Шаблон отчёта по лабораторной работе

Простейший вариант

Турсунов Баходурхон Азимджонович

Содержание

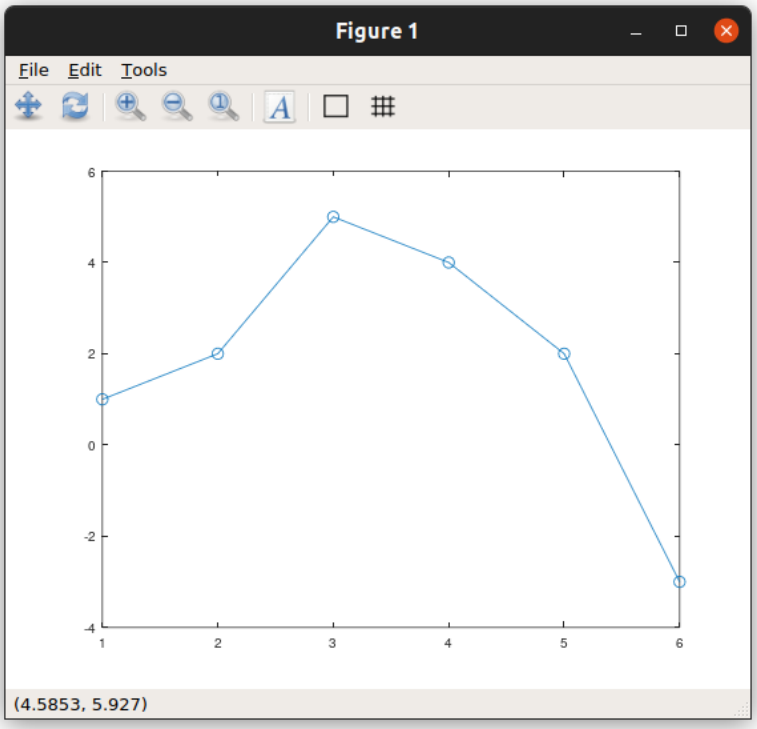
Выполнение 5 лабораторной работы

Подгонка полиномиальной кривой

1. 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. 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.

После рисуем точки графике. Результат (Рис 1)

```
>> 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

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



(Рис 1)

2. Далее с помощью команды *ones* создаем еденичную матрицу соответствующего размера, а затем переписываем первый и второй столбцы необходимыми данными:

```
>> 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 2)

3. Решение по методу наименьших квадратов получается из решения уравнения(Рис 3)

```
>> A'*A
ans =

   2275    441     91
    441     91     21
     91     21      6

>> A'*ydata
ans =

   60
   28
   11
```

(Рис 3)

4. Решим задачу методом Гаусса. Запишем расширенную матрицу:

```
>> 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
```

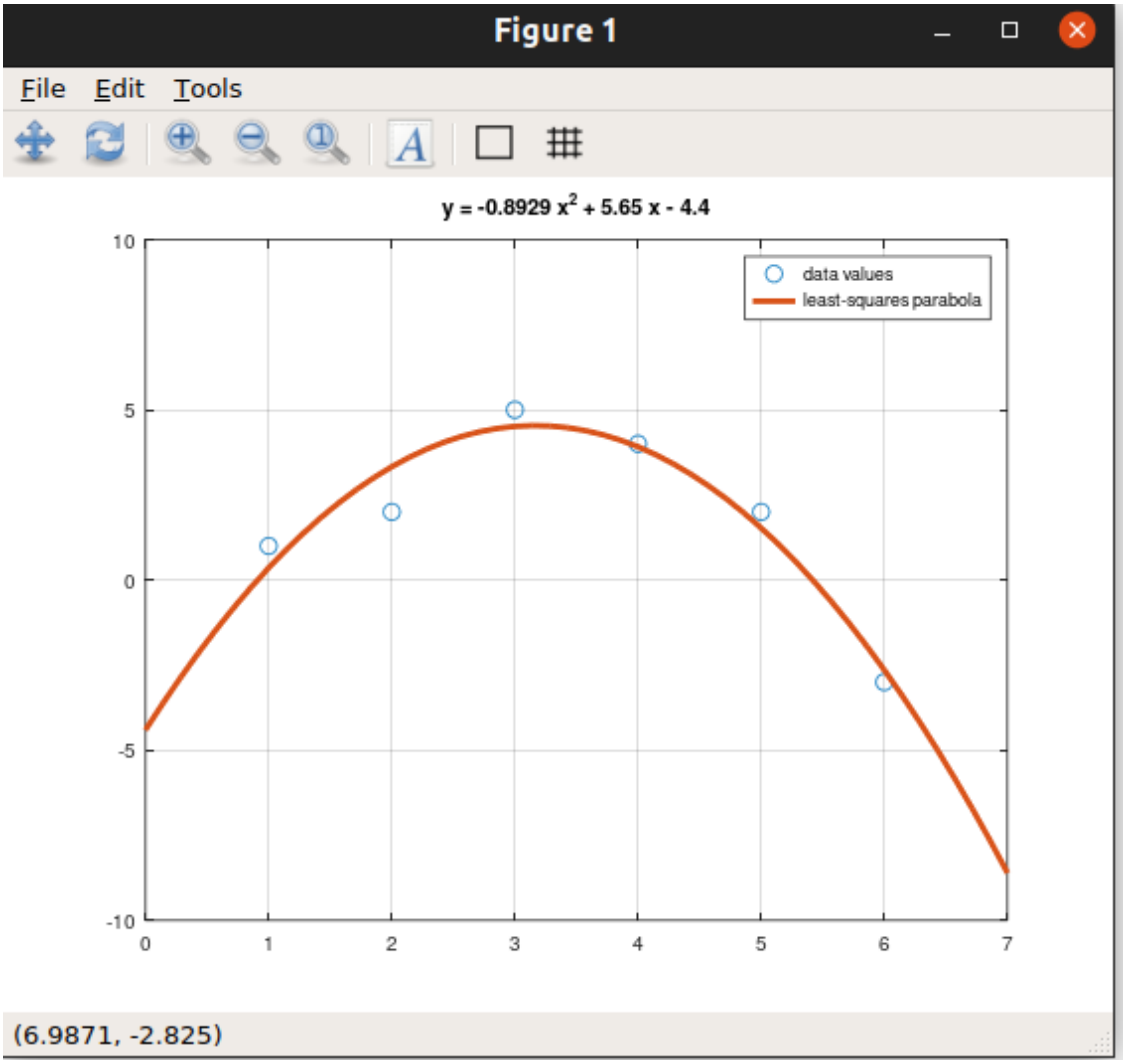
(Рис 4)

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

(Рис 5)

5. Таким образом, искомое квадратное уравнение имеет вид $y = -0.89286x2 + 5.65x - 4.4$. Построим соответствующий график параболы.

