РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

Дисциплина: Методы машинного обучения

Студент: Шалыгин Георгий

Группа: НФИ-02

Москва **2023**

Вариант № 8

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**.**

   $f(x) = (x^3 + 2) \cdot e^{-x}$, $x \in [0,5]$

   max, min: (1.9670681, 0.8948191)

In [128]:

```
import tensorflow_datasets as tfds
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [129]:

```
x = tf.random.uniform(shape=[50], minval=0, maxval=5)
x = tf.Variable(x)
```

In [130]:

```
y = (x**3 + 2) * tf.exp(-x)
```

In [131]:

```
ymax = tf.reduce_max(y).numpy()
ymin = tf.reduce_min(y).numpy()
ymax, ymin
```

Out[131]:

```
(1.8220662, 0.895902)
```

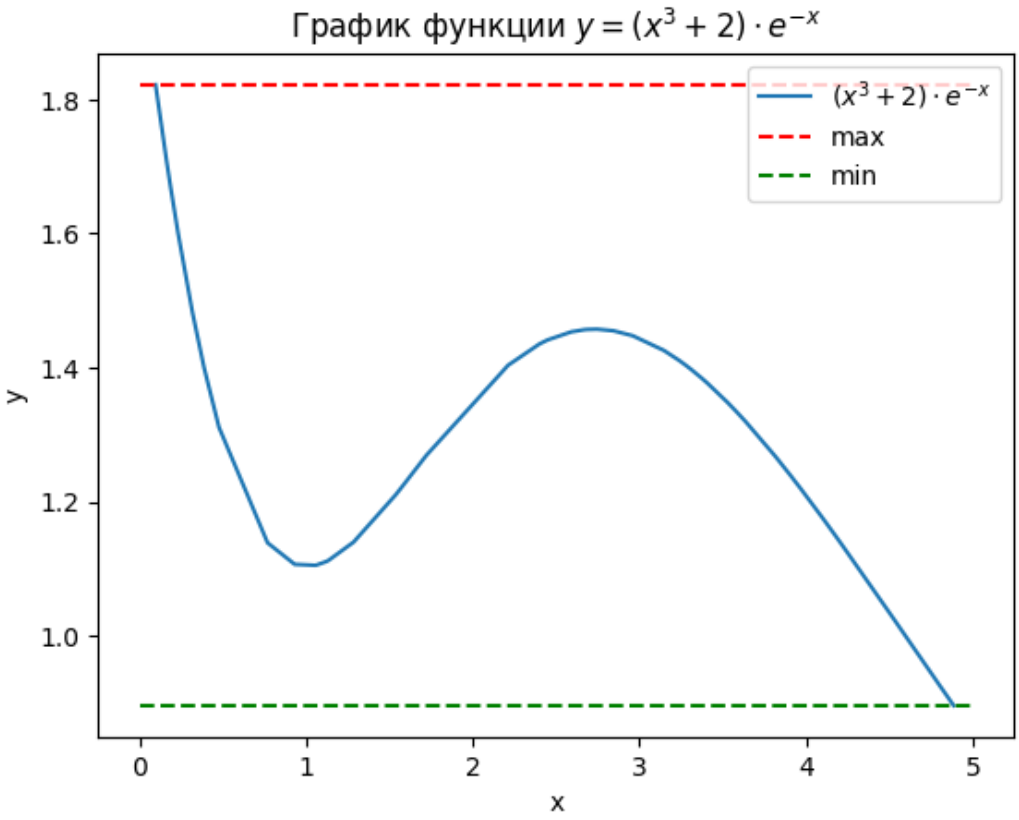
1. Постройте график функции с прямыми, соответствующими максимальному и минимальному значениям, подписывая оси и рисунок и создавая легенду.

In [228]:

```
plt.plot(tf.sort(x), y.numpy()[np.argsort(x.numpy())], label=r'$(x^3+2)\cdot e^{-x}$')
plt.plot([ymax]*6, 'r--', label='max')
plt.plot([ymin]*6, 'g--', label='min')
plt.xlabel('x')
plt.ylabel('y')
plt.title('График функции $y=(x^3+2)\cdot e^{-x}$')
plt.legend()
```

Out[228]:

```
<matplotlib.legend.Legend at 0x7f08c6853be0>
```



1. Найдите значения производной от функции порядка, указанного в индивидуальном задании, и постройте график полученной функции, подписывая оси и рисунок.

   Порядок: **4.**

In [133]:

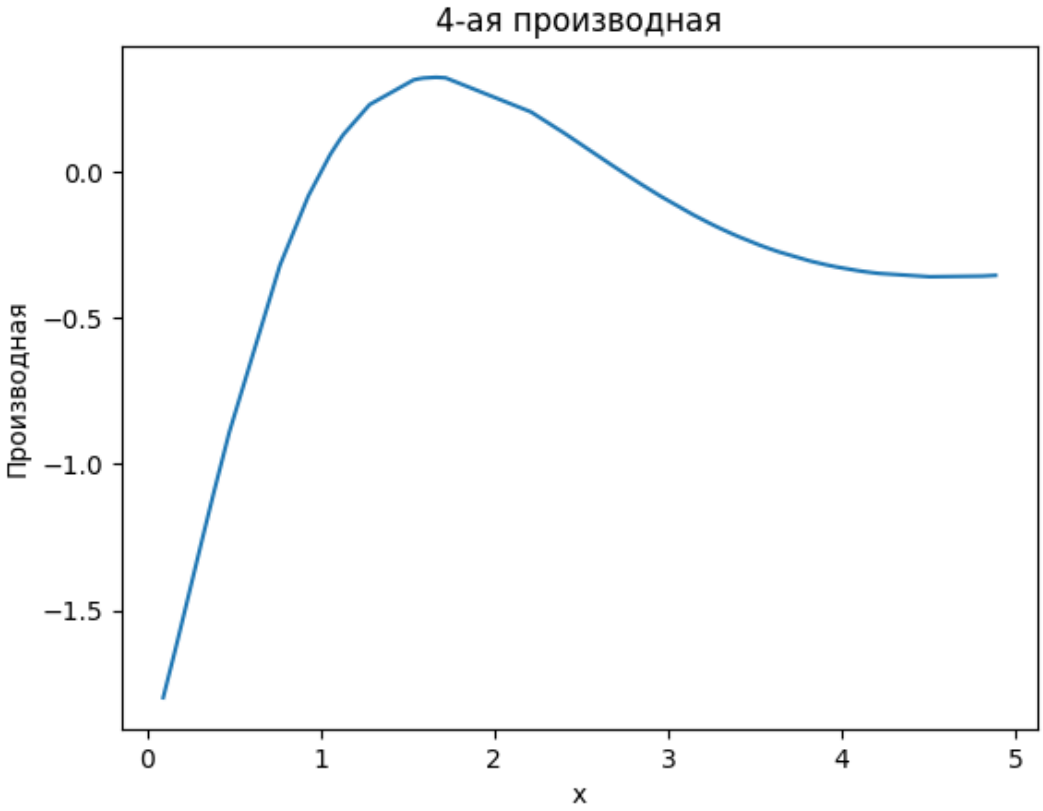
```
with tf.GradientTape() as tape1:
  with tf.GradientTape() as tape2:
    with tf.GradientTape() as tape3:
      with tf.GradientTape() as tape4:
        y = (x**3 + 2) * tf.exp(-x)
      dy = tape4.gradient(y, x)
    ddy = tape3.gradient(dy, x)
  dddy = tape2.gradient(ddy, x)
ddddy = tape1.gradient(y, x)
```

In [134]:

```
plt.plot(tf.sort(x), ddddy.numpy()[np.argsort(x.numpy())])
plt.xlabel('x')
plt.ylabel('Производная')
plt.title('4-ая производная')
```

Out[134]:

```
Text(0.5, 1.0, '4-ая производная')
```



1. 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**.**

   $f(x,y) = e^{-x}\sin$
   $(y),\ (x,y) \in [0,2]$
   $\times\,[0,3]$

   max, min: (0.9997786587835266, 0.0)

In [143]:

```
xs = tf.linspace(0, 2, 50)
ys = tf.linspace(0, 3, 50)
X, Y = tf.meshgrid(xs,ys)
X, Y = tf.Variable(X), tf.Variable(Y)
```

In [144]:

```
F = tf.exp(-X) * tf.sin(Y)
```

In [145]:

```
fmax = tf.reduce_max(F).numpy()
fmin = tf.reduce_min(F).numpy()
fmax, fmin
```

Out[145]:

```
(0.9997786587835266, 0.0)
```

1. Постройте **3d** график поверхности функции двух переменных**,** подписывая оси и рисунок**.**

In [230]:

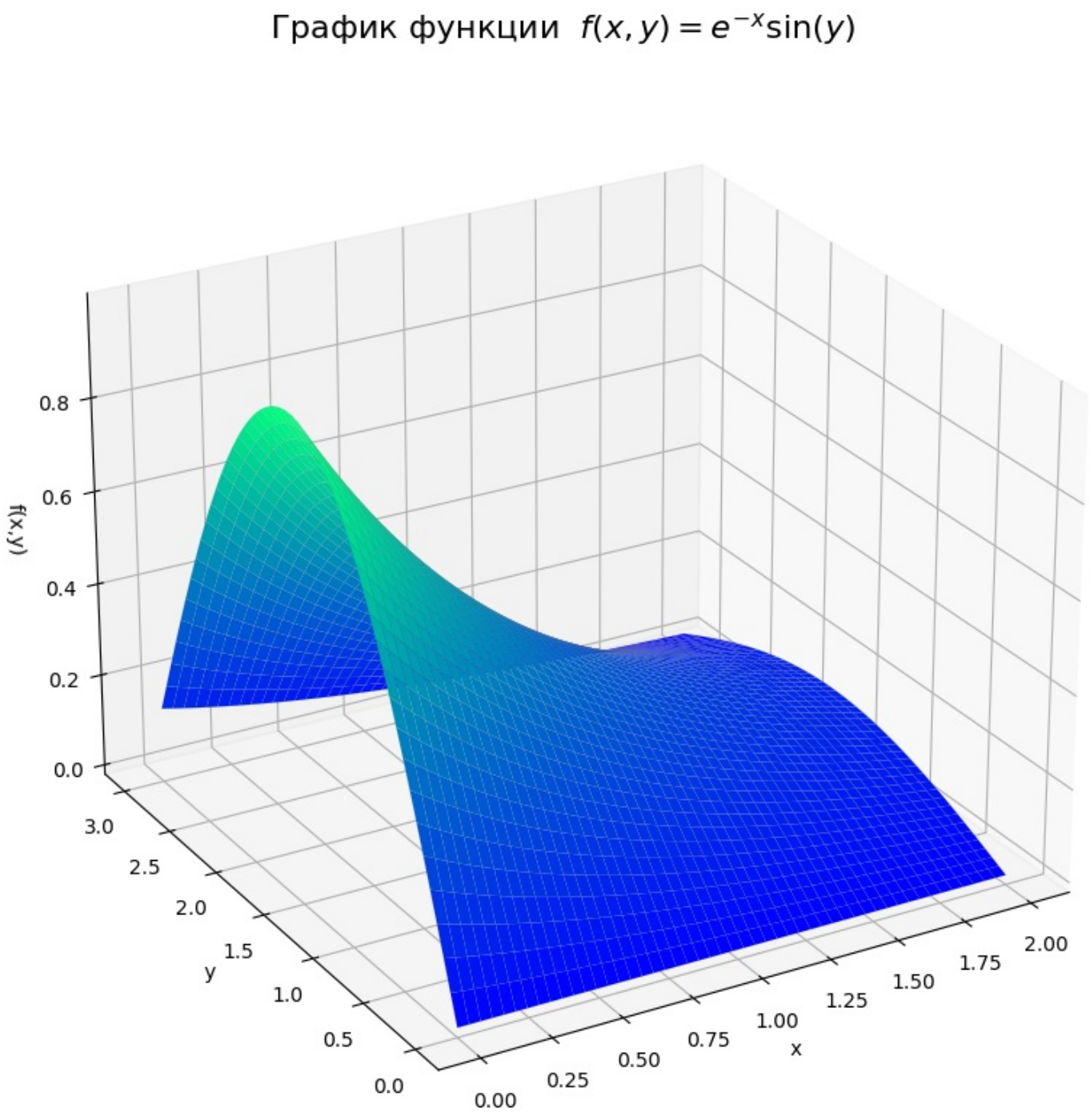
```
from mpl_toolkits import mplot3d
```

```
from matplotlib import cm
```

In [232]:

```
fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')


#ax.scatter( X, Y, F, s=100 )
ax.plot_surface(X, Y, F, \
       rstride=1, cstride=1, linewidth=0.05, cmap=cm.winter, antialiased=True, \
       edgecolors='gray')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('f(x,y)')
ax.set_title('График функции  $f(x,y) = e^{-x}\sin(y)$', fontsize=16)
ax.view_init( azim=-120, elev=25 )
```



1. Найдите значения смешанной производной от функции порядка, указанного в индивидуальном задании, и постройте **3d** график поверхности полученной функции, подписывая оси и рисунок.

   d^4

$\overline{d^3xdy}$

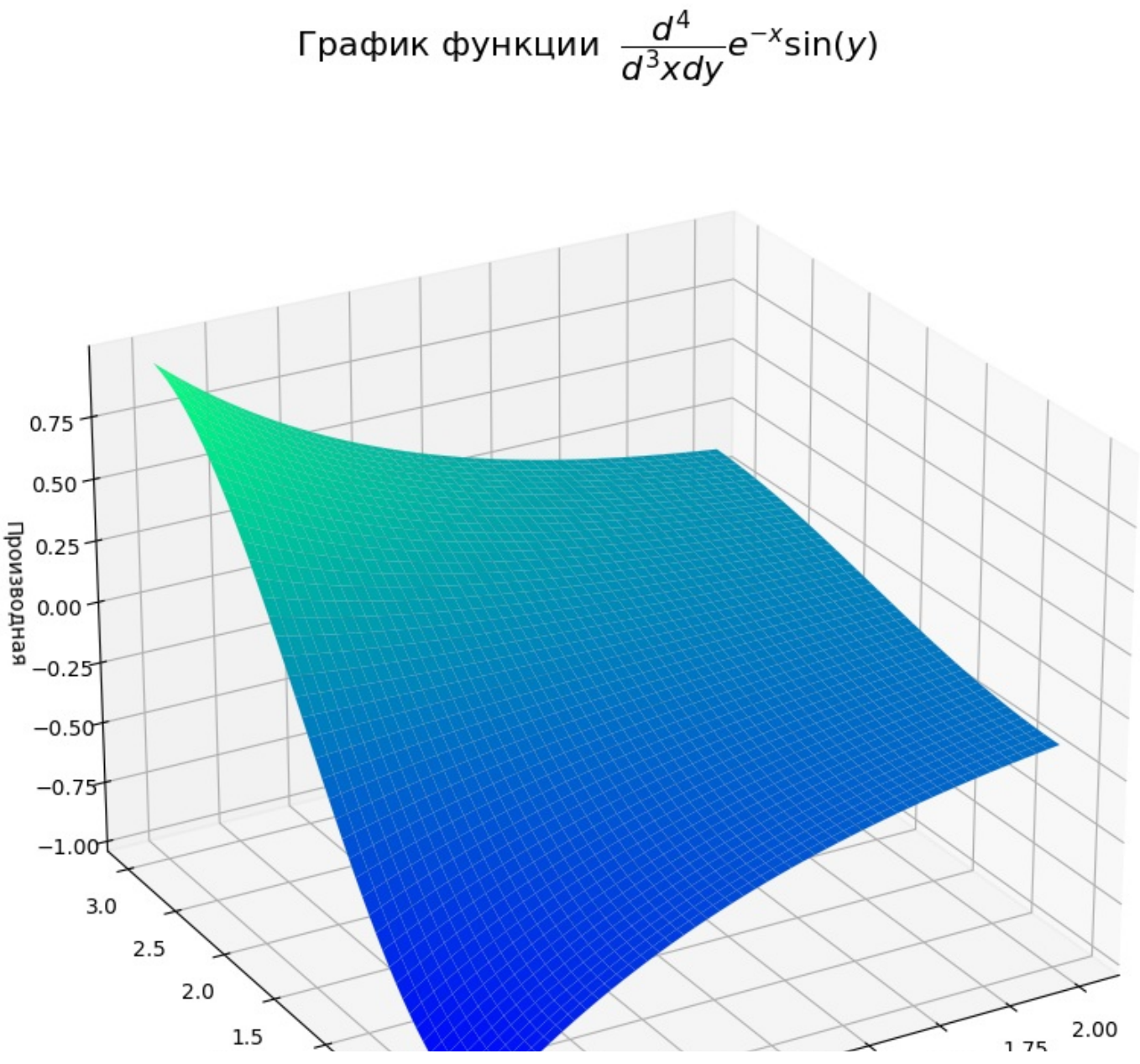
In [148]:

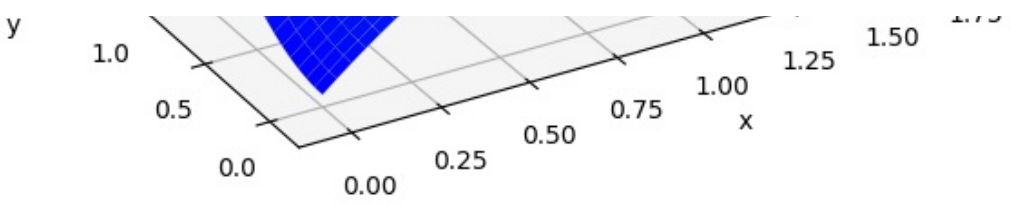
```
with tf.GradientTape() as t1:
  with tf.GradientTape() as t2:
    with tf.GradientTape() as t3:
      with tf.GradientTape() as t4:
        F = tf.exp(-X) * tf.sin(Y)
      dfdy = t4.gradient(F, Y)
    dfdydx = t3.gradient(dfdy, X)
  dfdydxdx = t2.gradient(dfdydx, X)
dfdydxdxdx = t1.gradient(dfdydxdx, X)
```

In [234]:

```
fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')


#ax.scatter( X, Y, dfdydxdxdx, s=100 )
ax.plot_surface(X, Y, dfdydxdxdx, \
       rstride=1, cstride=1, linewidth=0.05, cmap=cm.winter, antialiased=True, \
       edgecolors='gray')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('Производная')
ax.set_title('График функции  $\dfrac{d^4}{d^3xdy} e^{-x}\sin(y)$', fontsize=16)
ax.view_init( azim=-120, elev=25 )
```





1. 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.

In [223]:

```
w = tf.Variable(0.0)
b = tf.Variable(0.0)

def reg(x):
  return w * x + b

def loss(ypred, y):
  return tf.reduce_mean(tf.square(ypred - y))

def maxEr(ypred, y):
  return tf.math.reduce_max(tf.math.abs(ypred - y))
```

In [224]:

```
def train(x, y, lr=0.1):
  with tf.GradientTape() as t:
    ypred = reg(x)
    cur_loss = loss(ypred, y)
  dw, db = t.gradient(cur_loss, [w, b])
  w.assign_sub(lr * dw)
  b.assign_sub(lr * db)
  return cur_loss, maxEr(ypred, y)
```

In [225]:

```
epochs = 30
learning_rate = 0.1
history_loss = []
history_maxer = []

for i in range(epochs):
  cl, maxer = train(x, y, learning_rate)
  history_loss.append(cl.numpy())
  history_maxer.append(maxer.numpy())
  print(f'Epoch #{i}. Loss: {cl.numpy()}, MaxEr: {maxer.numpy()}')
```

```
Epoch #0. Loss: 1.729651689529419, MaxEr: 1.8220661878585815
Epoch #1. Loss: 1.0377914905548096, MaxEr: 2.3222451210021973
Epoch #2. Loss: 0.6854184865951538, MaxEr: 1.6292312145233154
Epoch #3. Loss: 0.4993059039115906, MaxEr: 1.57151198387146
Epoch #4. Loss: 0.3952536880970001, MaxEr: 1.4871877431869507
Epoch #5. Loss: 0.3322763741016388, MaxEr: 1.3779546022415161
Epoch #6. Loss: 0.2903704345226288, MaxEr: 1.3742138147354126
Epoch #7. Loss: 0.2597249448299408, MaxEr: 1.3007147312164307
Epoch #8. Loss: 0.23548439145088196, MaxEr: 1.2793651819229126
Epoch #9. Loss: 0.2152068018913269, MaxEr: 1.2250763177871704
Epoch #10. Loss: 0.1976291835308075, MaxEr: 1.196963906288147
Epoch #11. Loss: 0.18206769227981567, MaxEr: 1.1536962985992432
Epoch #12. Loss: 0.1681261658668518, MaxEr: 1.1239217519760132
Epoch #13. Loss: 0.15553374324321747, MaxEr: 1.0875463485717773
Epoch #14. Loss: 0.14417554438114166, MaxEr: 1.0584371089935303
```

