

Лабораторная работа 5

Отчет по лабораторной работе 5

Милёхин Александр НПМмд-02-21

Содержание

1	Цель работы	4
2	Теоретические сведения	5
3	Задание	6
4	Выполнение лабораторной работы	7
5	Выводы	28

List of Figures

4.1	Ввод матрицы данных	8
4.2	Нанесение точек на плоскость	9
4.3	Создание матрицы А	12
4.4	Построение уравнений по методу наименьших квадратов.....	13
4.5	Решение задачи методом Гаусса.....	14
4.6	Построение графика параболы	14
4.7	График параболы	15
4.8	Подгоночный полином	16
4.9	Граф исходных и подгоночных данных	16
4.10	Реализация построения графа	18
4.11	Полученный граф	19
4.12	Поворот на 90 градусов	21
4.13	Поворот на 225 градусов	22
4.14	Реализация и результаты вращения.....	23
4.15	Задание отражения.....	24
4.16	Результат отражения	25
4.17	Реализация дилатации	26
4.18	Результат увеличения.....	27

1 Цель работы

Ознакомиться с некоторыми операциями в среде Octave для решения таких задач, как подгонка полиномиальной кривой, матричных преобразований, вращений, отражений и дилатаций.

2 Теоретические сведения

Вся теоретическая часть по выполнению лабораторной работы была взята из инструкции по лабораторной работе №5 (“Лабораторная работа №5. Описание”) на сайте: <https://esystem.rudn.ru/course/view.php?id=12766>

3 Задание

Выполните работу и задокументируйте процесс выполнения.

4 Выполнение лабораторной работы

1. Подгонка полиномиальной кривой

В статистике часто рассматривается проблема подгонки прямой линии к набору данных. Решим более общую проблему подгонки полинома к множеству точек. Пусть нам нужно найти параболу по методу наименьших квадратов для набора точек, заданных матрицей:

$$D = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 5 \\ 4 & 4 \\ 5 & 2 \\ 6 & -3 \end{pmatrix}$$

В матрице заданы значения x в столбце 1 и значения y в столбце 2. Введём матрицу данных в Octave и извлечём вектора x и y . Данные операции показаны на Fig. 1.

```
C:\Users\alexm
Командное окно
For information about changes from previous versions, type 'news' ^

>> diary on
>> 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

>> |
```

Figure 4.1: Ввод матрицы данных

Нарисуем точки на графике, см. Fig. 2.

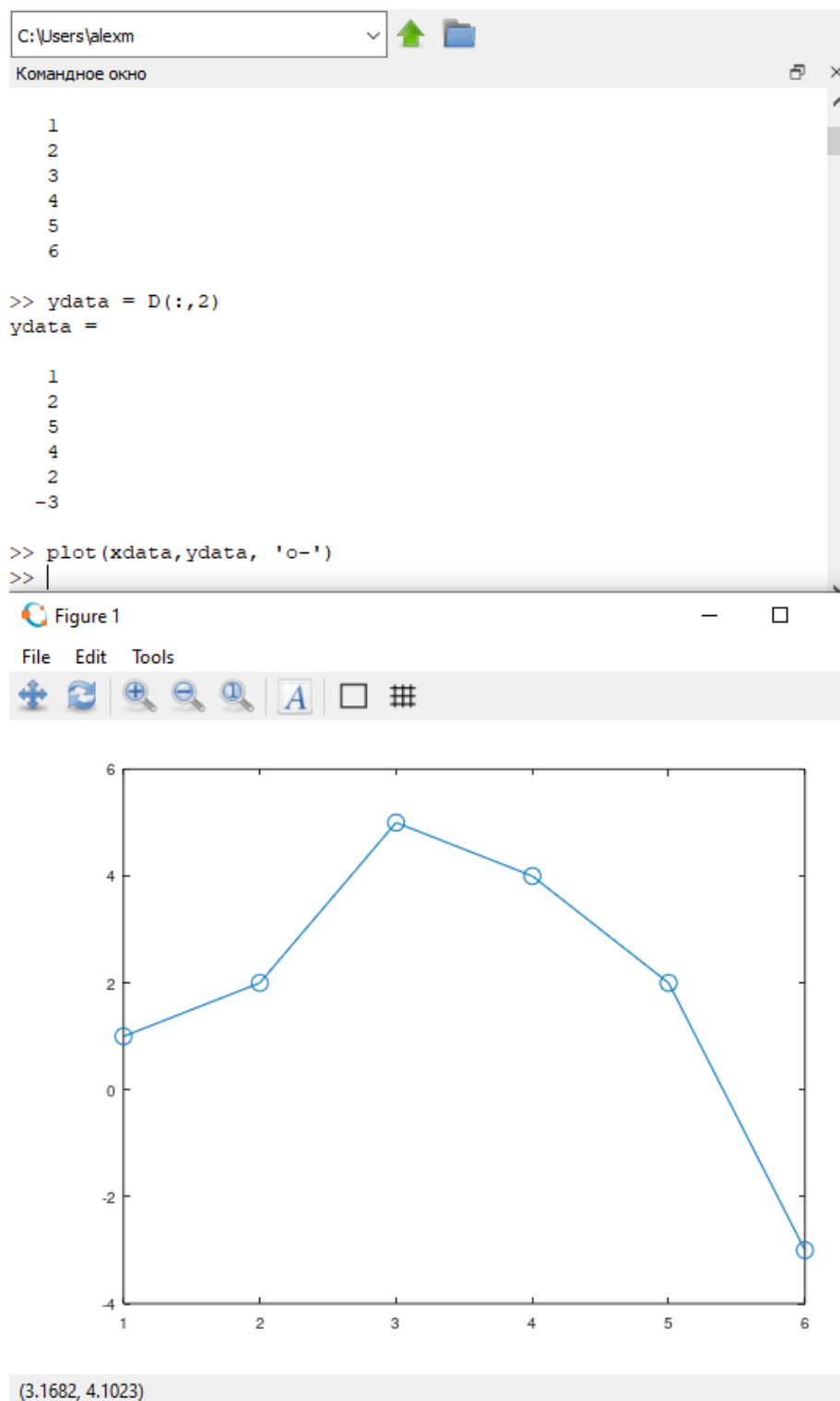


Figure 4.2: Нанесение точек на плоскость

Построим уравнение вида $y = ax^2 + bx + c$. Подставляя данные, получаем следующую систему линейных уравнений.

$$\begin{pmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \\ 36 & 6 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 5 \\ 4 \\ 2 \\ -3 \end{pmatrix}.$$

Обратим внимание на форму матрицы коэффициентов A . Третий столбец – все единицы, второй столбец – значения x , а первый столбец – квадрат значений x . Правый вектор – это значения y . Есть несколько способов построить матрицу

коэффициентов в Octave. Один из подходов состоит в том, чтобы использовать команду `ones` для создания матрицы единиц соответствующего размера, а затем перезаписать первый и второй столбцы необходимыми данными. Это показано на Fig. 3.

```
C:\Users\alexm
Командное окно
>> plot(xdata,ydata, 'o-')
>> 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

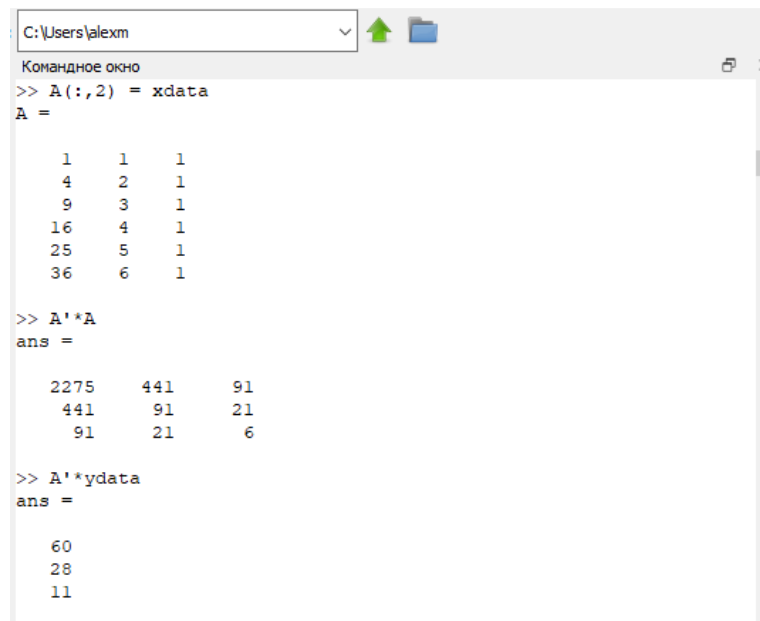
>> A(:,2) = xdata
A =

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

>> |
```

Figure 4.3: Создание матрицы A

Решение по методу наименьших квадратов получается из решения уравнения $A^T A b = A^T b$, где b – вектор коэффициентов полинома. Используем Octave для построения уравнений, как показано на Fig. 4



```

C:\Users\alexm
Командное окно
>> A(:,2) = xdata
A =

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

>> A'*A
ans =

   2275   441   91
   441    91   21
    91    21    6

>> A'*ydata
ans =

   60
   28
   11
  
```

Figure 4.4: Построение уравнений по методу наименьших квадратов

Решим задачу методом Гаусса (См. Fig. 5). Для этого запишем расширенную матрицу:

$$B = \begin{pmatrix} 2275 & 441 & 91 & 60 \\ 441 & 91 & 21 & 28 \\ 91 & 21 & 6 & 11 \end{pmatrix}.$$

Таким образом, искомое квадратное уравнение имеет вид:

$$y = -0.89286x^2 + 5.65x - 4.4$$

```

C:\Users\alexm
Командное окно
91    21    6

>> A'*ydata
ans =

    60
    28
    11

>> A' * ydata
ans =

    60
    28
    11

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

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

>> a1 = B_res(1,4)
a1 = -0.8929
>> a2 = B_res(2,4)
a2 = 5.6500
>> a3 = B_res(3,4)
a3 = -4.4000
>> |

```

Figure 4.5: Решение задачи методом Гаусса

После чего построим соответствующий график параболы. Построение можно увидеть на Figure 6, а вид самой параболы на Figure 7.

```

C:\Users\alexm
Командное окно
28
11

>> A'*ydata
ans =

    60
    28
    11

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

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

>> 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=-0.89286x^2 + 5.65x - 4.4')
>> |