В итоге получили подобный граф

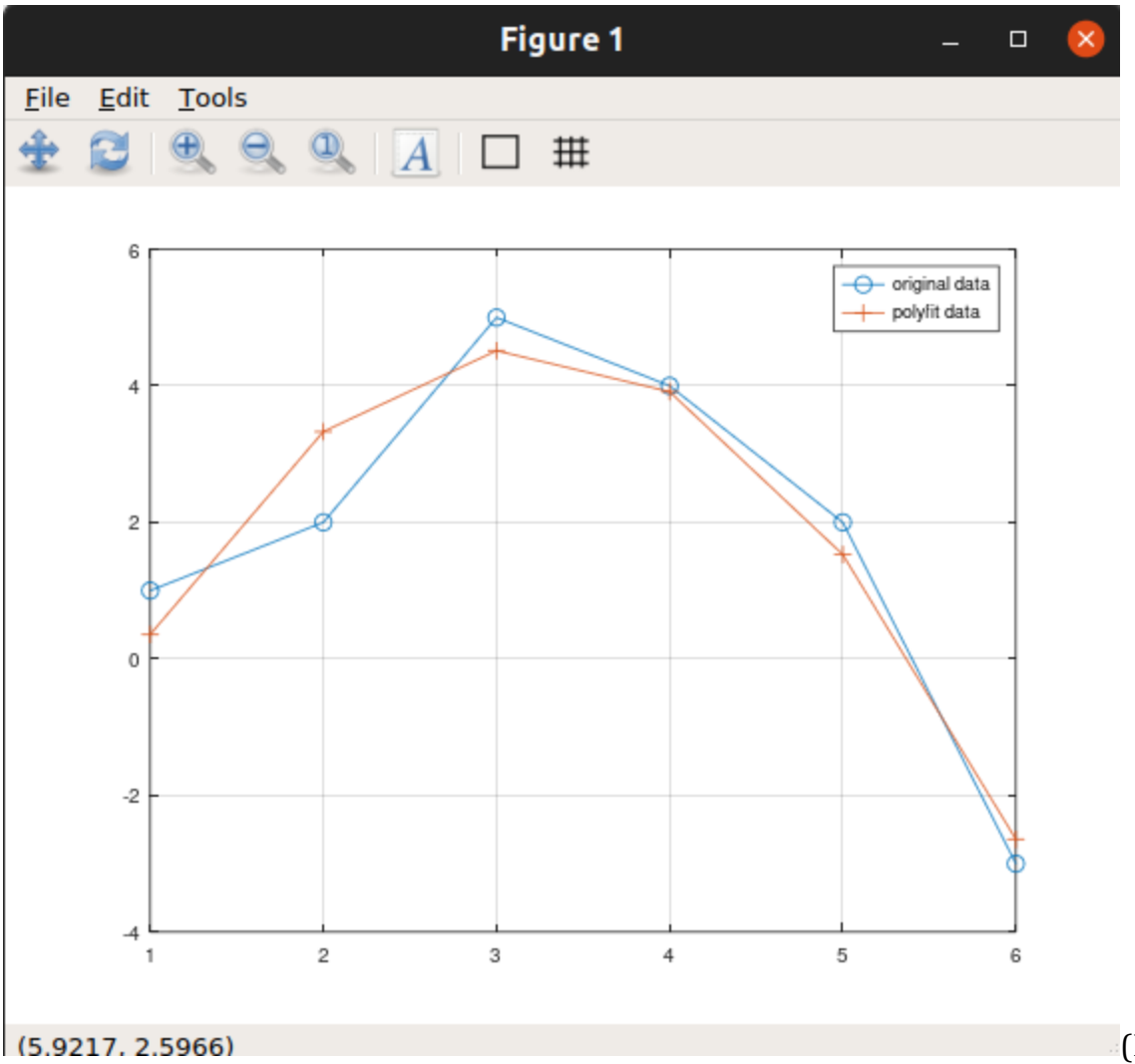


(Рис 6)

6. 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. Синтаксиса: polyval (P, x). Получим подгоночный полином.

```
>> 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

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

(Рис 7)



(Рис 8)

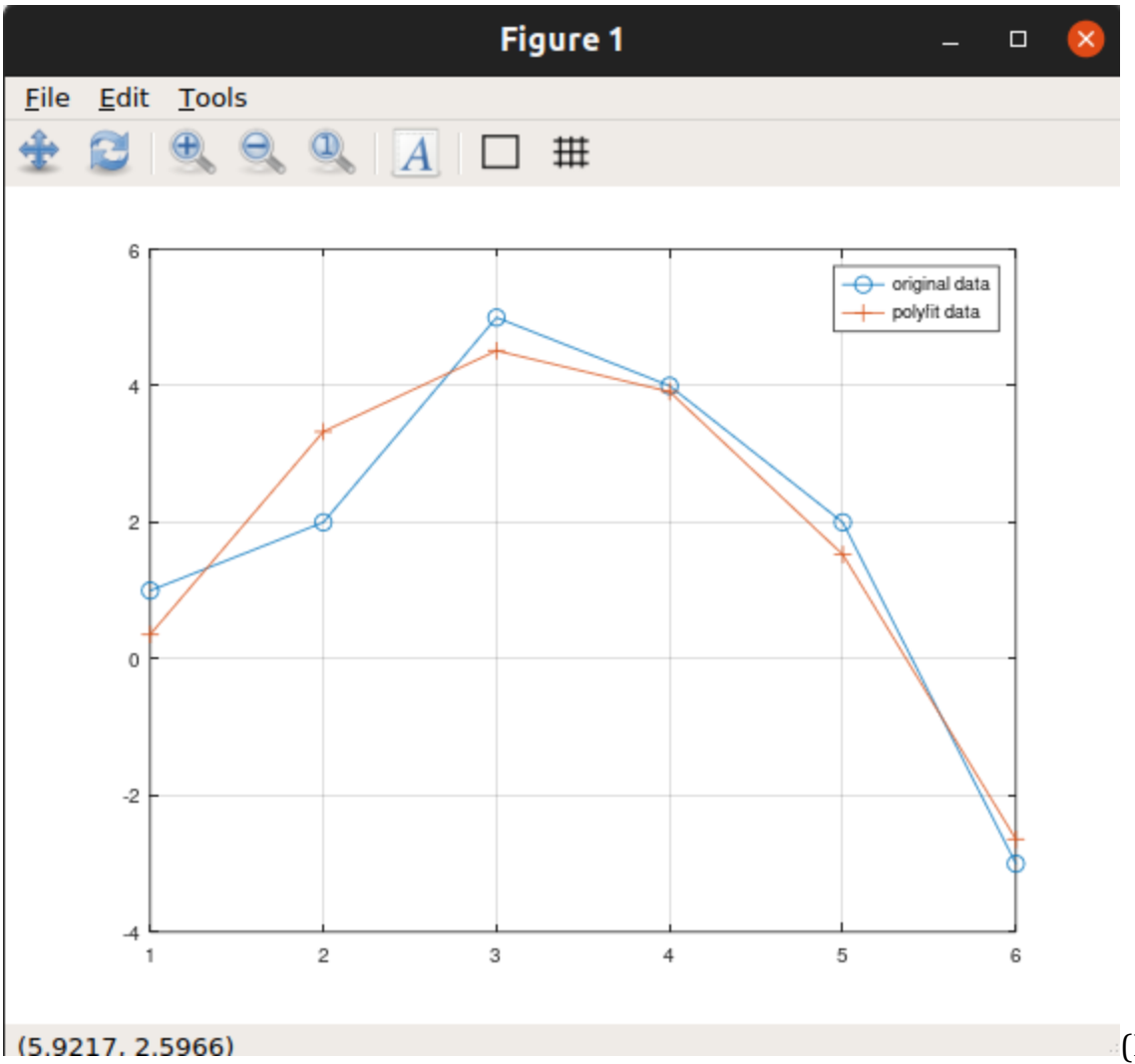
Матричные преобразования

1. 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 = [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)
```

(Рис 8)

В итоге получился такой граф (Рис 9)



(Рис 9)

Вращение

1. В этом пункте мы попробуем повернуть граф дома на 90 и 225 градусов. Вначале переведем угол в радианы

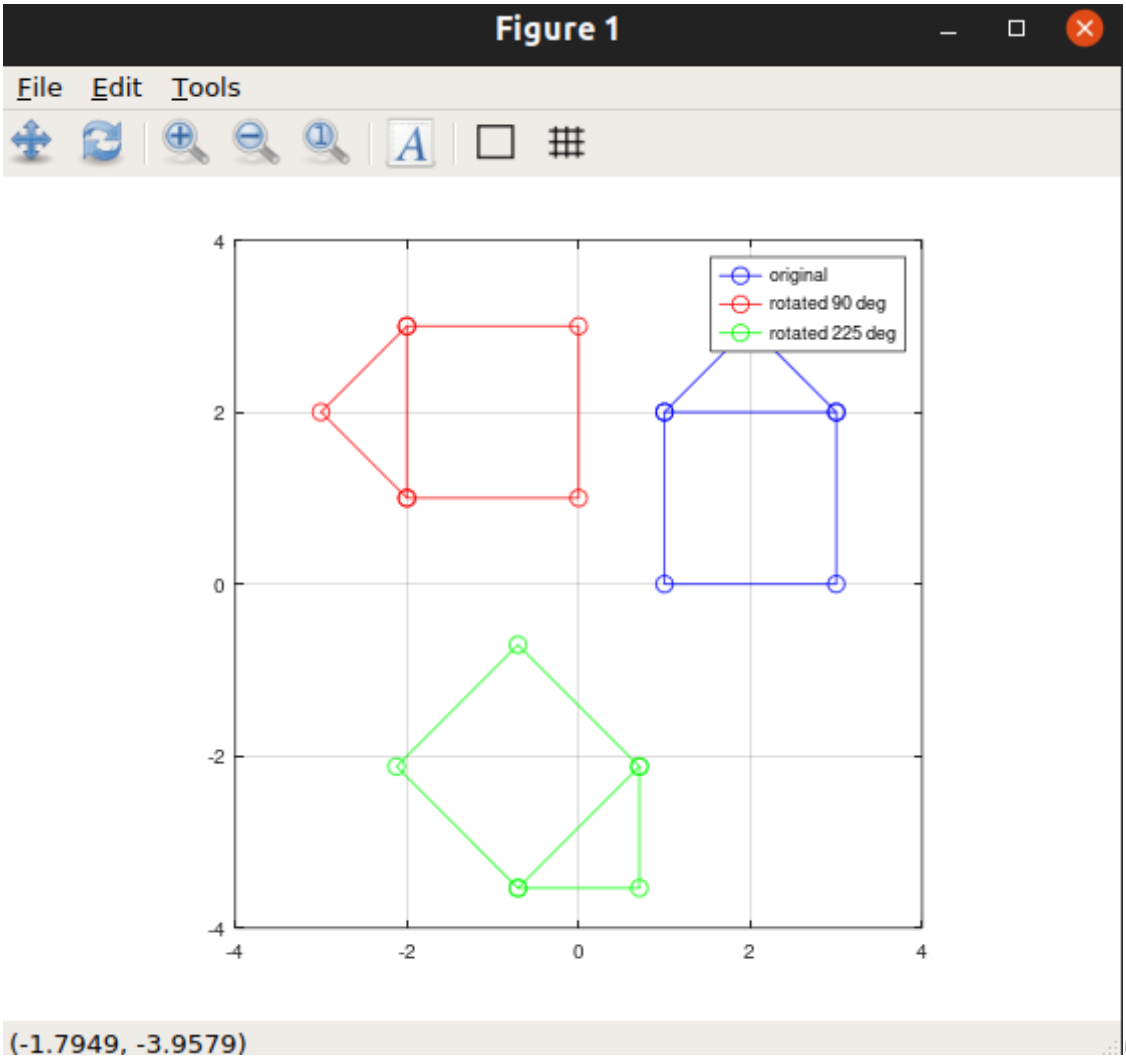
```
>> theta1 = 90*pi/180
theta1 = 1.5708
>> R1=[cos(theta1) -sin(theta1); sin(theta1) cos(theta1)]
R1 =

   6.1232e-17  -1.0000e+00
   1.0000e+00   6.1232e-17

>> RD1 = R1*D
RD1 =

  -2.0000e+00   6.1232e-17   1.8370e-16  -2.0000e+00  -3.0000e+00  -2.0000e+00  -2.0000e+00
   1.0000e+00   1.0000e+00   3.0000e+00   3.0000e+00   2.0000e+00   1.0000e+00   3.0000e+00

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

  -2.0000e+00   6.1232e-17   1.8370e-16  -2.0000e+00  -3.0000e+00  -2.0000e+00  -2.0000e+00

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

   1.0000   1.0000   3.0000   3.0000   2.0000   1.0000   3.0000
```

(Рис 10)

```
>> 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')
```

(Рис 11)

В результате получился такой вид графа(Рис 12)



(Рис 12)