```
Epoch #15. Loss: 0.1338583528995514, MaxEr: 1.026809573173523
Epoch #16. Loss: 0.1244935691356659, MaxEr: 0.9993607997894287
Epoch #17. Loss: 0.1159885972738266, MaxEr: 0.9713122844696045
Epoch #18. Loss: 0.10826221108436584, MaxEr: 0.9458848834037781
Epoch #19. Loss: 0.1012420579791069, MaxEr: 0.9207319617271423
Epoch #20. Loss: 0.09486299753189087, MaxEr: 0.8973900675773621
Epoch #21. Loss: 0.08906625956296921, MaxEr: 0.8746953010559082
Epoch #22. Loss: 0.08379855006933212, MaxEr: 0.8533693552017212
Epoch #23. Loss: 0.07901152968406677, MaxEr: 0.8328244686126709
Epoch #24. Loss: 0.07466129958629608, MaxEr: 0.8133890628814697
Epoch #25. Loss: 0.07070799916982651, MaxEr: 0.7947571277618408
Epoch #26. Loss: 0.0671154037117958, MaxEr: 0.7770682573318481
Epoch #27. Loss: 0.06385058164596558, MaxEr: 0.7601548433303833
Epoch #28. Loss: 0.06088364124298096, MaxEr: 0.7440668344497681
Epoch #29. Loss: 0.05818740278482437, MaxEr: 0.728705644607544
```

1. 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`.

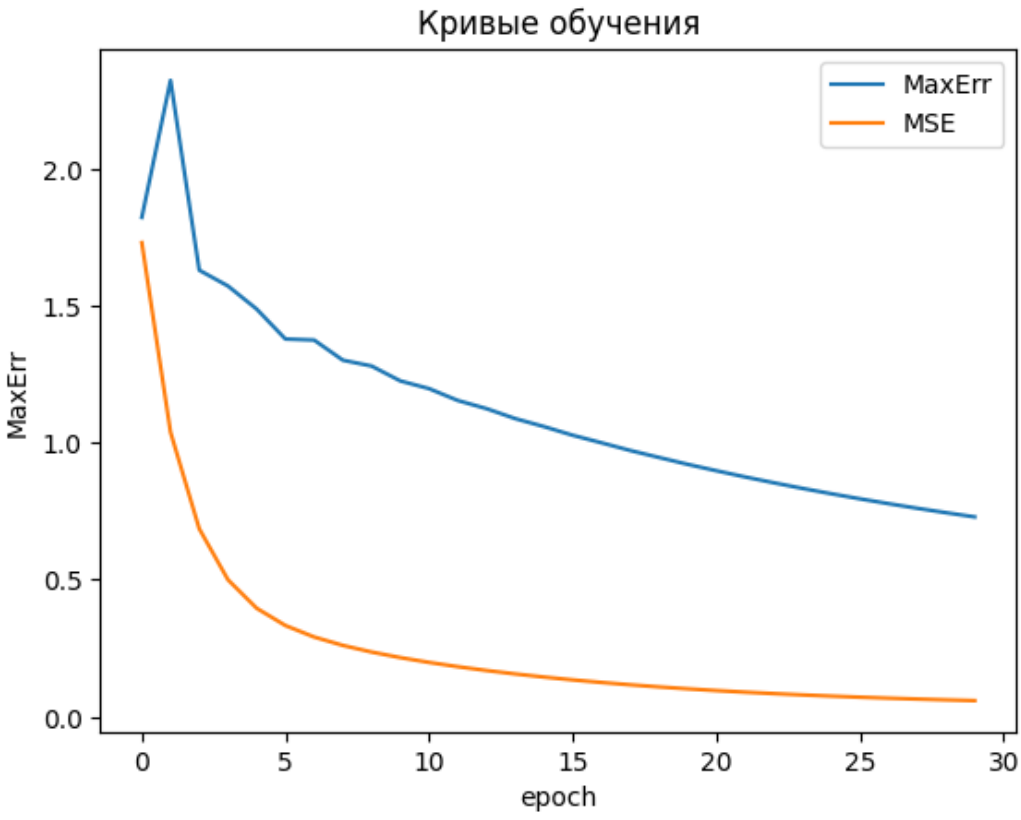
   MaxEr

In [235]:

```
plt.plot(range(epochs), history_maxer, label='MaxErr')
plt.plot(range(epochs), history_loss, label='MSE') #просто так
plt.xlabel('epoch')
plt.ylabel('MaxErr')
plt.title('Кривые обучения')
plt.legend()
```

Out[235]:

```
<matplotlib.legend.Legend at 0x7f08c5b7bf40>
```



1. Изобразите на графике точки набора данных (независимой и зависимой переменных) и линию построенной парной регрессии, подписывая оси и рисунок, и создавая легенду.

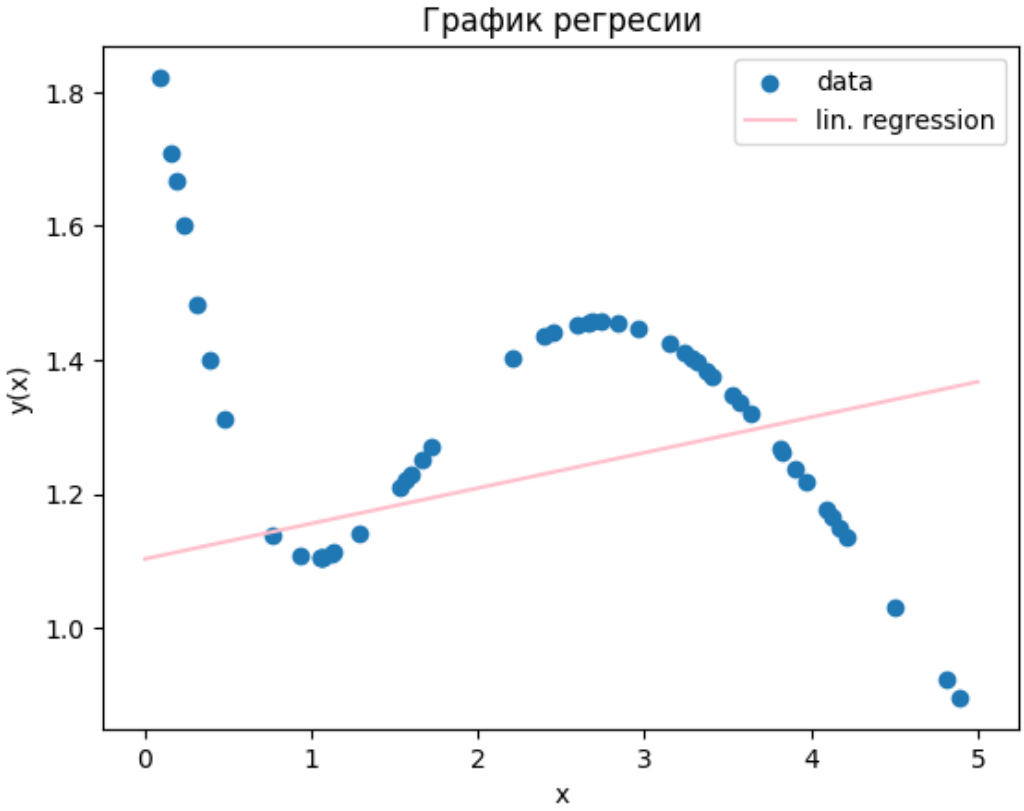
In [227]:

```
x_reg = tf.Variable([0.0, 5])
```

```
y_reg = x_reg * w + b
plt.scatter(x, y, label='data')
plt.plot(x_reg, y_reg, label='lin. regression', color='pink')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.title('График регресии')
plt.legend()
```

Out[227]:

```
<matplotlib.legend.Legend at 0x7f08c674f280>
```



In [ ]: