Научное програмирование

Отчет по лабораторной работе № 5

Кейела Патачона НПМмд-02-21

Содержание

List of Figures

1 Цель работы

Решение проблемы подгонки полинома к множеству точек и изучение матричные преобразования.

2 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.

```
>> 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-')
>> print 01.png -dpng
```

Figure 2.1: Ввод данных

Нарисуем точки на графике.

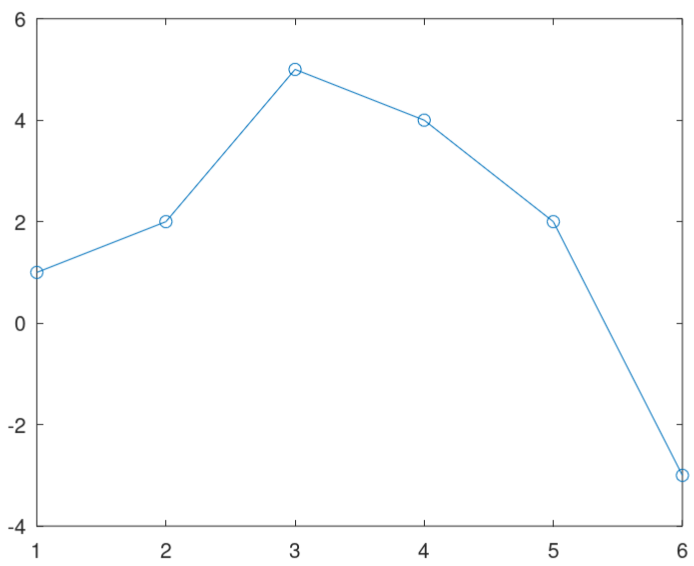

Figure 2.2: график исходных данных

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

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

Обращаем 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.

```
>> 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 2.3: Заполнение матрицы коэффициентов

Решение по методу наименьших квадратов получается из решения уравнения $A^T A b = A^T y$, где b – вектор коэффициентов полинома. Используем Octave для построения уравнений. Запишем расширенную матрицу для решения задачи методом Гаусса.

```
>> A'*A
ans =

   2275    441     91
    441     91     21
     91     21      6

>> A'*ydata
ans =

   60
   28
   11

>> B = A'*A
B =

   2275    441     91
    441     91     21
     91     21      6

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

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

Figure 2.4: Уравнения

Таким образом, искомое квадратное уравнение имеет вид

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

. Построим соответствующий график параболы.

```
>> 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)
```

Figure 2.5: Решение

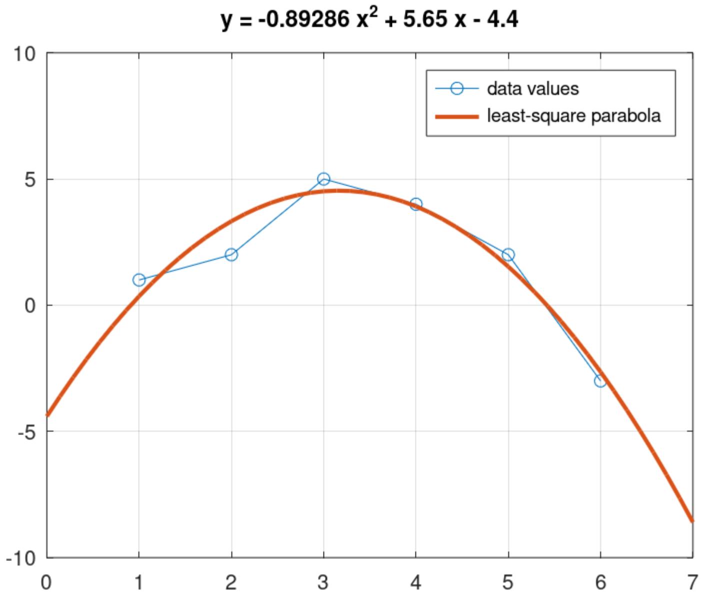

Figure 2.6: Результат подгонки

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.

```
>> y = a1*x.^2 + a2*x + a3;
>> plot(xdata,ydata,'o-',x,y,'linewidth',2)
>> grid on;
>> legend('data values','least-square parabola')
>> title('y = -0.89286 x^2 + 5.65 x - 4.4')
>> print 02.png -dpng
>> 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')
>>
```

Figure 2.7: Процесс построения подгонки

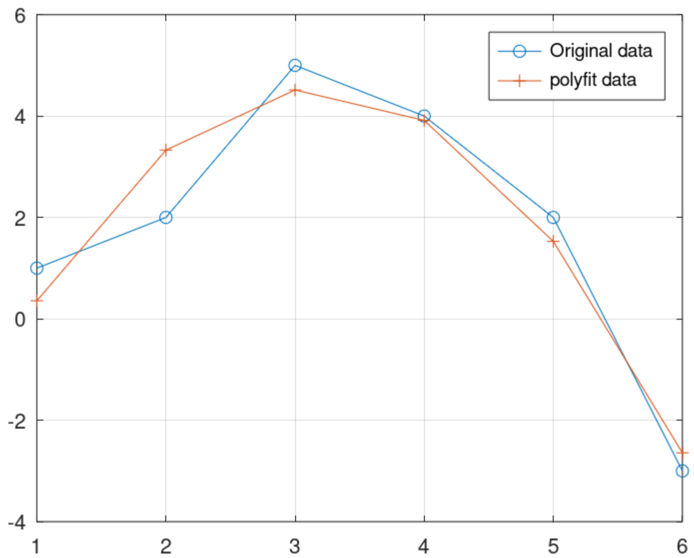

Figure 2.8: Результат подгонки

3 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).

