In [ ]:
```
# Adrian Kokoszka 19727
# importowanie potrzebnych bibliotek
import tensorflow as tf #biblioteka tensorflow
import numpy as np #biblioteka numpy, aby móc korzystać z funkcji związanych z obl
import matplotlib.pyplot as plt #biblioteka matplotlib, aby móc korzystać z funkcj
from mpl_toolkits.mplot3d import Axes3D #biblioteka do rysowania wykresów 3D
print(tf.__version__) #sprawdzenie wersji biblioteki tensorflow
```

```
2.11.0
```

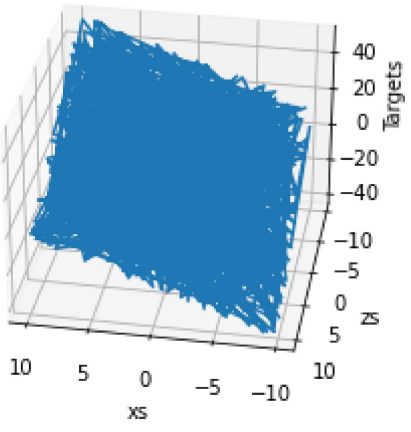
In [ ]:
```
observations=1000 #ilość obserwacji
xs = np.random.uniform(low=-10,high=10, size=(observations,1)) #losowanie wartości
xz = np.random.uniform(low=-10,high=10, size=(observations,1)) #losowanie wartości
inputs=np.column_stack((xs,xz)) #łączenie dwóch tablic w jedną
print(inputs.shape) #wypisanie rozmiaru tablicy, w tym przypadku 1000 wierszy i 2 k
```

```
(1000, 2)
```

In [ ]:
```
noise = np.random.uniform(low=-1,high=1, size=(observations,1)) # losowanie wartoś
targets = 2*xs - 3*xz + 5 + noise #funkcja liniowa, która ma być wyznaczona przez s
np.savez('TF_dataset', inputs=inputs, targets=targets)# zapisanie danych do pliku
print(targets.shape) # wypisanie rozmiaru tablicy, w tym przypadku 1000 wierszy i 1
```

```
(1000, 1)
```

In [ ]:
```
targets = targets.reshape(observations,) #zmiana rozmiaru tablicy targets bez zmian
xs = xs.reshape(observations,) #zmiana rozmiaru tablicy xs, bez zmiany jej zawartos
xz = xz.reshape(observations,) #zmiana rozmiaru tablicy xz bez zmiany jej zawartoś
fig = plt.figure() #tworzenie wykresu
ax = fig.add_subplot(111, projection='3d') #wyswietlanie wykresu w trzech wymiarach
ax.plot(xs,xz,targets) #rysowanie wykresu
ax.set_xlabel('xs') #podpisanie osi x
ax.set_ylabel('zs') #podpisanie osi y
ax.set_zlabel('Targets') #podpisanie osi z
ax.view_init(azim=100) #ustawienie kąta widzenia wykresu na 100 stopni
plt.show() #wyświetlenie wykresu
```



In [ ]:
```
init_range = 0.1 #zakres losowania wartości początkowych
weights = np.random.uniform(low=-init_range,high=init_range, size=(2,1)) #losowani
biases = np.random.uniform(low=-init_range,high=init_range, size=1) #losowanie odch
print(weights,biases)   #wypisanie wartości początkowych wag i odchylenia
```

```
[[0.0033705 ]
 [0.09864253]] [-0.09122773]
```

In [ ]:
```
targets = targets.reshape(observations,1) #Przekształcenie tablicy targets do post
eta = 0.02 #współczynnik uczenia
```

```
for i in range (100): #pętla wykonująca się 100 razy
    outputs = np.dot(inputs, weights) + biases #obliczenie wartości wyjściowej siec
    deltas = outputs - targets #obliczenie wartości błędu sieci neuronowej

    loss = np.sum(deltas ** 2)/2/observations #obliczenie wartości funkcji kosztu
    print(loss) #wypisanie wartości funkcji kosztu

    deltas_scaled = deltas/observations #obliczenie wartości deltas_scaled dzieląc
    weights = weights - eta * np.dot(inputs.T, deltas_scaled) #obliczenie nowych w
    biases = biases - eta * np.sum(deltas_scaled) #obliczenie nowych wartości odch
```

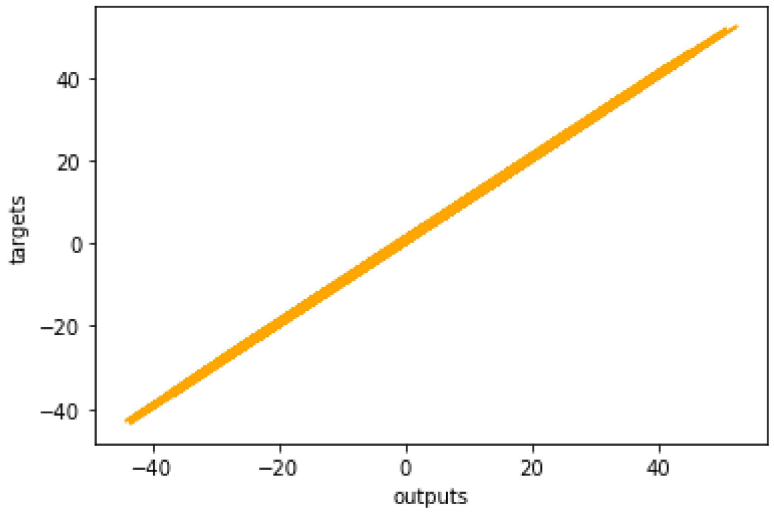
```
231.23127651835836
39.798716611413724
15.462055166458388
12.00830636264627
11.1825447050718
10.701456303169202
10.278528176135604
9.87725332823896
9.492482795693599
9.123024207106244
8.768203589801074
8.427432858664513
8.100154655825856
7.785834627410942
7.483959694385214
7.194037107238165
6.915593628102412
6.648174756583876
6.39134398767038
6.144682099202046
5.907786467575935
5.680270410547366
5.461762556052499
5.251906236021629
5.0503589041936365
4.856791576981348
4.670888296475077
4.492345614707868
4.320872098340566
4.156187852958253
3.998024066201537
3.846122568986976
3.700235414100394
3.5601244714752576
3.425561039495475
3.296325471688166
3.17220681819702
3.0530024814510783
2.938517885466859
2.8285661582440085
2.722967826736126
2.6215505238987795
2.5241487073366105
2.4306033890902206
2.34076187612179
2.2544775210758545
2.171609482908363
2.0920224969933296
2.0155866543317993
1.9421771895027713
1.8716742770099073
1.8039628356916304
1.7389323408753754
1.6764766439693104
1.6164937991971178
1.558885897192976
1.5035589051851348
1.4504225135072066
1.3993899881866523
1.3503780293698355
1.3033066353525395
1.2580989719940343
1.2146812473015185
1.1729825909802247
```

```
1.1329349387525822
1.0944729212576163
1.0575337573492254
1.0220571516191812
0.9879851959775691
0.9552622751300169
0.9238349757974422
0.8936519995301168
0.8646640789737473
0.8368238974508906
0.810086011726442
0.7844067778311193
0.7597442798218742
0.7360582613629318
0.7133100600158111
0.6914625441310192
0.6704800522384637
0.6503283348376019
0.6309744984923441
0.6123869521394465
0.5945353555227492
0.5773905696690966
0.560924609325097
0.5451105972770892
0.5299227204797514
0.5153361879217355
0.5013271901595627
0.48787286045372025
0.4749512374435201
0.46254122929980696
0.4506225792969798
0.4391758327481596
0.42818230524950446
0.4176240521818544
0.40748383941992133
0.39774511520120187
```

In [ ]:
```
print (weights, biases) #wypisanie wartości wag i odchyleń po 100 powtórzeniach pęt
```

```
[[ 1.99727464]
 [-2.99898783]] [4.30483458]
```

In [ ]:
```
plt.plot(outputs, targets, color='orange') #rysowanie wykresu, gdzie osie x i y to
plt.xlabel('outputs') #podpisanie osi x jako outputs
plt.ylabel('targets') #podpisanie osi y jako targets
plt.show() #wyświetlenie wykresu
```



In [ ]: