3. АППРОКСИМАЦИЯ

Аппроксимация – это замена одной функции $f(x)$ другой, похожей функцией $Y(x)$. Методы аппроксимации применяют как в случае, когда функция $f(x)$ задана в табличном виде, так и тогда, когда функция $f(x)$ является непрерывной и есть необходимость получить упрощенное математическое описание имеющейся сложной зависимости. На практике задача аппроксимации часто возникает тогда, когда по экспериментальным данным требуется подобрать такую аналитическую функцию, которая проходила бы как можно ближе к экспериментальным точкам. Построенная в результате решения задачи аппроксимации кривая сглаживает обрабатываемые экспериментальные данные.

Пусть в результате эксперимента получены данные, представленные в виде табл. 3.1.

Таблица 3.1

Экспериментальные данные

x_i	x_0	x_1	x_2	x_3	…	x_m
y_i	y_0	y_1	y_2	y_3	…	y_m

Необходимо заменить таблично заданную функцию аналитической функцией $Y=f(x,c_0,c_1,\ldots,c_k)$. При этом искомая функция $f(x,c_0,c_1,\ldots,c_k)$ может зависеть от параметров c_j $(j=\overline{0,k})$ как линейно, и тогда говорят о линейной аппроксимации, так и не линейно. В последнем случае говорят о нелинейной аппроксимации. При решении задачи надо таким образом подобрать коэффициенты $c_0, c_1, \ldots, c_k$ функции Y, чтобы отклонения экспериментальных значений y_i от модельных $Y_i = f(x_i,c_0,c_1,\ldots,c_k)$ были в каком-то смысле минимальными. Решение задачи аппроксимации состоит из нескольких этапов. На первом надо определить вид зависимости, т. е. выбрать такую кривую, которая лучше всего подходит для описания экспериментальных данных. При этом исходят или из аналитических предпосылок или выбирают вид кривой визуально исходя из графика с экспериментальными данными. Если выбрать одну из нескольких кривых затруднительно, можно воспользоваться методом средних точек. Метод средних точек заключается в том, что для каждой из кривых, которые в принципе могут быть использованы для решения задачи аппроксимации,

рассчитывается средняя точка. Все эти точки наносятся на график с экспериментальной кривой. Выбирается та кривая, средняя точка которой находится ближе всего к экспериментальной кривой. После того как вид зависимости выбран, необходимо указать способ получения неизвестных коэффициентов $c_0, c_1, \ldots, c_k$. Число искомых коэффициентов $k+1 \le m$. После их нахождения нужно оценить правильность решения.

Одним из наиболее часто используемых способов аппроксимации кривых для функционально связанных экспериментальных данных является полиномиальная аппроксимация. В этом случае аппроксимирующая функция имеет вид $f(x, c_0, c_1, \ldots, c_k) = c_0 + c_1 x + c_2 x^2 + \cdots + c_k x^k$. Среди других часто используемых способов аппроксимации следует отметить логарифмическую, экспоненциальную, степенную аппроксимацию, аппроксимацию тригонометрическими функциями, в том числе рядами Фурье.

3.1. Меры погрешности аппроксимации

Близость исходной $f(x)$ и аппроксимирующей $Y=f(x, c_0, c_1, \ldots, c_k)$ функций определяется некоторой числовой мерой, называемой критерием аппроксимации или критерием близости. Наиболее часто используемыми подходами к решению задачи аппроксимации являются метод наименьших модулей, минимаксный подход и метод наименьших квадратов.

При решении задачи аппроксимации *методом наименьших модулей* в качестве критерия близости выбирают критерий $M = \sum_{i=0}^{m} \left| y_i - f(x_i, c_0, c_1, \ldots, c_k) \right|$, а коэффициенты c_j ищут из условия $M \to \min_{c_j}$. Минимизируются суммы (иногда – взвешенные) абсолютных значений невязок. Невязка – это разность между экспериментальными и модельными значениями функции $y_i - f(x_i, c_0, c_1, \ldots, c_k)$ $(i = \overline{0, m})$. На рис. 3.1 это отрезки прямых синего цвета.