```

Figure 4.6: Построение графика параболы

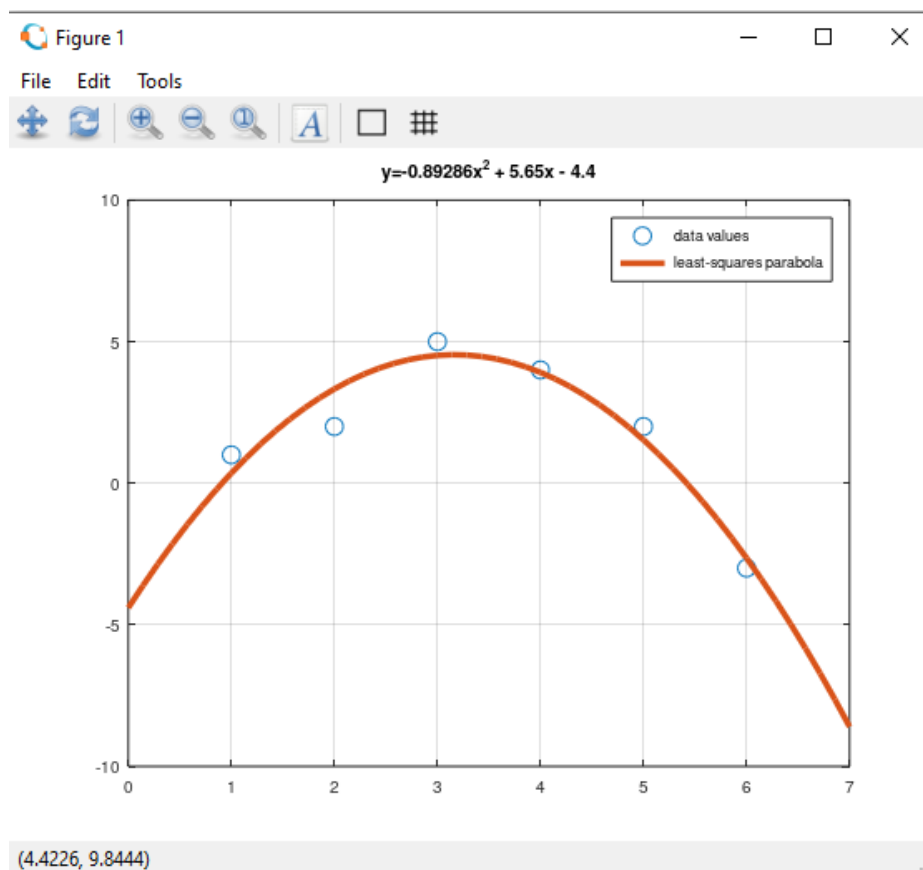


Figure 4.7: График параболы

Процесс подгонки может быть автоматизирован встроенными функциями Octave. Для этого мы можем использовать встроенную функцию для подгонки полинома `polyfit`. Синтаксис: `polyfit (x, y, order)`, где `order` – это степень полинома. Значения полинома `P` в точках, задаваемых вектором-строкой `x` можно получить с помощью функции `polyval`. Синтаксис: `polyval (P, x)`.

На Figure 8 получим подгоночный полином.

```

C:\Users\alexm
Командное окно
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=-0.89286x^2 + 5.65x - 4.4')
>> p = polyfit(xdata, ydata, 2)
p =
    -0.8929    5.6500   -4.4000

>> y = polyval(p, xdata)
error: 'P' undefined near line 1, column 1
>> y = polyval(p, xdata)
y =
    0.3571
    3.3286
    4.5143
    3.9143
    1.5286
   -2.6429

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

```

Figure 4.8: Подгоночный полином

После чего рассчитаем значения в точках и построим исходные данные. Это показано на Fig. 9.

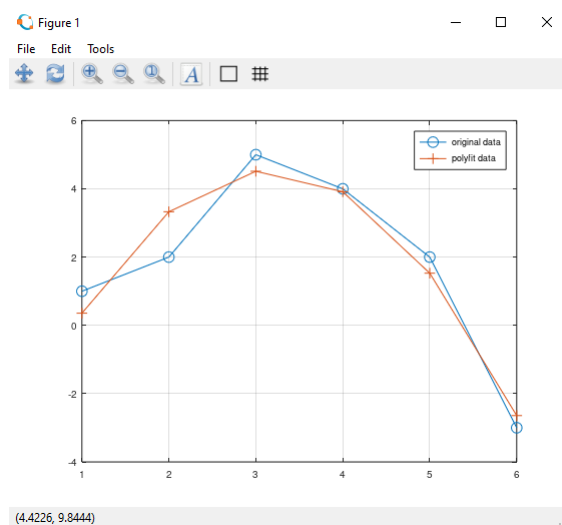


Figure 4.9: Граф исходных и подгоночных данных

2. Матричные преобразования

Матрицы и матричные преобразования играют ключевую роль в компьютерной графике. Существует несколько способов представления изображения в виде матрицы. Подход, который мы здесь используем, состоит в том, чтобы перечислить ряд вершин, которые соединены последовательно, чтобы получить ребра простого графа. Мы записываем это как матрицу $2 \times n$, где каждый столбец представляет точку на рисунке. В качестве простого примера, давайте попробуем закодировать граф-домик. Есть много способов закодировать это как матрицу. Эффективный метод состоит в том, чтобы выбрать путь, который проходит по каждому ребру ровно один раз (цикл Эйлера).

$$D = \begin{pmatrix} 1 & 1 & 3 & 3 & 2 & 1 & 3 \\ 2 & 0 & 0 & 2 & 3 & 2 & 2 \end{pmatrix}.$$

Реализация показана на Figure 10.

```
C:\Users\alexm
Командное окно
error: 'P' undefined near line 1, column 1
>> y = polyval(p, xdata)
y =

    0.3571
    3.3286
    4.5143
    3.9143
    1.5286
   -2.6429

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

Figure 4.10: Реализация построения графа

Полученный граф можно увидеть на Figure 11.

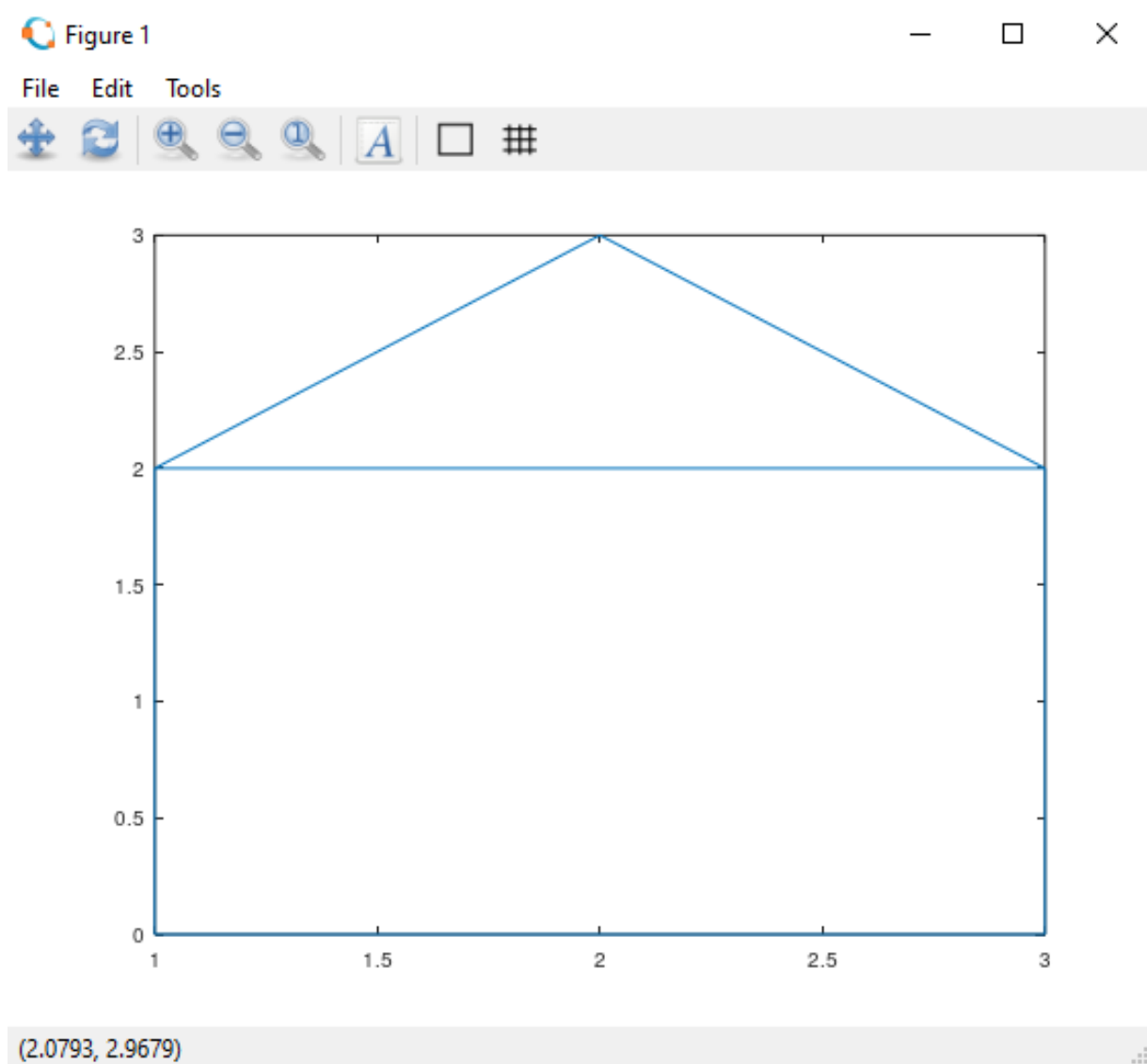


Figure 4.11: Полученный граф

3. Вращение

Рассмотрим различные способы преобразования изображения. Вращения могут быть получены с использованием умножения на специальную матрицу. Вращение точки (x, y) относительно начала координат определяется как

$$R \begin{pmatrix} x \\ y \end{pmatrix},$$

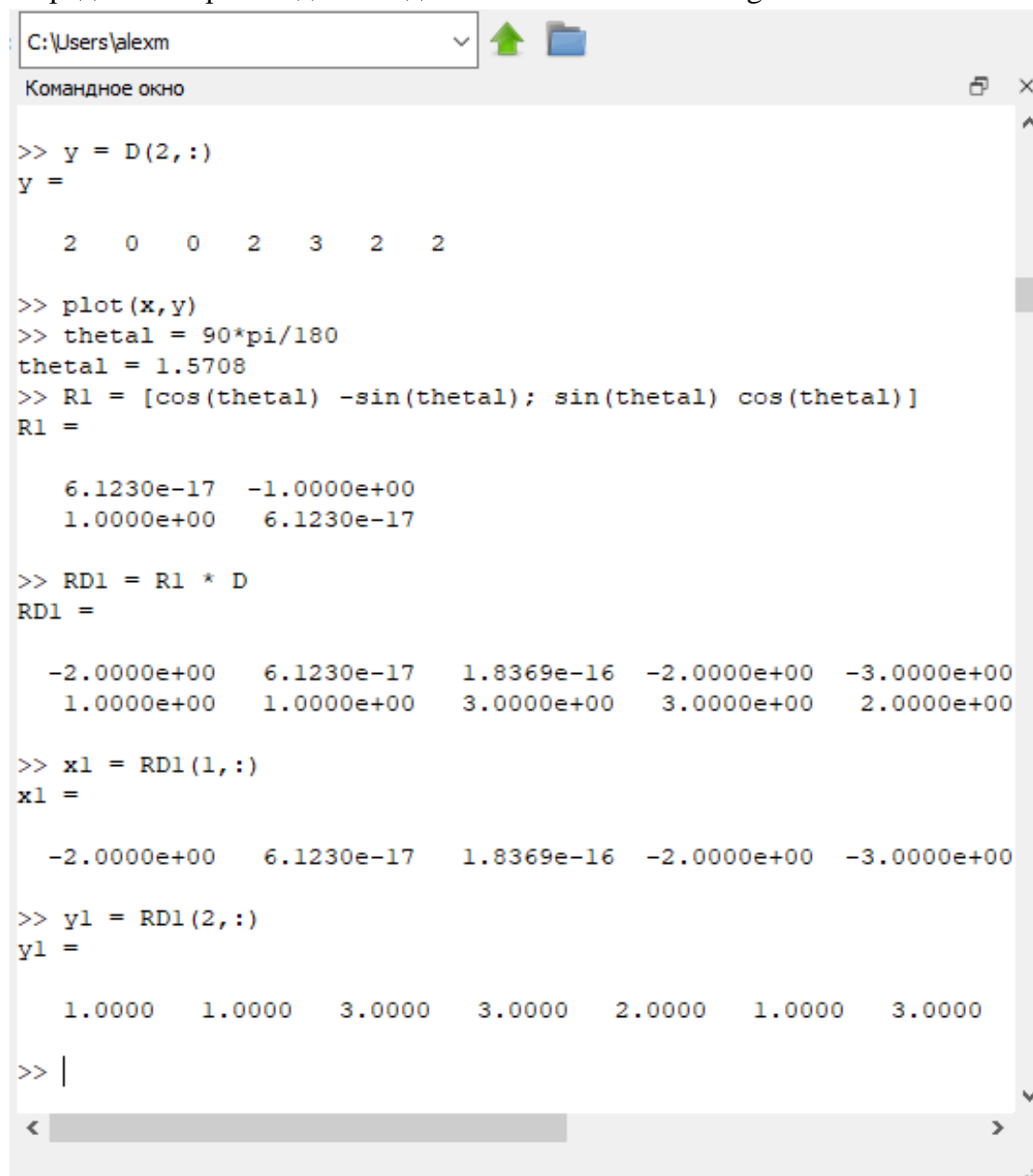
где

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

θ - угол поворота (измеренный против часовой стрелки).

Теперь, чтобы произвести повороты матрицы данных D , нам нужно вычислить произведение матриц RD . Повернём граф дома на 90° и 225° . Вначале переведём

угол в радианы. Произведенные действия показаны на Figure 12 - 14.



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

    2    0    0    2    3    2    2

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

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

>> RD1 = R1 * D
RD1 =

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

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

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

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

    1.0000    1.0000    3.0000    3.0000    2.0000    1.0000    3.0000

>> |
```

Figure 4.12: Поворот на 90 градусов

```
C:\Users\alexm
Командное окно

1.0000  1.0000  3.0000  3.0000  2.0000  1.0000  3.0000

>> 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

>> 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');
>> |
```

Figure 4.13: Поворот на 225 градусов

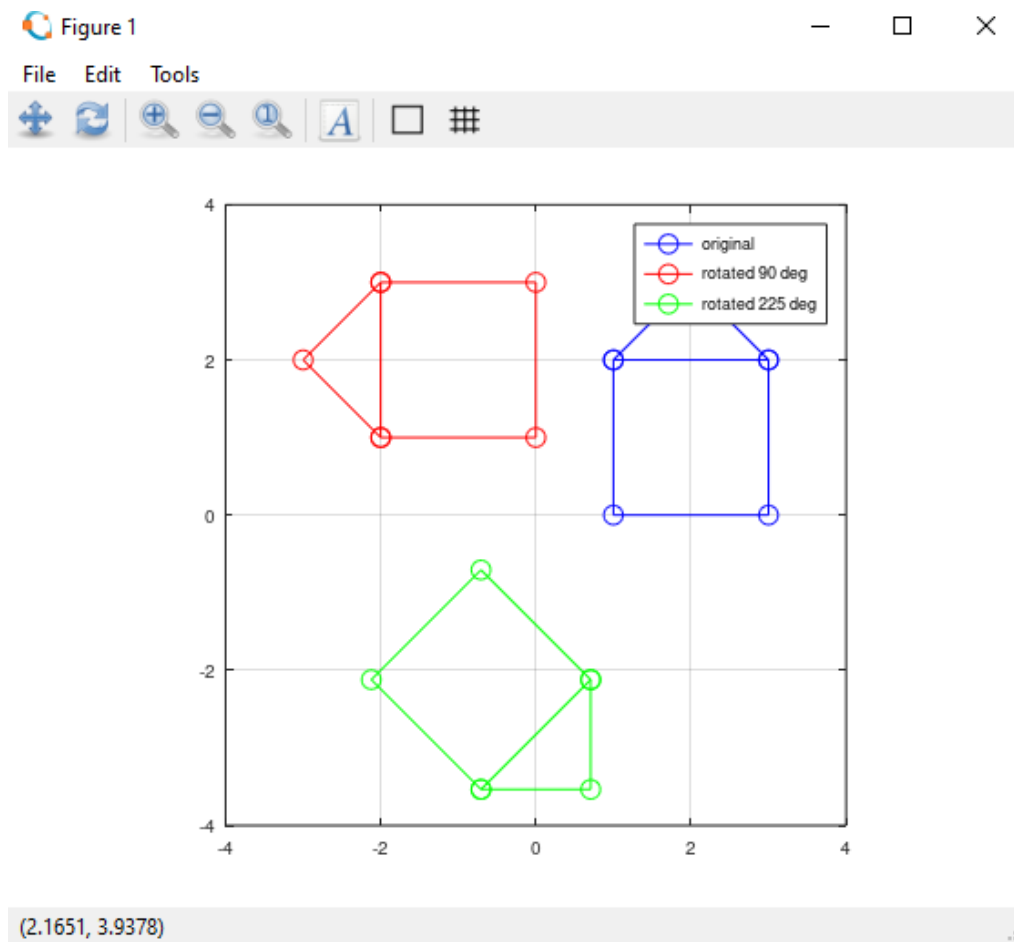


Figure 4.14: Реализация и результаты вращения

4. Отражение

Если l – прямая, проходящая через начало координат, то отражение точки (x, y) относительно прямой l определяется как

$$R \begin{pmatrix} x \\ y \end{pmatrix},$$

где

$$R = \begin{pmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{pmatrix},$$

θ - угол между прямой l и осью абсцисс (измеренный против часовой стрелки). Отразим граф дома относительно прямой $y = x$. Зададим матрицу отражения, как показано на Figure 15.

```
C:\Users\alexm
Командное окно
>> 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');
>> 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')
>> |
```

Figure 4.15: Задание отражения

Далее на Figure 16 показано, какой результат получился в ходе этих действий.

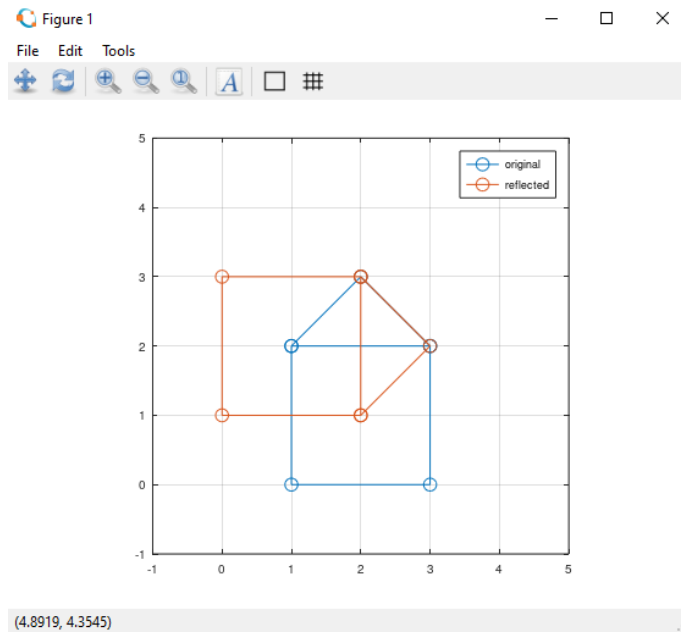


Figure 4.16: Результат отражения

5. Дилатация

Дилатация (то есть расширение или сжатие) также может быть выполнено путём умножения матриц. Пусть:

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

Тогда матричное произведение TD будет преобразованием дилатации D с коэффициентом k . Увеличим граф дома в 2 раза. Реализация показана на Figure17.

```

C:\Users\alexm
Командное окно
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')
>> T = [2 0; 0 2]
T =
    2    0
    0    2
>> TD = T * D
TD =
    2    2    6    6    4    2    6
    4    0    0    4    6    4    4
>> x1 = TD(1,:); y1 = TD(2,:);
>> plot(x, y, 'o-', x1, y1, 'o-')
>> axis([-1 7 -1 7], 'equal');
>> grid on;
>> legend('original', 'expended')
>>

```

Figure 4.17: Реализация дилатации

После чего на Figure 18 можно увидеть результат данной операции.

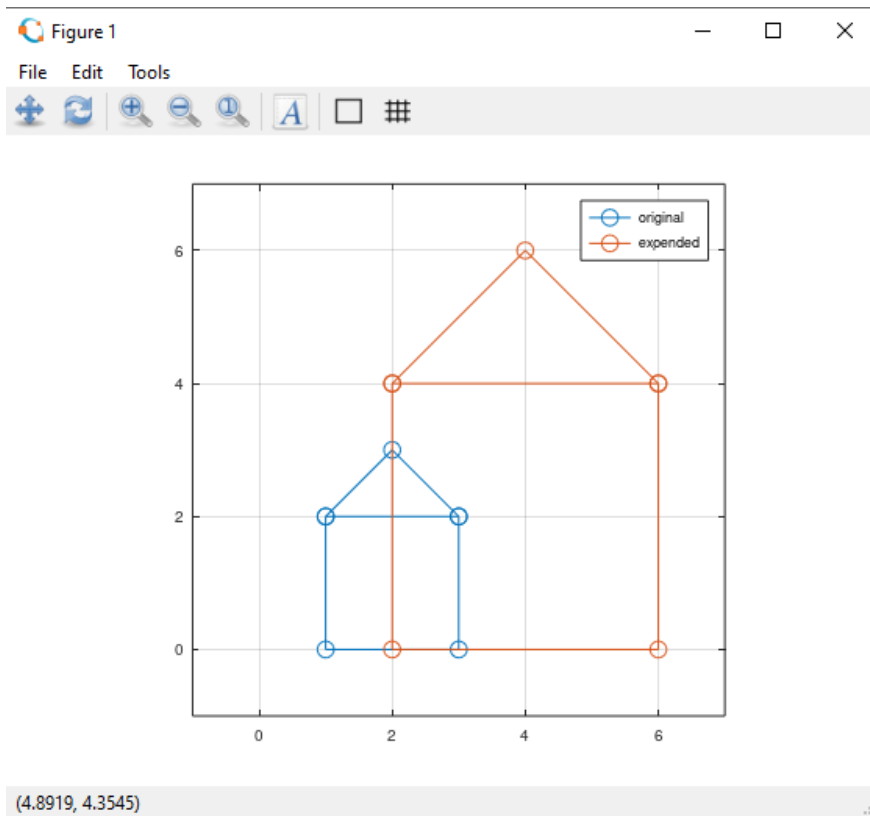


Figure 4.18: Результат увеличения

5 Выводы

Я ознакомился с некоторыми операциями в среде Octave для решения таких задач, как подгонка полиномиальной кривой, матричных преобразований, вращений, отражений и дилатаций.