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

```
>> 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
```

Figure 3.1: Кодировка домика

3.1 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

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

```
>> 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 =

  -2.0000e+00   6.1230e-17   1.8369e-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   6.1230e-17   1.8369e-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
```

Figure 3.2: Вращение домика 1

```
>> theta2 = 255*pi/180
theta2 = 4.4506
>> 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 degrees','Rotated 225 degrees')
>> print 03.png -dpng
>>
```

Figure 3.3: Вращение домика 2

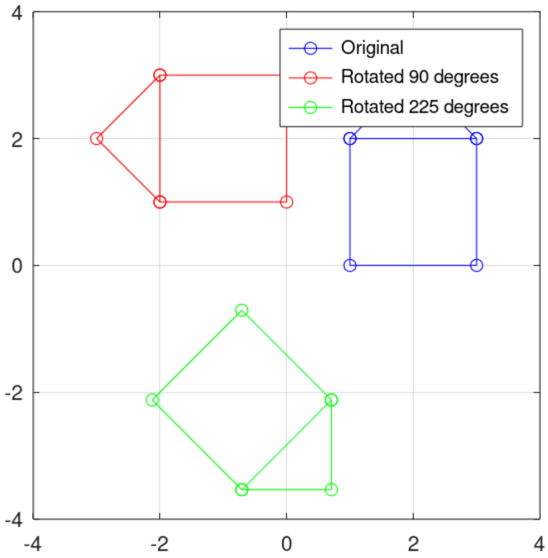

Figure 3.4: Результат вращения

3.2 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 (поясните, почему она такая).

```
>>
>> 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
```

Figure 3.5: Отражение домика 1

```
>> 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')
>> axis([-1 4 -1 4],'equal')
>> grid on
>> legend('Original','Reflected')
>>
```

Figure 3.6: Отражение домика 2

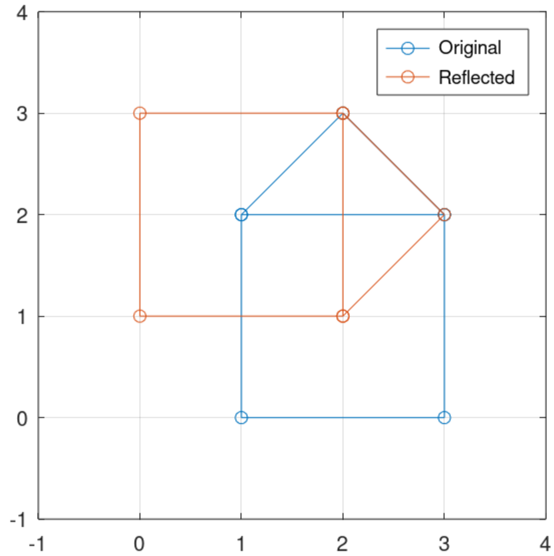

Figure 3.7: Результат отражения

3.3 Дилатация

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

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

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

```
>> 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,:);
>> print 04.png -dpng
>>
>>
>> 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','Expanded')
>> print 05.png -dpng
>> clear;
>> clf;
```

Figure 3.8: Дилатация домика

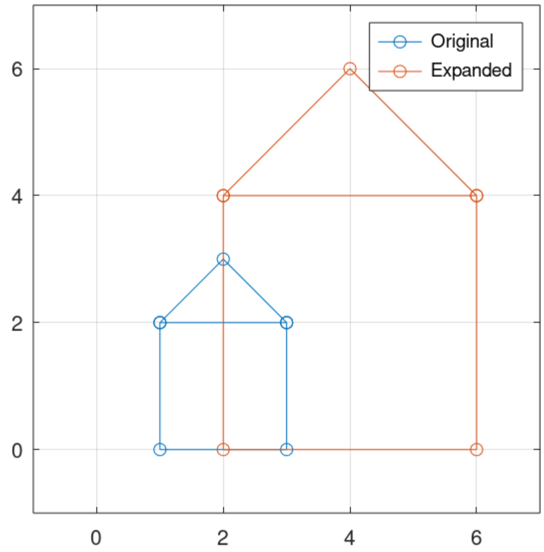

Figure 3.9: Результат дилатации

4 Вывод

В ходе выполнения данной работы я научился решить задачу подгонки и работать с матричными преобразованиями.