При решении задачи аппроксимации с использованием *минимаксного подхода* (равномерное приближение) в качестве меры

погрешности выбирают $R = \max_i |y_i - f(x_i, c_0, c_1, ..., c_k)|$. Коэффициенты c_j подбирают таким образом, чтобы R было минимальным: $\min_{c_j} \max_i |y_i - f(x_i, c_0, c_1, ..., c_k)|$. Этот подход осуществляет равномерное приближение функций. Геометрически это означает, что из всех кривых заданного вида выбирают ту, у которой максимальная по модулю невязка минимальна. У кривой, изображенной на рис. 3.2, модуль максимальной невязки равен длине отрезка AB.

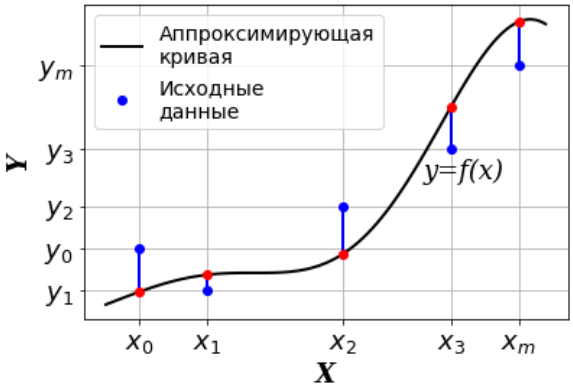

Рис. 3.1. Аппроксимация методом наименьших модулей

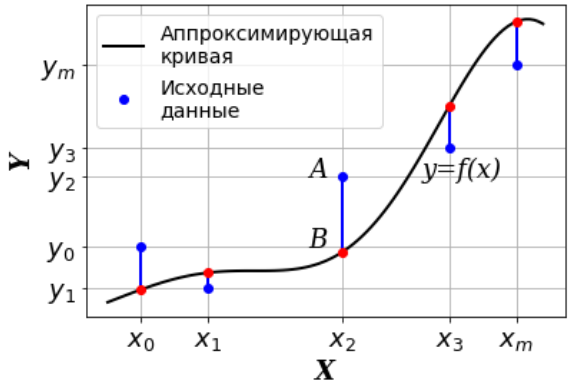

Рис. 3.2. Минимаксный подход к решению задачи аппроксимации

Чаще всего при точечной аппроксимации используют меру $K = \sum_{i=0}^{m} [y_i - f(x_i, c_0, c_1, ..., c_k)]^2$, а коэффициенты c_j $(j = \overline{0,k})$ ищут из условия $K \to \min_{c_j}$. Квадратичный критерий дифференцируем и обеспечивает единственное решение задачи аппроксимации для полиномиальных аппроксимирующих функций. Описанный подход к задаче аппроксимации называется *методом наименьших квадратов*. Геометрически это означает, что из всех кривых заданного вида (полиномов заданной степени, кривых вида $y = ae^{bx}$ и т. д.) выбирают ту, у которой сумма площадей квадратов отклонений наименьшая (рис. 3.3).

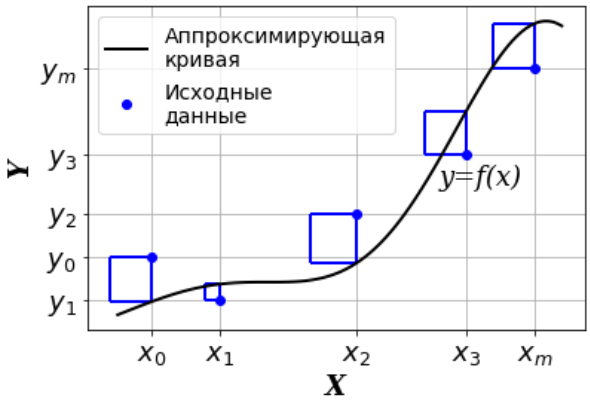

Рис. 3.3. Решение задачи аппроксимации методом наименьших квадратов

Неизвестные коэффициенты c_j $(j = \overline{0,k})$ можно найти, приравняв к нулю частные производные по этим коэффициентам: $\frac{\partial K}{\partial c_j} = 0,\ j = \overline{0,k}.$

В данном пособии остановимся лишь на решении задачи аппроксимации методом наименьших квадратов. При этом всюду будем предполагать, что вид аппроксимирующей кривой нам известен.

