РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

Дисциплина: Методы машинного обучения

Студент: Кузнецов Юрий Владимирович

Группа: НФИбд 01-20

Москва 2023

Вариант №16

1. Функция одной переменной $f(x) = (3x - x^3)sin(x)$ на отрезке [0,2]
2. Порядок производной функции одной переменной 4
3. Функция двух переменных $f(x, y) = xln(x + y)$ в области [1,5] X [1,5]
4. Порядок смешанной производной функции двух переменных $\delta^3/(\delta z^2 \delta y)$
5. Показатель качнства регрессии: максимальная ошибка (MaxErr)

Задание

В соответствии с индивидуальным заданием, указанным в записной книжке команды, выполните следующие работы:

1. Постройте тензор ранга 1 (вектор) со значениями заданной в индивидуальном задании функции одной переменной на заданном в индивидуальном задании отрезке и определите максимальное и минимальное значения функции.
2. Постройте график функции с прямыми, соответствующими максимальному и

минимальному значениям, подписывая оси и рисунок и создавая легенду.

3. Найдите значения производной от функции порядка, указанного в индивидуальном задании, и постройте график полученной функции, подписывая оси и рисунок.
4. Постройте тензор ранга 2 (матрицу) со значениями заданной в индивидуальном задании функции двух переменных на заданном в индивидуальном задании прямоугольнике и определите максимальное и минимальное значения функции.
5. Постройте 3d график поверхности функции двух переменных, подписывая оси и рисунок.
6. Найдите значения смешанной производной от функции порядка, указанного в индивидуальном задании, и постройте 3d график поверхности полученной функции, подписывая оси и рисунок.
7. Решите задачу парной линейной регрессии при помощи модели TensorFlow, рассматривая тензор ранга 1 из пункта 1 задания как значения зависимой переменной (отклика), а точки отрезка из индивидуального задания как значения независимой переменной (предиктора). Оцените качество полученной модели по показателю качества регрессии, указанному в индивидуальном задании. Количество эпох, скорость обучения и начальные значения весов выберите самостоятельно, обеспечивая сходимость итерационной процедуры.
8. Постройте кривую обучения для показателя качества регрессии, указанного в индивидуальном задании, с зависимостью от количества эпох. Показатель качества регрессия реализуйте как функцию с использованием функций модуля tf.math.
9. Изобразите на графике точки набора данных (независимой и зависимой переменных) и линию построенной парной регрессии, подписывая оси и рисунок и создавая легенду.

Решение:

Построим тензор ранга 1 (вектор) со значениями заданной в индивидуальном задании функции одной переменной на заданном в индивидуальном задании отрезке и определим максимальное и минимальное значения функции:

```
import numpy as np
import tensorflow as tf


def f(x):
    return (3 * x - x**3) * np.sin(x)

start = 0
end = 2
n = 32

x = tf.constant(np.linspace(start, end, n))
y = f(x)

for i in range(n):
    print("f({:.2f}) = {:.3f}".format(x[i], y[i]))

print("Тензор:", y)
```

```
f(0.00) = 0.000
f(0.06) = 0.012
f(0.13) = 0.050
f(0.19) = 0.110
f(0.26) = 0.193
f(0.32) = 0.296
f(0.39) = 0.416
f(0.45) = 0.551
f(0.52) = 0.696
f(0.58) = 0.848
f(0.65) = 1.002
f(0.71) = 1.154
f(0.77) = 1.299
f(0.84) = 1.433
f(0.90) = 1.549
f(0.97) = 1.645
f(1.03) = 1.714
f(1.10) = 1.754
f(1.16) = 1.759
f(1.23) = 1.727
f(1.29) = 1.655
f(1.35) = 1.541
f(1.42) = 1.383
f(1.48) = 1.180
f(1.55) = 0.933
f(1.61) = 0.642
f(1.68) = 0.311
f(1.74) = -0.059
f(1.81) = -0.462
f(1.87) = -0.895
f(1.94) = -1.349
f(2.00) = -1.819
Тензор: tf.Tensor(
[ 0.          0.01246102  0.04953305  0.11028801  0.19319552  0.29614755
  0.41649231  0.55107737  0.69630129  0.8481733   1.0023804   1.15436104
  1.29938461  1.4326359   1.54930343  1.64467081  1.71420988  1.75367476
  1.75919541  1.72736985  1.65535365  1.54094571  1.38266908  1.17984587
  0.93266497  0.64224182  0.31066904 -0.05894288 -0.46243558 -0.8945844
 -1.34909202 -1.81859485], shape=(32,), dtype=float64)
```

```
max = np.max(y)
min = np.min(y)

print("Максимальное значение функции:", max)
print("Минимальное значение функции:", min)
```

```
Максимальное значение функции: 1.7591954077889664
Минимальное значение функции: -1.8185948536513634
```

Построим график функции с прямыми, соответствующими максимальному и минимальному значениям:

```
import matplotlib.pyplot as plt


plt.plot(x, y, color='purple', label='f(x)')

max = np.max(y)
min = np.min(y)
plt.axhline(y=max, color='red', linestyle='--', label='Максимальное значение')
plt.axhline(y=min, color='blue', linestyle='--', label='Минимальное значение')

plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('График функции f(x)')

plt.legend(loc='best')

plt.show()
```