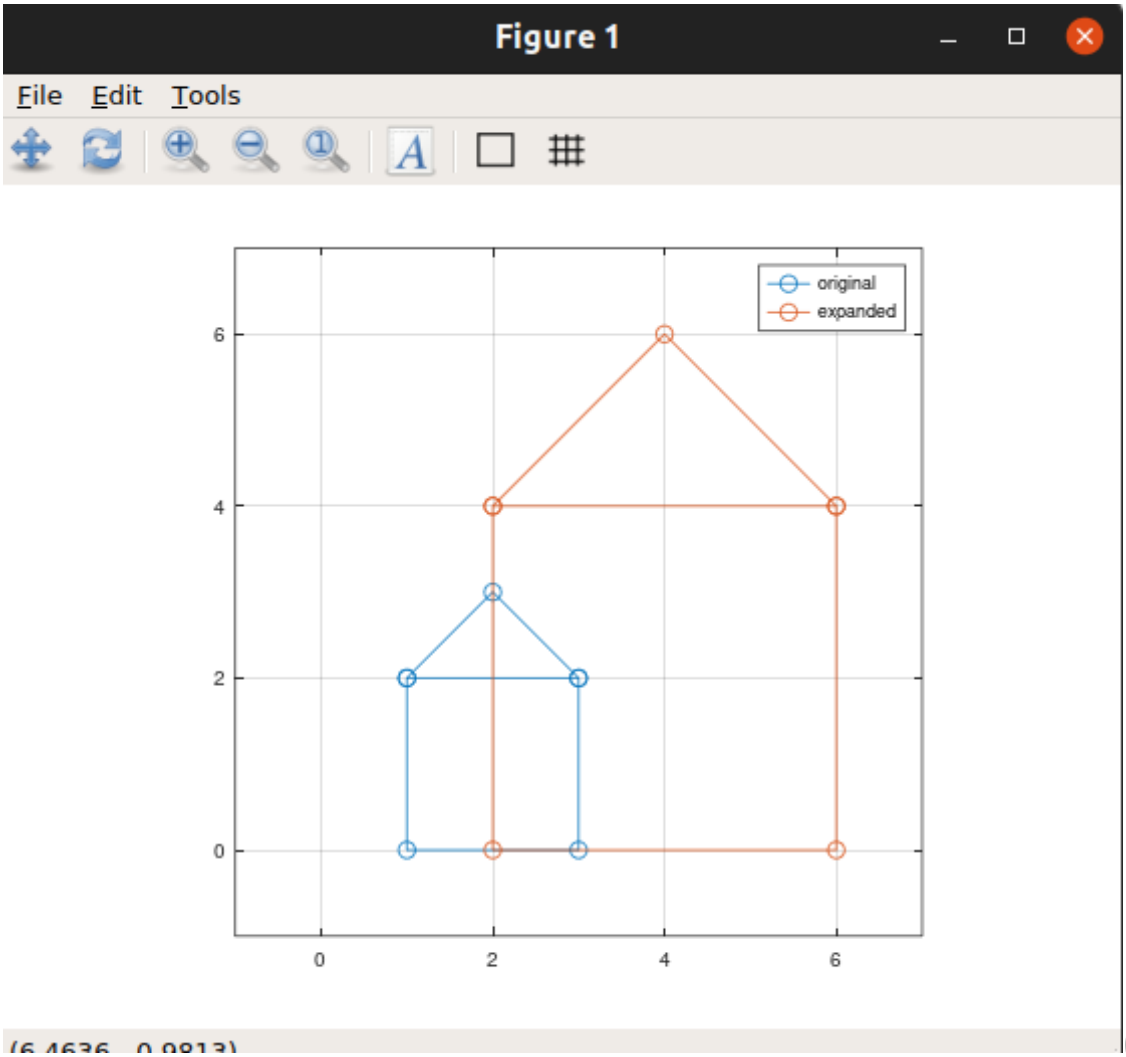
Отражение

Диллатация

1. В этом пункте мы отразим граф дома относительно прямой $y = x$. Зададим матрицу отражения. И сразу увеличиваем размеры дома в два раза

```
>> 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-'
plot(x,y,'o-',x1,y1,'o-')
error: parse error:

  syntax error

>>> plot(x,y,'o-',x1,y1,'o-')
       ^
>> plot (x,y,'o-',x1,y1,'o-')
>> axis ([-1 7 -1 7], 'equal');
>> grid on;
>> legend ('original', 'expanded')
>>
```

(Рис 13)



(Рис 13)

Такой граф получился, потому что, точки x и y соприкасаются друг с другом.

Вывод

Научился подгонять полиномиальные кривые