Решение задачи аппроксимации с использованием других подходов можно найти в учебниках по численным методам[1].

[1] Самарский А. А., Гулин А. В. Численные методы. Москва: Наука, 1989. 432 с.; Мудров В. И., Кушко В. Л. Метод наименьших модулей. Москва: Знание, 1971. 64 с.

3.2. Метод наименьших квадратов для решения задачи аппроксимации. Функции linalg.lstsq(), curve_fit() и optimize.leastsq()

Сначала рассмотрим решение задачи точечной аппроксимации методом наименьших квадратов в случае, когда апппроксимирующая кривая является алгебраическим полиномом. Это, как правило, самый простой способ аппроксимации, когда решение можно найти и без применения специальных функций. Покажем, как это можно сделать, на примере.

Пример 3.1. Методом наименьших квадратов построить аппроксимирующую параболу $y^m=c_0+c_1x+c_2x^2$ для функции, заданной в табличном виде:

x	0	1	2	3
y	−1,2	−0,1	1,3	1,8

Найти сумму площадей квадратов отклонений. Решить задачу с помощью функции lstsq() модуля scipy.linalg. Дать графическое решение задачи.

Решение. Поскольку задано 4 узла таблицы, можно заменять такую таблично заданную функцию полиномами до 3-й степени. В соответствии с условием задачи нужно подобрать такой полином 2-й степени $y^m=c_0+c_1x+c_2x^2$, чтобы $K=\sum_{i=0}^{m}\left(y_i-c_0-c_1x_i-c_2x_i^2\right)^2\to\min\limits_{c_0,c_1,c_2}$, где x_i, y_i – заданные точки таблицы. Неизвестные коэффициенты найдем из решения системы нормальных уравнений, получить которую можно, приравняв к нулю частные производные от K по неизвестным коэффициентам:

$$\begin{cases}\dfrac{\partial K}{\partial c_0}=-2\sum\limits_{i=0}^{3}\left(y_i-c_0-c_1x_i-c_2x_i^2\right)=0\\ \dfrac{\partial K}{\partial c_1}=-2\sum\limits_{i=0}^{3}\left(y_i-c_0-c_1x_i-c_2x_i^2\right)x_i=0\\ \dfrac{\partial K}{\partial c_2}=-2\sum\limits_{i=0}^{3}\left(y_i-c_0-c_1x_i-c_2x_i^2\right)x_i^2=0\end{cases}.$$

После преобразований получаем

$$\begin{cases} \sum_{i=0}^{3} y_i = (3+1)c_0 + c_1 \sum_{i=0}^{3} x_i + c_2 \sum_{i=0}^{3} x_i^2 \\ \sum_{i=0}^{3} x_i y_i = c_0 \sum_{i=0}^{3} x_i + c_1 \sum_{i=0}^{3} x_i^2 + c_2 \sum_{i=0}^{3} x_i^3 \\ \sum_{i=0}^{3} x_i^2 y_i = c_0 \sum_{i=0}^{3} x_i^2 + c_1 \sum_{i=0}^{3} x_i^3 + c_2 \sum_{i=0}^{3} x_i^4 \end{cases}.$$

Для нахождения сумм составим таблицу

x	y	xy	x^2y	x^2	x^3	x^4
0	-1,2	0	0	0	0	0
1	-0,1	-0,1	-0,1	1	1	1
2	1,3	2,6	5,2	4	8	16
3	1,8	5,4	16,2	9	27	81
6	**1,8**	**7,9**	**21,3**	**14**	**36**	**98**

Подставив полученные значения в систему уравнений, получаем

$$\begin{cases} 1{,}8 = 4c_0 + 6c_1 + 14c_2 \\ 7{,}9 = 6c_0 + 14c_1 + 36c_2 \\ 21{,}3 = 14c_0 + 36c_1 + 98c_2 \end{cases}.$$

Решаем эту систему уравнений, например, с помощью функции solve() модуля scipy.linalg:

```
import numpy as np;from scipy.linalg import *
A = np.array([ [4,6,14], [6,14,36], [14,36,98]])
# Преобразуем строку в столбец
b = np.array([1.8,7.9,21.3]).reshape(3,1)
c=solve(A,b)
for i in range(len(b)):
  print("c(%i)=%.7f" % (i,c[i]))
```

Получаем решение:

```
c(0)=-1.2600000
c(1)=1.4900000
c(2)=-0.1500000
```

Таким образом, уравнение аппроксимирующей параболы $y^m = -1.26+1.49x-0.15x^2$. Вычислив значения этой функции при

x = [0, 1, 2, 3], получаем y=[-1.26, 0.08, 1.12, 1.86]. Сумма квадратов невязок равна $(-1.26+1.2)^2 + (0.08+0.1)^2 + (1.12-1.3)^2 + (1.86-1.8)^2 = 0.072$. У всех других парабол 2-й степени сумма площадей квадратов отклонений будет больше, чем 0.072.

Решим теперь задачу аппроксимации в модуле scipy.linalg. Используем для этого метод lstsq(). Он позволяет найти такой вектор c, который минимизирует невязку системы линейных алгебраических уравнений (СЛАУ) $Ac=b$ методом наименьших квадратов (МНК).

Составим матрицу A и вектор b следующим образом:

$$A = \begin{pmatrix} 1\ x_0\ x_0^2 \\ 1\ x_1\ x_1^2 \\ 1\ x_2\ x_2^2 \\ \ldots\ldots\ldots \\ 1\ x_m\ x_m^2 \end{pmatrix}, b = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \ldots. \\ y_m \end{pmatrix}.$$

Запишем систему линейных уравнений в матричной форме $Ac=b$, где $c = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix}$ – вектор-столбец неизвестных коэффициентов. Умножение A на c дает $y_i^m = c_0 + c_1 x_i + c_2 x_i^2, (i = \overline{0,m})$ – значения аппроксимирующей параболы в заданных точках таблицы. Правая часть уравнения – табличные значения функции y_i. Разница между ними дает вектор невязок.

Полученную систему решаем с помощью функции lstsq(). Синтаксис функции lstsq():

lstsq(a, b, rcond='warn').

Обязательными входными параметрами являются матрица A указанного выше вида и вектор значений функции y. Результатом решения являются вектор c со значениями искомых коэффициентов, сумма площадей квадратов отклонений, ранг матрицы A и сингулярные числа матрицы A. Сингулярными числами вещественной матрицы A размерности m на n называются арифметические значения квадратных корней из собственных чисел $\lambda_1, \lambda_2, \ldots, \lambda_k$ матрицы A^TA, где k=min(n,m).

Код программы решения задачи:

```
import scipy.linalg
import numpy as np
```

```
from numpy import *
import matplotlib.pyplot as plt
fig = plt.figure();ax = fig.add_subplot(111)
x=np.array([0, 1, 2, 3])
y=np.array([-1.2, -0.1, 1.3, 1.8])
A=x[:,np.newaxis]**[0,1,2]
res=linalg.lstsq(A,y,rcond=None)
print("Значения коэффициентов")
for i in range(3):
  print("c(%i)=%.7f" % (i,res[0][i]))
a, b, c = res[0][0], res[0][1], res[0][2]
print("Уравнение кривой Ym=%.6f+(%.6f)*x+(%.6f)*x^2" %
(a,b,c))
v=res[1]
print("Сумма площадей квадратов отклонений равна  %.6f" % v)
x1 = np.linspace(min(x), max(x), 100);
ym= a+b*x1+c*x1**2
plt.plot(x, y, 'bo', label="Исходные\nданные")
plt.plot(x1,ym,  label="Аппроксимирующая\nкривая",c='g')
plt.legend()
plt.show()
```

Результатами решения являются значения коэффициентов, уравнение аппроксимирующей параболы, сумма площадей квадратов отклонений и график с исходными данными и аппроксимирующей параболой.

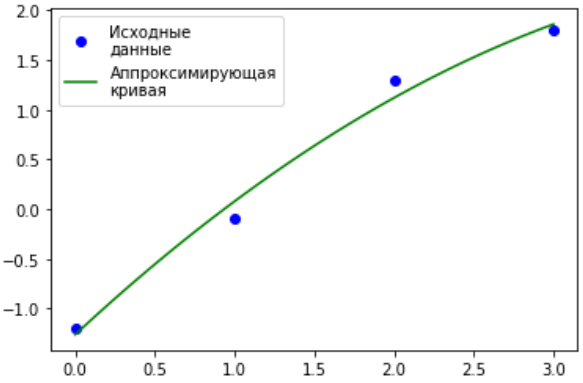

Рис. 3.4. Графическое решение задачи аппроксимации

Значения коэффициентов:

c(0)=-1.2600000
c(1)=1.4900000
c(2)=-0.1500000
Уравнение кривой: Ym=-1.260000+(1.490000)*x+(-0.150000)*x^2.
Сумма площадей квадратов отклонений равна 0.072000.

Комментарий к программе. Подгружаем необходимые библиотеки и модули. Задаем исходные данные (x и y). В коде мы использовали объект библиотеки Numpy newaxis, с помощью которого можно создать новые оси. Применение его дало нам искомую матрицу A:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix}.$$

Решаем задачу аппроксимации с помощью функции linalg.lstsq(), параметрами которой являются созданная матрица A и вектор y. Параметру rcond присваиваем значение None, чтобы не получать ненужных нам в этой задаче сообщений. Результатами решения являются вектор res[0], элементы которого – искомые коэффициенты и res[1], содержащий значение суммы площадей квадратов отклонений. Другие компоненты решения в данной задаче нам не требуются. Далее выводим полученные результаты и строим график с исходными данными и аппроксимирующей параболой.

Таким способом можно искать решение задачи аппроксимации в виде линейной комбинации любых функций, т. е. в случае, когда аппроксимирующая кривая выглядит так: $Y^m(x) = g(x) = c_0 g_0(x) + c_1 g_1(x) + \ldots + c_k g_k(x)$. Соответственно, матрица A будет иметь вид

$$A = \begin{pmatrix} g_0(x_0) & \ldots & g_k(x_0) \\ g_0(x_1) & \ldots & g_k(x_1) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ g_0(x_m) & \ldots & g_k(x_m) \end{pmatrix}.$$

Для поиска коэффициентов выбранной зависимости методом наименьших квадратов в SciPy есть еще одна функция – curve_fit(). Ее можно использовать как в случае, когда аппроксимирующая кривая является алгебраическим полиномом, так и в случае неполиномиальной регрессии.

Синтаксис функции curve_fit():

curve_fit(f,xdata,ydata,p0=None,sigma=None,absolute_sigma = False, check_finite=True, bounds=(-np.inf,np.inf),method=None, jac=None, **kwargs).

Основными входными параметрами функции являются:

f – имя аппроксимирующей функции. Функция должна содержать независимую переменную в качестве первого аргумента. Искомые параметры должны быть остальными параметрами функции *f*;

xdata – независимая переменная, массив или последовательность исходных данных;

ydata – зависимые данные, массив или последовательность исходных данных. Значения функции;

p0 – массив или последовательность. Начальное приближение. Если установлено значение None или параметр опущен, то все начальные значения будут равны 1.

Выходными параметрами функции curve_fit() являются массив *popt*, содержащий оптимальные в смысле метода наименьших квадратов значения искомых коэффициентов выбранной зависимости, и *pcov* – ковариационная матрица.

Решим с помощью данной функции задачу нелинейной аппроксимации.

Пример 3.2. Методом наименьших квадратов определить коэффициенты аппроксимирующей кривой $Y=c_1x*cos(c_2x)+c_3$ для таблично заданной функции:

x	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6
y	5.02	6.08	3.33	−0.93	−0.22	7.83	16.52	15.55	2.67	−11.42	−11.78	5.09	25.25

Построить график полученной кривой, на который необходимо нанести исходные данные. Вычислить сумму площадей квадратных отклонений.

Решение. Код программы:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize
from scipy.optimize import *
x=np.linspace(0,6,13);
y=np.array([5.02, 6.08, 3.33, -0.93, -0.22, 7.83,
```

```
    16.52, 15.55, 2.67, -11.42, -11.78, 5.09, 25.25])
def f(x, a, b, c):
    return  a*x*np.cos(b*x)+c;
args,_  = curve_fit(f, x, y)
a, b, c = args[0], args[1], args[2]
print("Коэффициенты кривой:")
print("a= %.6f" % a)
print("b= %.6f" % b)
print("c= %.6f" % c)
print("Уравнение кривой Y=%.6f*x*cos(%.6fx)+%.6f" % (a,b,c))
v=np.sum((f(x, a, b, c)-y)**2)
print("Сумма площадей квадратов отклонений равна  %.6f" % v)
plt.plot(x, y, 'bo', label="Исходные\nданные")
plt.plot(x, f(x, a, b, c), label="Аппроксимирующая\nкривая")
plt.legend()
plt.show()
```

Выполнив программу, получаем коэффициенты кривой, ее уравнение и график:

Коэффициенты кривой:

a= 1.802327
b= 0.833727
c= 6.899873
Уравнение кривой: Y=1.802327*x*cos(0.833727x)+6.899873.
Сумма площадей квадратов отклонений равна 1148.003767.

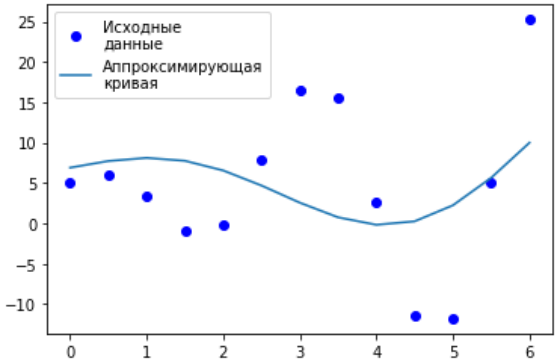

Рис. 3.5. Решение задачи аппроксимации с автоматическим выбором начального приближения

Из рис. 3.5 видно, что полученное решение неудовлетворительное, о чем говорит и сумма площадей квадратов отклонений. Улучшить решение поможет «правильное» задание начального приближения. Таким приближением может быть следующее: a = 1, b = 2, c = 3. Для наглядности увеличим и количество точек, по которым будем строить график аппроксимирующей кривой. Изменения были внесены в следующие операторы:

```
args,_  = curve_fit(f, x, y, [1,2,3])
x1=np.linspace(0,6,100);
plt.plot(x1, f(x1, a, b, c),
      label="Аппроксимирующая\nкривая",c='g')
```

Результат графического решения представлен на рис. 3.6.

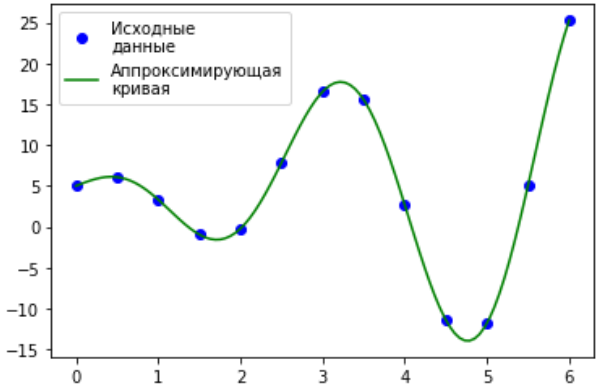

Рис. 3.6. Решение задачи аппроксимации с начальным приближением p0=[1, 2, 3]

Коэффициенты кривой:

a= 4.000196
b= 1.999998
c= 4.999244
Уравнение кривой: Y=4.000196*x*cos(1.999998x)+4.999244.
Сумма площадей квадратов отклонений равна 0.001124.

Анализ результатов решения показывает, что найдено вполне приемлемое решение.

Вывод. Мы вновь убедились, как важно при решении задач численными методами правильно подобрать начальное приближение.

Пример 3.2 можно решить и с помощью функции библиотеки scipy optimize.leastsq(), которая находит минимум квадратов невязок одним из численных методов. Для ее использования нужно сформировать функцию, вычисляющую невязки, и передать ее в качестве первого параметра функции optimize.leastsq(). Второй параметр функции optimize.leastsq() – вектор начальных приближений. В коде это кортеж x0_init.

Программа нахождения необходимых значений и построения соответствующих графиков:

```
from scipy import optimize
import matplotlib.pyplot as plt
import numpy as np
# задаем исходные данные x и y
x=np.linspace(0,6,13);
y=[5.02,6.08,3.33,-0.93,-0.22,7.83,16.52,15.55,
  2.67,-11.42,-11.78,5.09,25.25];
 # задаем аппроксимирующую кривую как функцию
# аргумента x и трех искомых параметров a, b и c
def f(x, a, b, c):
   return a*x*np.cos(b*x)+c;
def g(x0):
   return y- f(x, *x0) # определяем функцию невязок
x0_init = (1,2,3) # задаем начальное приближение
# передаем в функцию leastsq требуемые параметры
x_opt, _ = optimize.leastsq(g, x0_init)
a, b, c =x_opt[0], x_opt[1], x_opt[2]
print("Коэффициенты подобранной зависимости:")
print("a= %.6f" % a)
print("b= %.6f" % b)
print("c= %.6f" % c)
print("Уравнение кривой Y=%.6f*x*cos(%.6fx)+%.6f" % (a,b,c))
v=np.sum((f(x, a, b, c)-y)**2)
print("Сумма площадей квадратов отклонений равна  %.6f" % v)
x1=np.linspace(0,6,40) # задаем пределы значений x
# для построения графика полученной кривой
w=f(x1,a,b,c) # вычисляем
# значения полученной кривой в этих точках и строим график
```

```
plt.plot(x1,w,'r')
plt.plot(x,y,'o')
plt.title("График аппроксимирующей кривой") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # наносим на график сетку
plt.legend(('Аппроксимирующая кривая', 'Исходные данные'))
plt.show()
```

Результат работы программы:
Коэффициенты подобранной зависимости:

a= 4.000196
b= 1.999998
c= 4.999244
Уравнение кривой: Y=4.000196*x*cos(1.999998x)+4.999244.
Сумма площадей квадратов отклонений равна 0.001124.

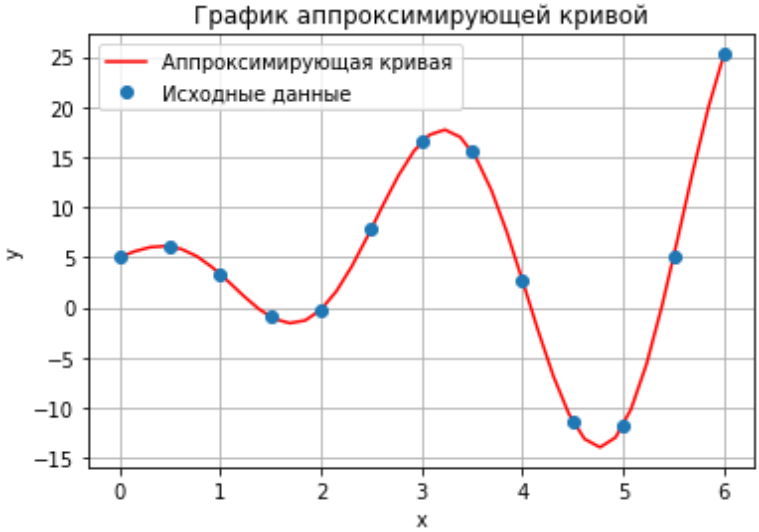

*Рис. 3.7. График аппроксимирующей кривой y=4.0002x·cos(2*x)+4.9992*

Еще раз отметим, что полученное решение существенно зависит от того, насколько хорошо подобрано начальное приближение.