A picture containing logo

Description automatically generated

Лабораторная работа №4

по Вычислительной Математики

Вариант 13

Выполнил:

Пурэвсурэн Билгуун

Группа Р3213

Преподователь:

Малышева Татьяна Алексеевна

г. Санкт-Петербург 2022г

1. **Цель работы:**

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

**Для исследования использовать:**

1. линейную функцию;
2. полиномиальную функцию 2-й степени;
3. полиномиальную функцию 3-й степени;
4. экспоненциальную функцию;
5. логарифмическую функцию;
6. степенную функцию.
7. **Методика проведения исследования:**
8. Вычислить меру отклонения: для всех исследуемых функций.
9. Уточнить значения коэффициентов эмпирических функций, минимизируя функцию S.
10. Сформировать массивы предполагаемых эмпирических зависимостей (.
11. Определить среднеквадратичное отклонение для каждой аппроксимирующей функции. Выбрать наименьшее значение и, следовательно, наилучшее приближение.
12. Построить графики полученных эмпирических функций.
13. **Вычислительная реализация задачи:**

а) Для заданной функции (см. картину 1) построить наилучшие линейное и квадратичное приближения по 11 точкам указанного интервала.

b) Найти среднеквадратические отклонения. Ответы дать с тремя знаками после запятой.

c) Построить графики линейного и квадратичного приближений и заданной функции.

d) ***Привести в отчете подробные вычисления***.

Картина 1



1. **Программная реализация задачи:**
   1. Предусмотреть ввод исходных данных из файла/консоли (таблица *y=f(x)* должна содержать 10 - 12 точек).
   2. Реализовать метод наименьших квадратов, исследуя все функции п.1.
   3. Предусмотреть вывод результатов в файл/консоль.
   4. Для линейной зависимости вычислить коэффициент корреляции Пирсона.
   5. Программа должна отображать наилучшую аппроксимирующую функцию.
   6. Организовать вывод графиков функций, графики должны полностью отображать весь исследуемый интервал (с запасом).
2. **Описание метода, расчетные формулы:**

**Метод наименьших квадратов (МНК) -** Задача заключается в нахождении коэффициентов линейной зависимости, при которых функция двух переменных *а* и *b* формула принимает наименьшее значение. То есть, при данных *а* и *b* сумма квадратов отклонений экспериментальных данных от найденной прямой будет наименьшей. В этом вся суть метода наименьших квадратов.

Эмпирические формулы различных функции:

1. Линейная
2. Квадратичная
3. Экспоненциальная
4. Степенная
5. Логарифмическая
6. **Реализация расчета:**

Вычислительная реализация:

1. Линейная аппроксимация



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| y | 0 | 0.2 | 0.4 | 0.58 | 0.7 | 0.75 | 0.71 | 0.61 | 0.5 | 0.4 | 0.32 |

Эмпирическая функция:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **x** | **y** | **P(x)** | **e** |
| 1 | 0 | 0 | 0.34 | 0.34 |
| 2 | 0.2 | 0.2 | 0.37 | 0.17 |
| 3 | 0.4 | 0.4 | 0.39 | -0.01 |
| 4 | 0.6 | 0.58 | 0.42 | -0.16 |
| 5 | 0.8 | 0.7 | 0.44 | -0.26 |
| 6 | 1 | 0.75 | 0.47 | -0.28 |
| 7 | 1.2 | 0.71 | 0.5 | -0.21 |
| 8 | 1.4 | 0.61 | 0.52 | -0.09 |
| 9 | 1.6 | 0.5 | 0.55 | 0.05 |
| 10 | 1.8 | 0.4 | 0.57 | 0.17 |
| 11 | 2 | 0.32 | 0.6 | 0.27 |

Мера отклонения:

Среднеквадратичное отклонение:

График:

Chart, line chart

Description automatically generated

1. Квадратичная аппроксимация

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| y | 0 | 0.2 | 0.4 | 0.58 | 0.7 | 0.75 | 0.71 | 0.61 | 0.5 | 0.4 | 0.32 |

Эмпирическая функция:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | x | y | P(x) | e |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0.2 | 0.2 | 0.23 | 0.03 |
| 3 | 0.4 | 0.4 | 0.42 | 0.02 |
| 4 | 0.6 | 0.58 | 0.56 | -0.02 |
| 5 | 0.8 | 0.7 | 0.65 | -0.05 |
| 6 | 1 | 0.75 | 0.7 | -0.05 |
| 7 | 1.2 | 0.71 | 0.7 | -0.01 |
| 8 | 1.4 | 0.61 | 0.66 | 0.05 |
| 9 | 1.6 | 0.5 | 0.57 | 0.07 |
| 10 | 1.8 | 0.4 | 0.43 | 0.03 |
| 11 | 2 | 0.32 | 0.25 | -0.07 |

Мера отклонения:

Среднеквадратичное отклонение:

График:

Chart, line chart

Description automatically generated

График с обеим аппроксимацией:

Chart, line chart

Description automatically generated

Программная реализация:

def lin\_func(dots):

""" Линейная аппроксимация """

data = {}

n = len(dots)

x = [dot[0] for dot in dots]

y = [dot[1] for dot in dots]

x\_avg = np.average(x)

y\_avg = np.average(y)

upper = 0

lower = 0

for i in range(0,n):

upper += (x[i]-x\_avg)\*(y[i]-y\_avg)

lowerx = (x[i]-x\_avg)\*\*2

lowery = (y[i]-y\_avg)\*\*2

lower += lowerx + lowery

lower = np.sqrt(lower)

r = upper/lower

print('\nКоэффициент коррелиации Пирсона:', r)

sx = sum(x)

sx2 = sum([xi \*\* 2 for xi in x])

sy = sum(y)

sxy = sum([x[i] \* y[i] for i in range(n)])

d = solve\_det([[sx2, sx],

[sx, n]])

d1 = solve\_det([[sxy, sx],

[sy, n]])

d2 = solve\_det([[sx2, sxy],

[sx, sy]])

try:

a = d1 / d

b = d2 / d

except ZeroDivisionError:

return None

data['a'] = a

data['b'] = b

f = lambda z: a \* z + b

data['f'] = f

data['str\_f'] = "fi = a\*x + b"

data['s'] = calc\_s(dots, f)

data['stdev'] = calc\_stdev(dots, f)

return data

def sqrt\_func(dots):

""" Квадратичная аппроксимация """

data = {}

n = len(dots)

x = [dot[0] for dot in dots]

y = [dot[1] for dot in dots]

sx = sum(x)

sx2 = sum([xi \*\* 2 for xi in x])

sx3 = sum([xi \*\* 3 for xi in x])

sx4 = sum([xi \*\* 4 for xi in x])

sy = sum(y)

sxy = sum([x[i] \* y[i] for i in range(n)])

sx2y = sum([(x[i] \*\* 2) \* y[i] for i in range(n)])

d = solve\_det([[n, sx, sx2],

[sx, sx2, sx3],

[sx2, sx3, sx4]])

d1 = solve\_det([[sy, sx, sx2],

[sxy, sx2, sx3],

[sx2y, sx3, sx4]])

d2 = solve\_det([[n, sy, sx2],

[sx, sxy, sx3],

[sx2, sx2y, sx4]])

d3 = solve\_det([[n, sx, sy],

[sx, sx2, sxy],

[sx2, sx3, sx2y]])

try:

c = d1 / d

b = d2 / d

a = d3 / d

except ZeroDivisionError:

return None

data['c'] = c

data['b'] = b

data['a'] = a

f = lambda z: a \* (z \*\* 2) + b \* z + c

data['f'] = f

data['str\_f'] = "fi = a\*x^2 + b\*x + c"

data['s'] = calc\_s(dots, f)

data['stdev'] = calc\_stdev(dots, f)

return data

def exp\_func(dots):

""" Экспоненциальная аппроксимация """

data = {}

n = len(dots)

x = [dot[0] for dot in dots]

y = []

for dot in dots:

if dot[1] <= 0:

return None

y.append(dot[1])

lin\_y = [log(y[i]) for i in range(n)]

lin\_result = lin\_func([(x[i], lin\_y[i]) for i in range(n)])

a = exp(lin\_result['b'])

b = lin\_result['a']

data['a'] = a

data['b'] = b

f = lambda z: a \* exp(b \* z)

data['f'] = f

data['str\_f'] = "fi = a\*e^(b\*x)"

data['s'] = calc\_s(dots, f)

data['stdev'] = calc\_stdev(dots, f)

return data

def log\_func(dots):

""" Логарифмическая аппроксимация """

data = {}

n = len(dots)

x = []

for dot in dots:

if dot[0] <= 0:

return None

x.append(dot[0])

y = [dot[1] for dot in dots]

lin\_x = [log(x[i]) for i in range(n)]

lin\_result = lin\_func([(lin\_x[i], y[i]) for i in range(n)])

a = lin\_result['a']

b = lin\_result['b']

data['a'] = a

data['b'] = b

f = lambda z: a \* log(z) + b

data['f'] = f

data['str\_f'] = "fi = a\*ln(x) + b"

data['s'] = calc\_s(dots, f)

data['stdev'] = calc\_stdev(dots, f)

return data

def pow\_func(dots):

""" Степенная аппроксимация """

data = {}

n = len(dots)

x = []

for dot in dots:

if dot[0] <= 0:

return None

x.append(dot[0])

y = []

for dot in dots:

if dot[1] <= 0:

return None

y.append(dot[1])

lin\_x = [log(x[i]) for i in range(n)]

lin\_y = [log(y[i]) for i in range(n)]

lin\_result = lin\_func([(lin\_x[i], lin\_y[i]) for i in range(n)])

a = exp(lin\_result['b'])

b = lin\_result['a']

data['a'] = a

data['b'] = b

f = lambda z: a \* (z \*\* b)

data['f'] = f

data['str\_f'] = "fi = a\*x^b"

data['s'] = calc\_s(dots, f)

data['stdev'] = calc\_stdev(dots, f)

return data

1. **Результат программы**

Graphical user interface

Description automatically generated

1. **Вывод**

В результате выполнения данной лабораторной работой я познакомился с аппроксимациями функции методом наименьших квадратов и реализовал их на языке программирования Python, закрепив знания.