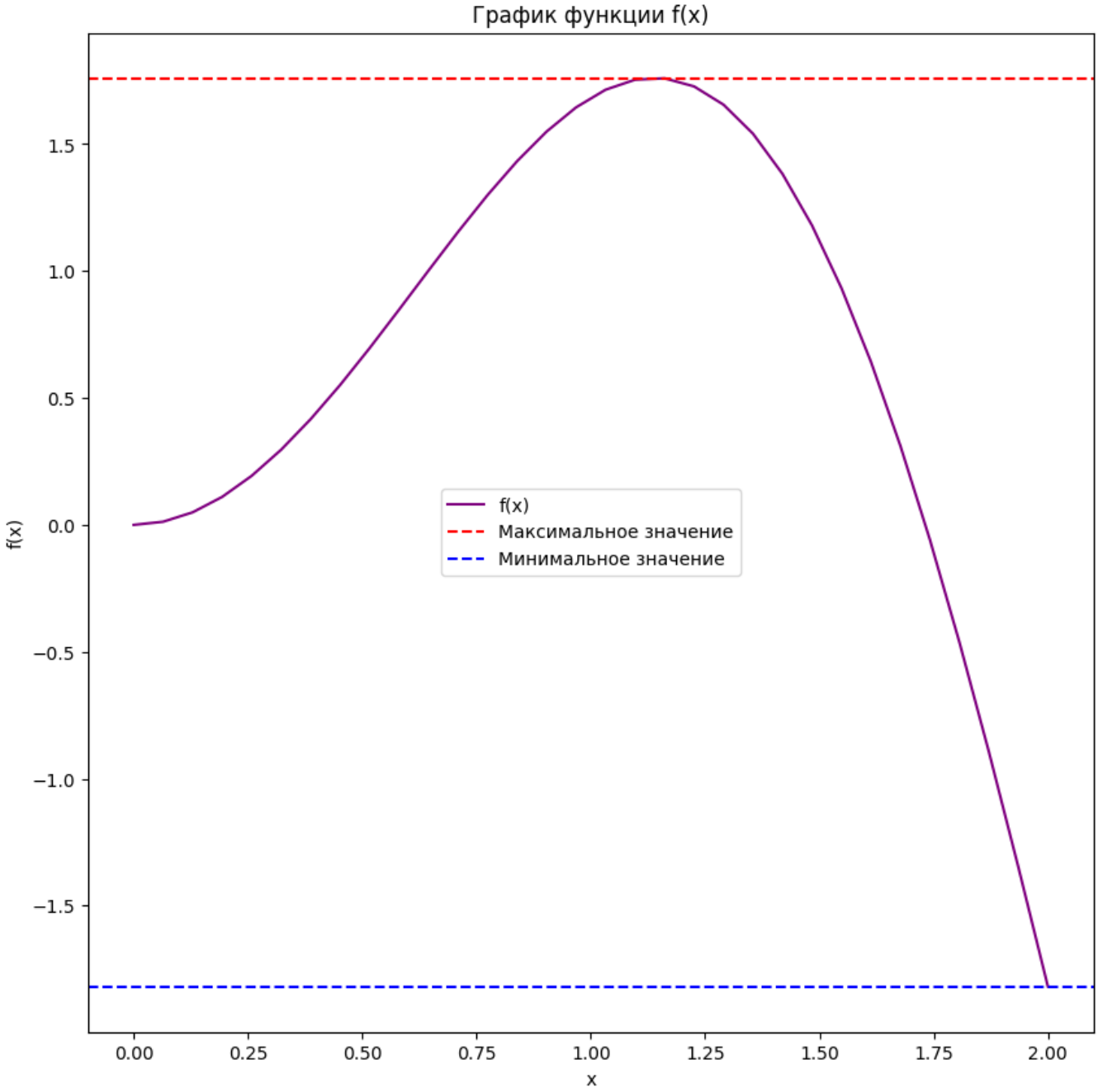


График функции f(x)
f(x)
Максимальное значение
Минимальное значение
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Найдём значения производной от функции порядка, указанного в индивидуальном задании, и построим график полученной функции:

```
def derivative(x, order):
    if order == 0:
        return f(x)
    elif order < 0:
        raise ValueError()
    else:
        derivative_order = f(x)
        for _ in range(order):
            derivative_order = np.gradient(derivative_order, x)
        return derivative_order

order = 4
derivative_values = derivative(x, order)

print(derivative_values)
```

```
[ -17.73774435 -116.29950845 -214.29463072 -122.97938965  -31.12211405
  -28.65045127  -25.68511564  -22.25870191  -18.40905112  -14.17891967
   -9.61560418   -4.77052526    0.30122693    5.54138161   10.88900726
   16.28105846   21.65294532   26.93912131   32.07368485   36.99098997
   41.6262613    45.91620854   49.79963552   53.21803906   56.11619266
   58.44271053   60.15058698   61.19770701   85.30195941   44.33858575
  -59.46954902  -98.56348538]
```

```
plt.plot(x, y, color='purple', label='f^({})(x)'.format(order))

plt.xlabel('x')
plt.ylabel('f^({})(x)'.format(order))
plt.title('График производной порядка {}'.format(order))

plt.legend(loc='best')

plt.show()
```

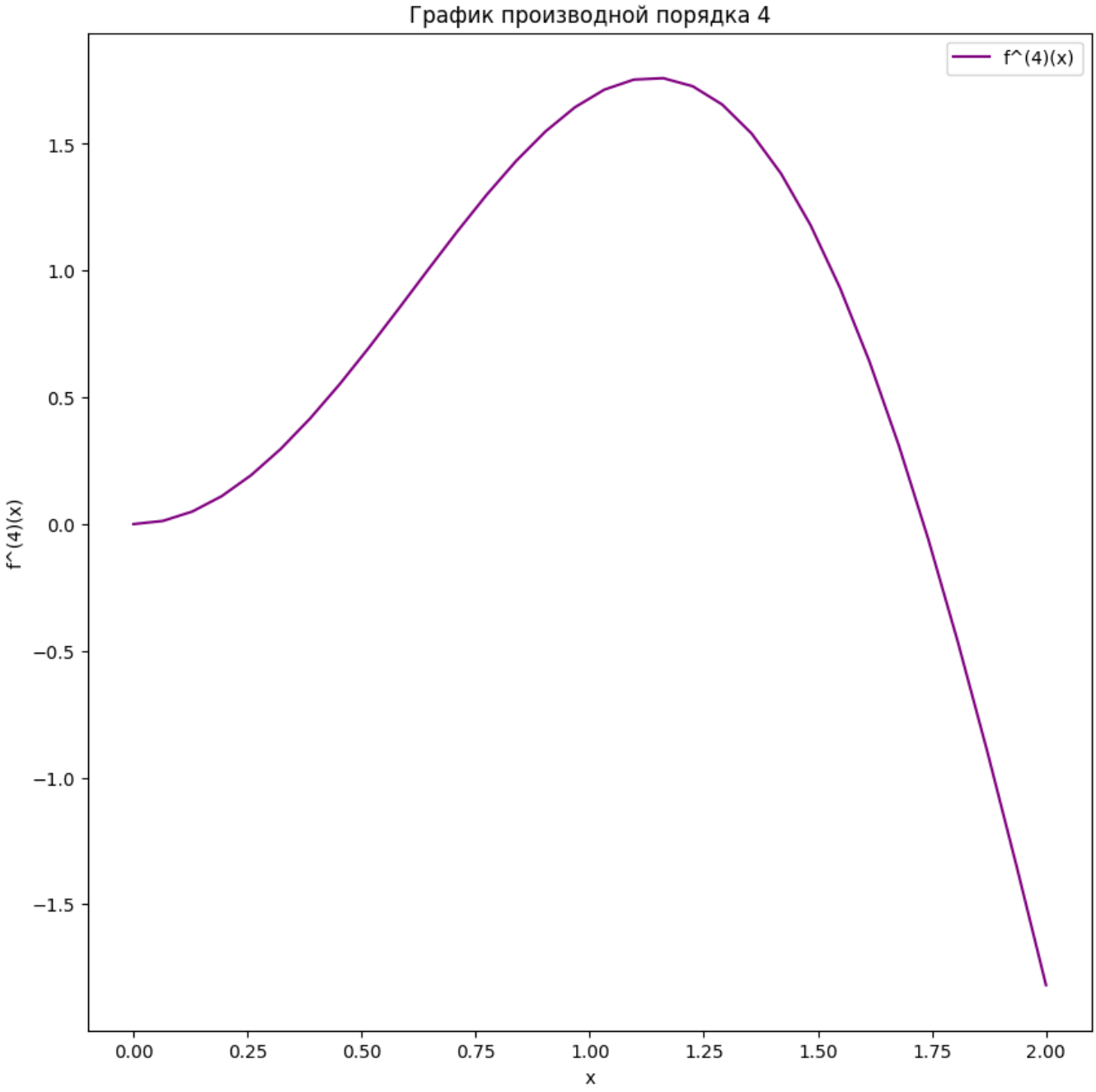


График производной порядка 4
f^(4)(x)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
f^(4)(x)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x

