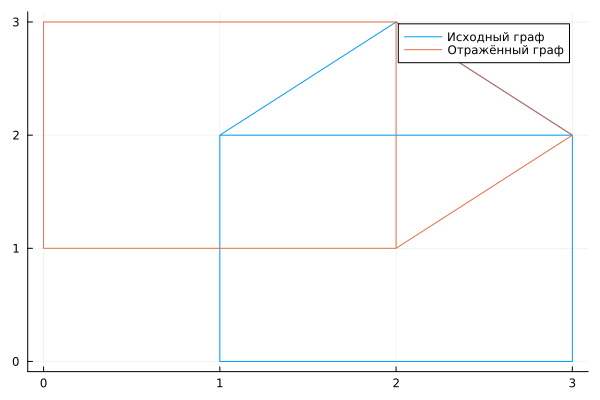


График с первоначальной и отражённой фигурами в Julia

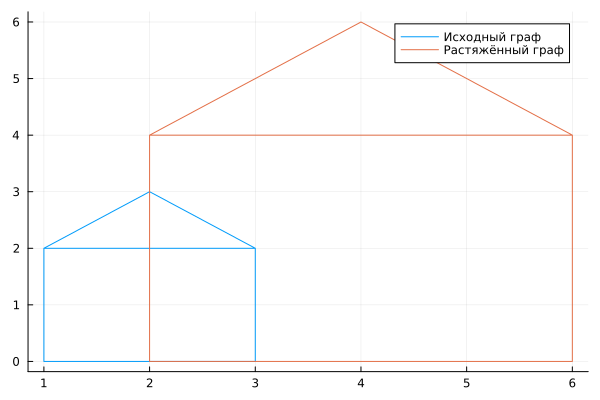


График с первоначальной и увеличенной в два раза фигурами в Julia

# 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>.