Построим тензор ранга 2 (матрицу) со значениями заданной в индивидуальном задании функции двух переменных на заданном в индивидуальном задании прямоугольнике и определим максимальное и минимальное значения функции:

```
def f(x, y):
    return x * np.log(x + y)

x_start = 1
x_end = 5

y_start = 1
y_end = 5

n_x = 10
n_y = 10

x = np.linspace(x_start, x_end, n_x)
y = np.linspace(y_start, y_end, n_y)

X, Y = np.meshgrid(x, y)

function_values = f(X, Y)

for i in range(n_y):
    for j in range(n_x):
        print("f({:.2f}, {:.2f}) = {:.3f}".format(X[i, j], Y[i, j], function_val

tensor = function_values
```

```
f(1.00, 1.00) = 0.693
f(1.44, 1.00) = 1.291
f(1.89, 1.00) = 2.004
f(2.33, 1.00) = 2.809
f(2.78, 1.00) = 3.692
f(3.22, 1.00) = 4.641
f(3.67, 1.00) = 5.648
f(4.11, 1.00) = 6.707
f(4.56, 1.00) = 7.812
f(5.00, 1.00) = 8.959
f(1.00, 1.44) = 0.894
f(1.44, 1.44) = 1.532
f(1.89, 1.44) = 2.274
f(2.33, 1.44) = 3.101
f(2.78, 1.44) = 4.001
f(3.22, 1.44) = 4.964
f(3.67, 1.44) = 5.982
f(4.11, 1.44) = 7.050
f(4.56, 1.44) = 8.162
```

```
f(5.00, 1.44) = 9.316
f(1.00, 1.89) = 1.061
f(1.44, 1.89) = 1.739
f(1.89, 1.89) = 2.511
f(2.33, 1.89) = 3.361
f(2.78, 1.89) = 4.279
f(3.22, 1.89) = 5.257
f(3.67, 1.89) = 6.288
f(4.11, 1.89) = 7.366
f(4.56, 1.89) = 8.488
f(5.00, 1.89) = 9.650
f(1.00, 2.33) = 1.204
f(1.44, 2.33) = 1.920
f(1.89, 2.33) = 2.721
f(2.33, 2.33) = 3.594
f(2.78, 2.33) = 4.532
f(3.22, 2.33) = 5.525
f(3.67, 2.33) = 6.570
f(4.11, 2.33) = 7.660
f(4.56, 2.33) = 8.792
f(5.00, 2.33) = 9.962
f(1.00, 2.78) = 1.329
f(1.44, 2.78) = 2.081
f(1.89, 2.78) = 2.910
f(2.33, 2.78) = 3.807
f(2.78, 2.78) = 4.763
f(3.22, 2.78) = 5.773
f(3.67, 2.78) = 6.832
f(4.11, 2.78) = 7.934
f(4.56, 2.78) = 9.077
f(5.00, 2.78) = 10.256
f(1.00, 3.22) = 1.440
f(1.44, 3.22) = 2.225
f(1.89, 3.22) = 3.082
f(2.33, 3.22) = 4.001
f(2.78, 3.22) = 4.977
f(3.22, 3.22) = 6.004
f(3.67, 3.22) = 7.076
f(4.11, 3.22) = 8.191
f(4.56, 3.22) = 9.345
f(5.00, 3.22) = 10.534
```

```
max = np.max(tensor)
min = np.min(tensor)

print("Максимальное значение функции:", max)
print("Минимальное значение функции:", min)
```

```
Максимальное значение функции: 11.51292546497023
Минимальное значение функции: 0.6931471805599453
```

Построим 3d график поверхности функции двух переменных:

```
from mpl_toolkits.mplot3d import Axes3D


plt.rcParams["figure.figsize"] = [10, 10]

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, function_values, cmap='viridis')

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('f(x, y)')
ax.set_title('График поверхности функции f(x, y)')

plt.show()
```

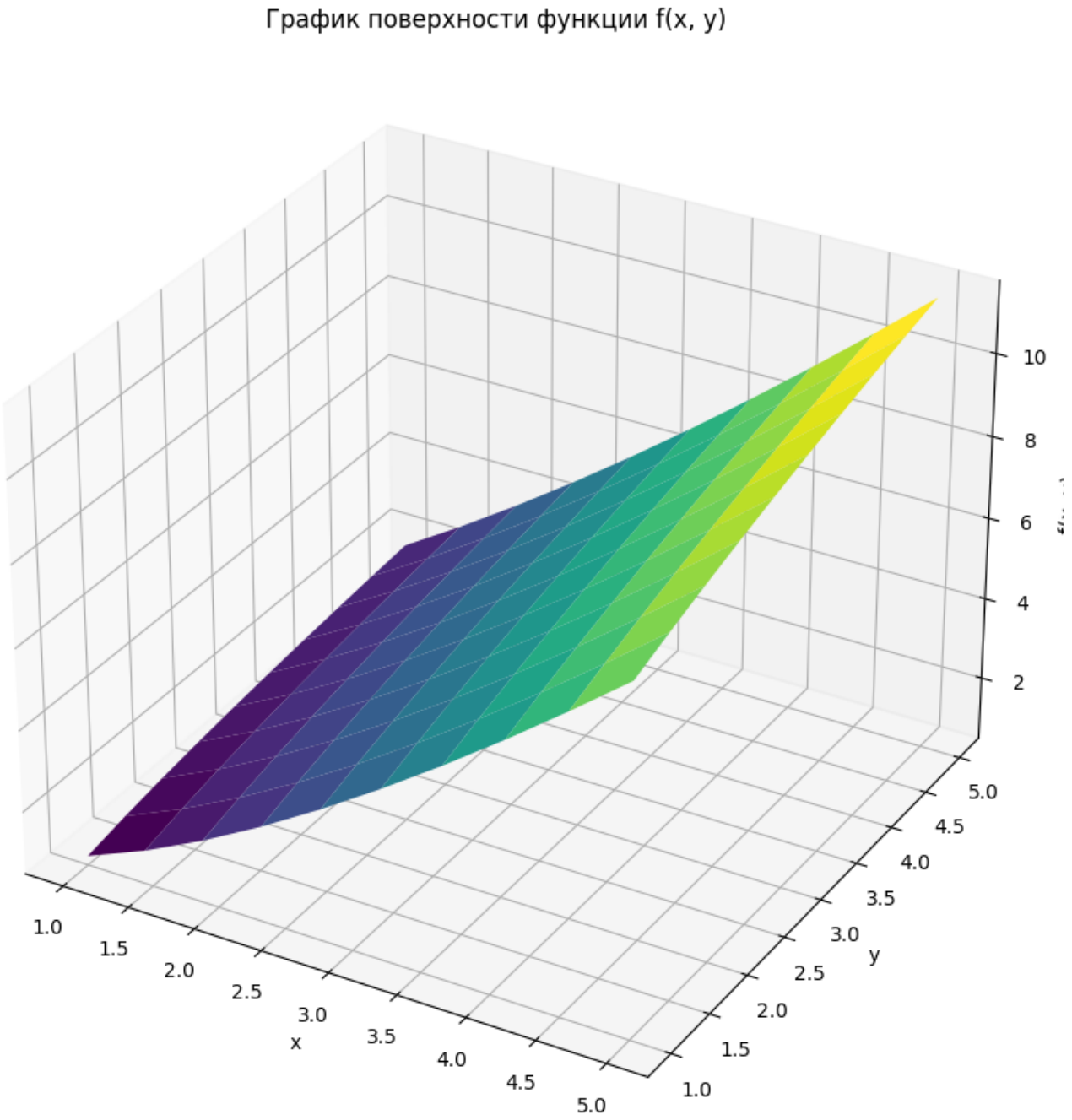


График поверхности функции f(x, y)
x
y
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
2
4
6
8
10

Найдём значения смешанной производной от функции порядка, указанного в индивидуальном задании, и построим 3d график поверхности полученной функции:

```
def f(x, y):
    return x * np.log(x + y)

x_start = 1
x_end = 5

y_start = 1
y_end = 5

n = 100

x = np.linspace(x_start, x_end, n)
y = np.linspace(y_start, y_end, n)

X, Y = np.meshgrid(x, y)

df_dx = np.gradient(f(X, Y), axis=0)
df_dy = np.gradient(f(X, Y), axis=1)
df_dz2_dy = np.gradient(df_dy, axis=1)


plt.rcParams["figure.figsize"] = [10, 10]

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, df_dz2_dy, cmap='viridis')

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('f(x, y)')
ax.set_title('График поверхности смешанной производной')

plt.show()
```

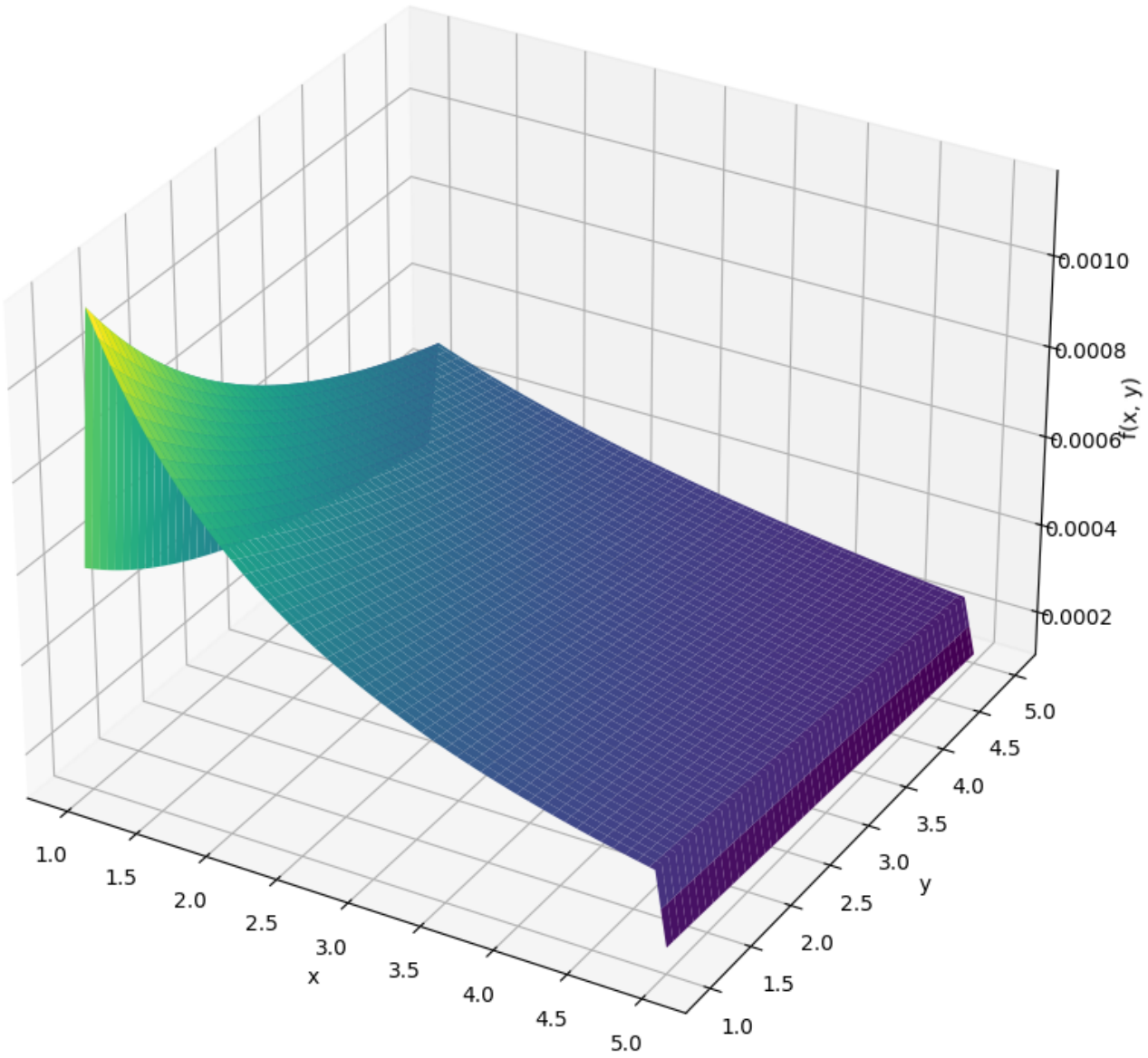


График поверхности смешанной производной
0.0010
0.0008
0.0006
0.0004
0.0002
f(x, y)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
x
y

Решим задачу парной линейной регрессии при помощи модели TensorFlow, рассматривая тензор ранга 1 из пункта 1 задания как значения зависимой переменной (отклика), а точки отрезка из индивидуального задания как значения независимой переменной (предиктора). Оценим качество полученной модели по показателю качества регрессии, указанному в индивидуальном задании:

```
def max_error(y_true, y_pred):
    return tf.reduce_max(tf.abs(y_true - y_pred))

x = np.array([0, 0.5, 1, 1.5, 2])
y = np.array([0, 1.79, 2.99, 3.89, 4.49])

model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1])
])

optimizer = tf.keras.optimizers.Adam(learning_rate=0.1)
loss_fn = tf.keras.losses.MeanSquaredError()

epochs = 1000

max_err_list = []
loss_list = []

for epoch in range(epochs):
    with tf.GradientTape() as tape:
        y_pred = model(x, training=True)
        loss_value = loss_fn(y, y_pred)
    grads = tape.gradient(loss_value, model.trainable_variables)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))

    max_err = max_error(y, y_pred)
    max_err_list.append(max_err.numpy())

    loss_list.append(loss_value.numpy())

y_pred = model(x, training=False)
max_err = np.max(np.abs(y - y_pred))

print("Максимальная ошибка (MaxErr):", max_err)
```

```
Максимальная ошибка (MaxErr): 2.632
```

Построим кривую обучения для показателя качества регрессии, указанного в индивидуальном задании, с зависимостью от количества эпох. Показатель качества регрессия реализуем как функцию с использованием функций модуля tf.math:

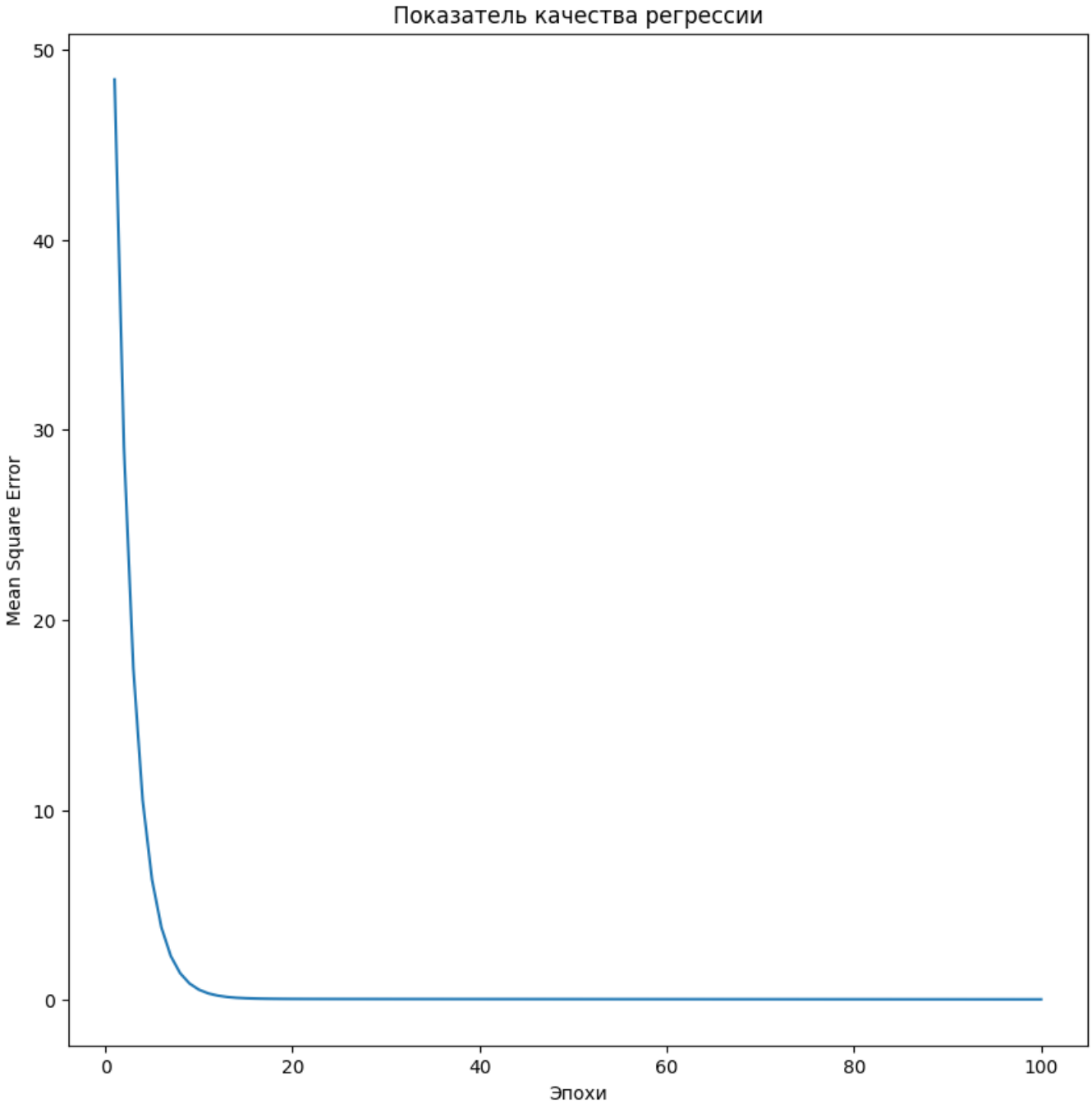
```
x = np.linspace(x_start, x_end, 50)
y = np.linspace(y_start, y_end, 50)

W = tf.Variable(tf.random.normal([1]), name="weight")
b = tf.Variable(tf.random.normal([1]), name="bias")

def linear_regression(x):
    return W * x + b

def mean_square_error(y_pred, y_true):
    return tf.reduce_mean(tf.square(y_pred - y_true))

optimizer = tf.optimizers.SGD(learning_rate=0.01)

def train_step(x, y):
    with tf.GradientTape() as tape:
        y_pred = linear_regression(x)
        loss = mean_square_error(y_pred, y)
    gradients = tape.gradient(loss, [W, b])
    optimizer.apply_gradients(zip(gradients, [W, b]))

epochs = 100
error_values = []

for epoch in range(epochs):
    train_step(x, y)
    y_pred = linear_regression(x)
    error = mean_square_error(y_pred, y)
    error_values.append(error)
    print(f"Epoch: {epoch + 1}, Error: {error.numpy()}")
```

```
Epoch: 1, Error: 48.434730529785156
Epoch: 2, Error: 29.096542358398438
Epoch: 3, Error: 17.490131378173828
Epoch: 4, Error: 10.524133682250977
Epoch: 5, Error: 6.343191623687744
Epoch: 6, Error: 3.8337676525115967
Epoch: 7, Error: 2.3275458812713623
Epoch: 8, Error: 1.4234199523925781
Epoch: 9, Error: 0.8806567192077637
Epoch: 10, Error: 0.5547746419906616
Epoch: 11, Error: 0.3590598702430725
Epoch: 12, Error: 0.2414683848619461
Epoch: 13, Error: 0.17076531052589417
```

```
Epoch: 13, Error: 0.17076331052389417
Epoch: 14, Error: 0.12820403277873993
Epoch: 15, Error: 0.10253334045410156
Epoch: 16, Error: 0.08700051158666611
Epoch: 17, Error: 0.07755279541015625
Epoch: 18, Error: 0.07175778597593307
Epoch: 19, Error: 0.06815572082996368
Epoch: 20, Error: 0.06587032973766327
Epoch: 21, Error: 0.06437583267688751
Epoch: 22, Error: 0.06335661560297012
Epoch: 23, Error: 0.06262321770191193
Epoch: 24, Error: 0.06206196919083595
Epoch: 25, Error: 0.061604686081409454
Epoch: 26, Error: 0.06121036410331726
Epoch: 27, Error: 0.060854386538267136
Epoch: 28, Error: 0.06052204594016075
Epoch: 29, Error: 0.060204461216926575
Epoch: 30, Error: 0.0598963126540184
Epoch: 31, Error: 0.059594426304101944
Epoch: 32, Error: 0.0592968575656414
Epoch: 33, Error: 0.059002455323934555
Epoch: 34, Error: 0.05871051177382469
Epoch: 35, Error: 0.05842062830924988
Epoch: 36, Error: 0.05813254415988922
Epoch: 37, Error: 0.057846084237098694
Epoch: 38, Error: 0.05756118893623352
Epoch: 39, Error: 0.057277776300907135
Epoch: 40, Error: 0.056995753198862076
Epoch: 41, Error: 0.056715212762355804
Epoch: 42, Error: 0.056436073035001755
Epoch: 43, Error: 0.05615827441215515
Epoch: 44, Error: 0.05588187277317047
Epoch: 45, Error: 0.05560685694217682
Epoch: 46, Error: 0.05533311888575554
Epoch: 47, Error: 0.05506080016493797
Epoch: 48, Error: 0.05478981137275696
Epoch: 49, Error: 0.05452011898159981
Epoch: 50, Error: 0.05425180494785309
Epoch: 51, Error: 0.05398477613925934
Epoch: 52, Error: 0.05371907353401184
Epoch: 53, Error: 0.05345468968153
Epoch: 54, Error: 0.05319159850478172
Epoch: 55, Error: 0.052929796278476715
Epoch: 56, Error: 0.05266929790377617
Epoch: 57, Error: 0.052410051226615906
Epoch: 58, Error: 0.052152086049318314
Epoch: 59, Error: 0.051895447075366974
Epoch: 60, Error: 0.05164037744721413
```

```
plt.plot(range(1, epochs + 1), error_values)

plt.xlabel("Эпохи")
plt.ylabel("Mean Square Error")
plt.title("Показатель качества регрессии")
```

```
plt.show()
```

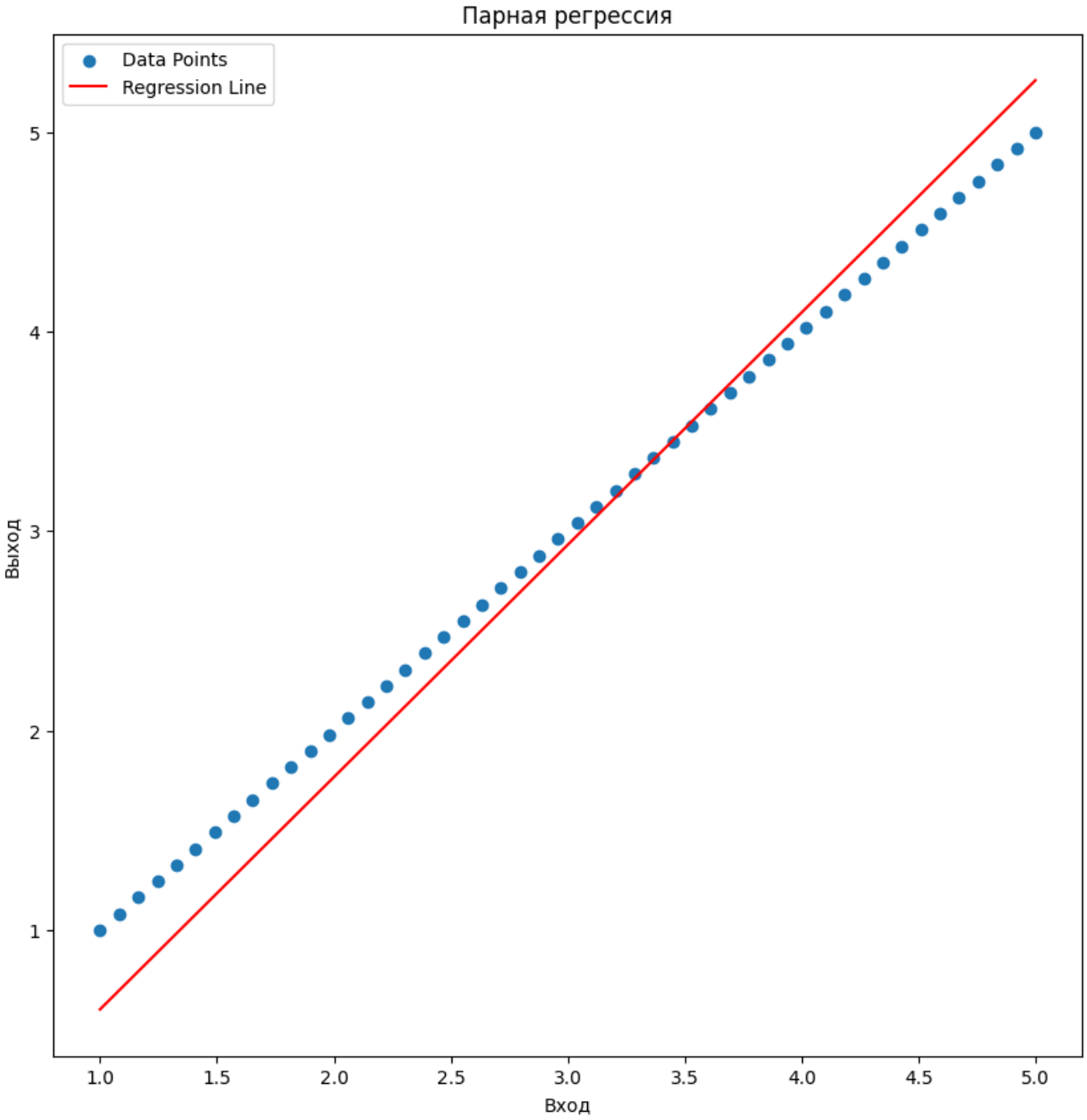

Показатель качества регрессии
Mean Square Error
Эпохи
0
10
20
30
40
50
0
20
40
60
80
100

Изобразим на графике точки набора данных (независимой и зависимой переменных) и линию построенной парной регрессии:

```
y_pred = linear_regression(x)
max_error = tf.reduce_max(tf.abs(y_pred - y))

print("MaxErr:", max_error.numpy())
```

```
MaxErr: 0.39848143
```

```
plt.scatter(x, y, label="Data Points")
plt.plot(x, linear_regression(x), color='red', label="Regression Line")

plt.xlabel("Вход")
plt.ylabel("Выход")
plt.title("Парная регрессия")
plt.legend()

plt.show()
```


Парная регрессия
Data Points
Regression Line
Выход
Вход
1
2
3
4
5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0