Мохляков П.А.
М8О-408Б-19
Вариант 3

Лабораторная работа 2

In [2]:
```
import tensorflow as tf
from tensorflow import keras
from keras import layers
import numpy as np
import matplotlib.pyplot as plt
import time
```

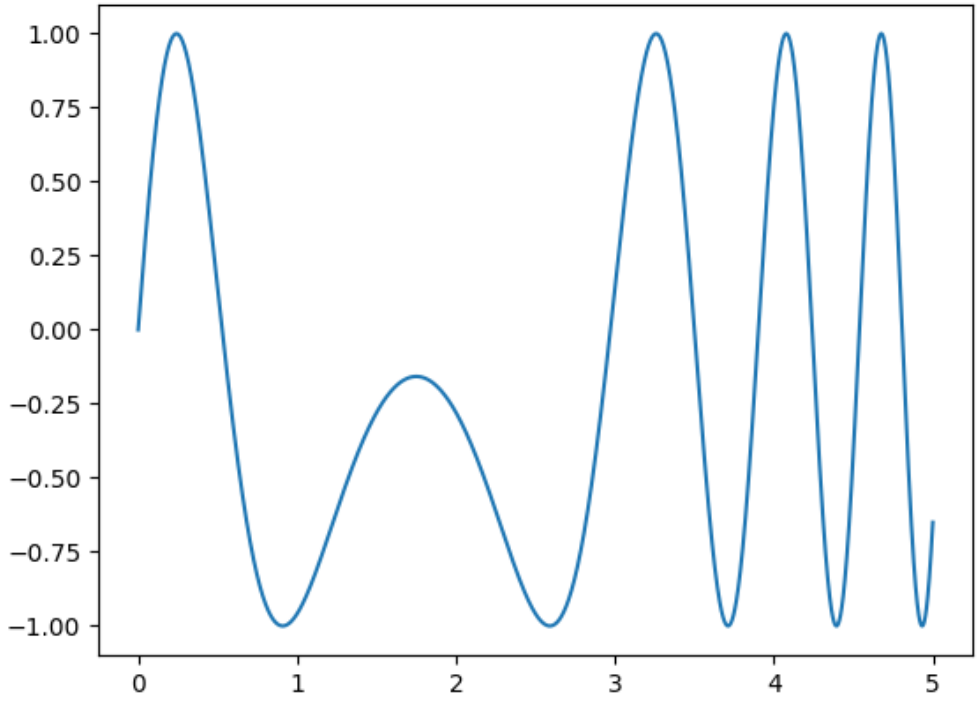
Прогнозирование

In [3]:
```
def func(t: float):
    return np.sin(-2*t**2 + 7*t)
```

In [4]:
```
h = 0.001
t = (0, 5)
D = 5
ans_x = np.arange(t[0],t[1] + h, h)
ans = func(ans_x)
```

In [5]:
```
plt.plot(ans_x, ans)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x7f448bf92070>]
```



Готовим датасет

In [6]:
```
X = [ans[i:i+D].tolist() for i in range(0, len(ans) - D)]
y = [ans[i] for i in range(D, len(ans))]
```

Создаем модель

In [8]:
```
predictor =  keras.Sequential([
        layers.Dense(1,input_dim=D, activation="linear", name="pred"),
    ]
```

```
)
predictor.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 pred (Dense)                (None, 1)                 6

=================================================================
Total params: 6
Trainable params: 6
Non-trainable params: 0
_________________________________________________________________
```

Компилируем модель

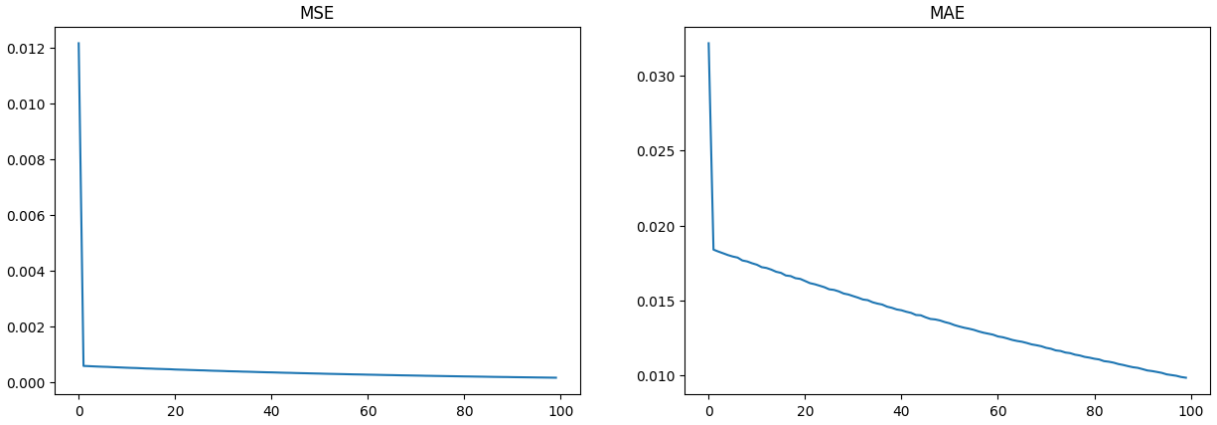
In [9]:
```
opt = keras.optimizers.SGD(learning_rate=0.1)
predictor.compile(loss='mse', optimizer=opt, metrics=['mae'])
```

Тренеруем модель

In [10]:
```
epochs = 100
time_start = time.time()
hist = predictor.fit(
    X,
    y,
    epochs=epochs,
    verbose=0,
    shuffle=True
    )
time_finish = time.time()
mse_loss, mae_loss = predictor.evaluate(X, y, verbose=0)

print(f'Fit time: {(time_finish - time_start):.{2}f}s')
print(f'Result MSE: {mse_loss}')
print(f'Result MAE: {mae_loss}')

fig, ax = plt.subplots(1, 2)
fig.set_figwidth(15)

ax[0].set_title('MSE')
ax[1].set_title('MAE')

ax[0].plot(range(epochs), hist.history['loss'])
ax[1].plot(range(epochs), hist.history['mae'])
```

```
Fit time: 14.22s
Result MSE: 0.00016730026982259005
Result MAE: 0.009755713865160942
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x7f448809fe20>]
```



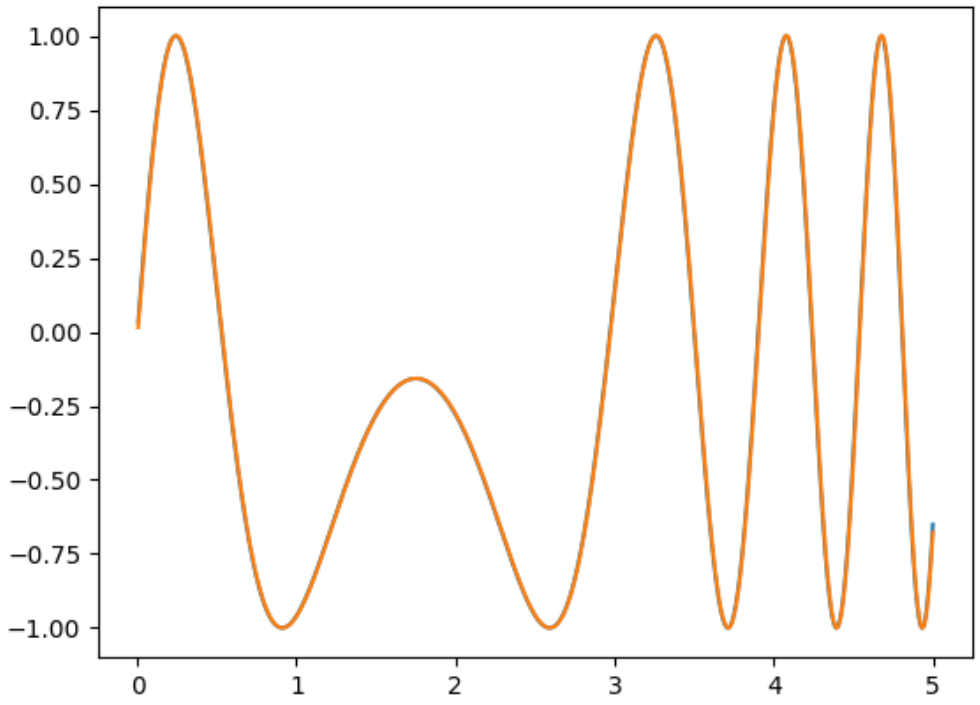
Получаем предсказания модели

In [11]:
```
my_ans = predictor.predict(X).flatten()
```

```
157/157 [==============================] - 0s 573us/step
```

In [12]:
```
plt.plot(ans_x[D:], y)
plt.plot(ans_x[D:], my_ans)
```

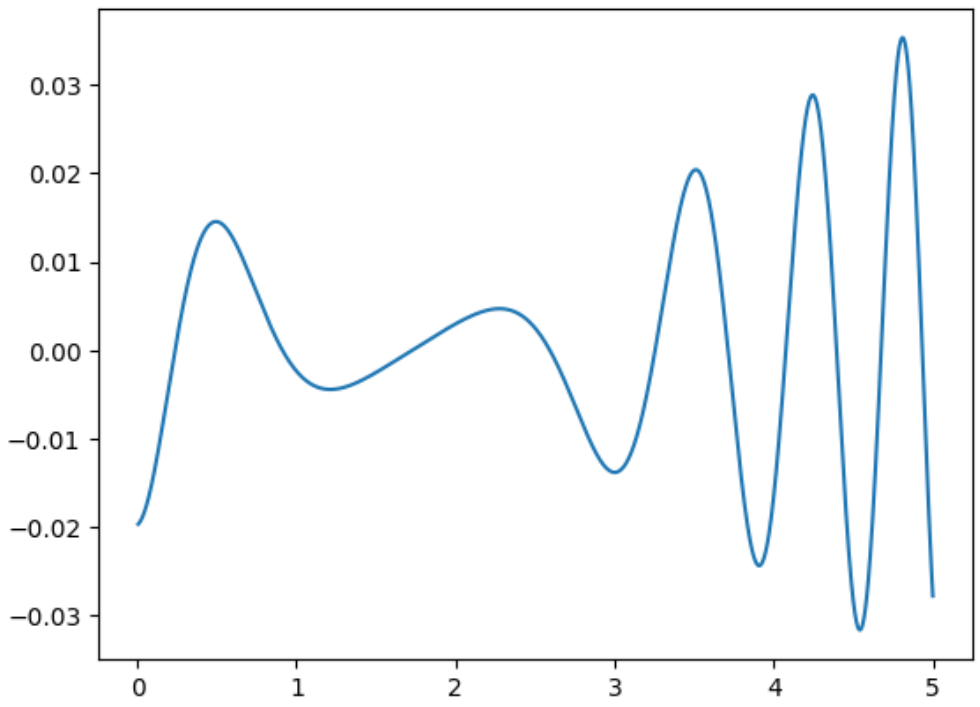
Out[12]:
```
[<matplotlib.lines.Line2D at 0x7f44881c1670>]
```



Находим абсолютное отклонение

In [13]:
```
errors = my_ans - y
plt.plot(ans_x[D:], errors)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x7f44881e33a0>]
```



Зашумленный сигнал в чистый

In [14]:
```
def noized(t):
```

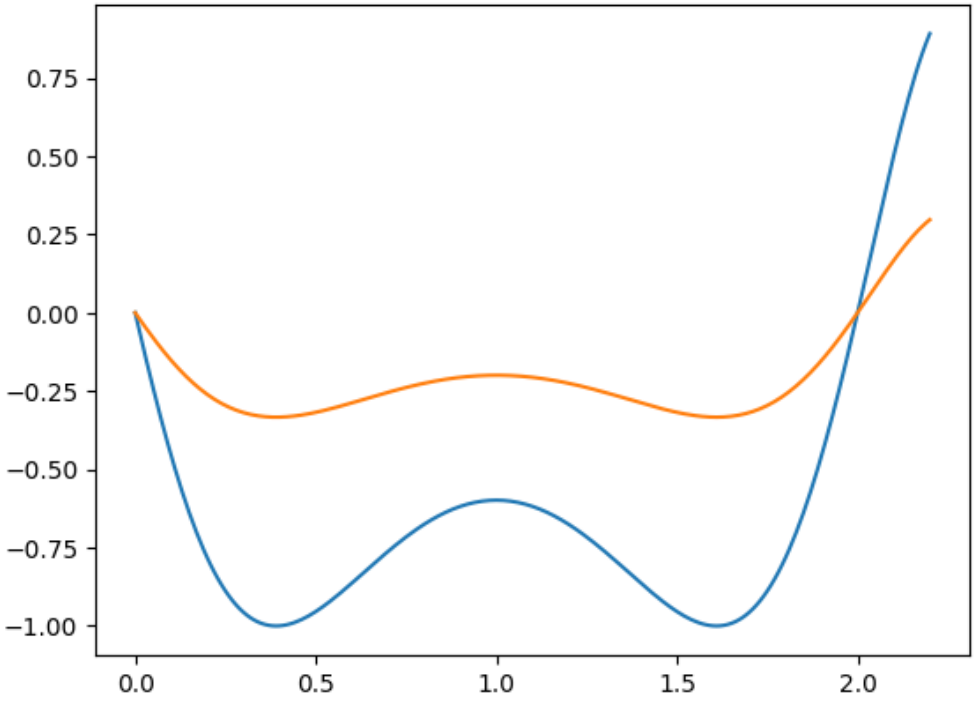
```
    return np.sin(2.5*t**2 - 5*t)

def resl_sig(t):
    return np.sin(2.5*t**2 - 5*t + 4*np.pi)/3
```

In [15]:
```
h = 0.01
t = (0, 2.2)
D = 4
```

In [16]:
```
x_points = np.arange(t[0], t[1] + h, h)
noized_points = noized(x_points)
real_points = resl_sig(x_points)
```

In [17]:
```
plt.plot(x_points, noized_points)
plt.plot(x_points, real_points)
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x7f44807670d0>]
```



Готовим датасет

In [18]:
```
X = [noized_points[i:i+D].tolist() for i in range(0, len(noized_points) - D)]
y = [real_points[i] for i in range(D, len(real_points))]
```

Создаем модель

In [24]:
```
predictor =  keras.Sequential([
        layers.Dense(1,input_dim=D, activation="linear", name="pred"),
    ]
)
predictor.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 pred (Dense)                (None, 1)                 5

=================================================================
Total params: 5
Trainable params: 5
Non-trainable params: 0
_________________________________________________________________
```

Компилируем модель

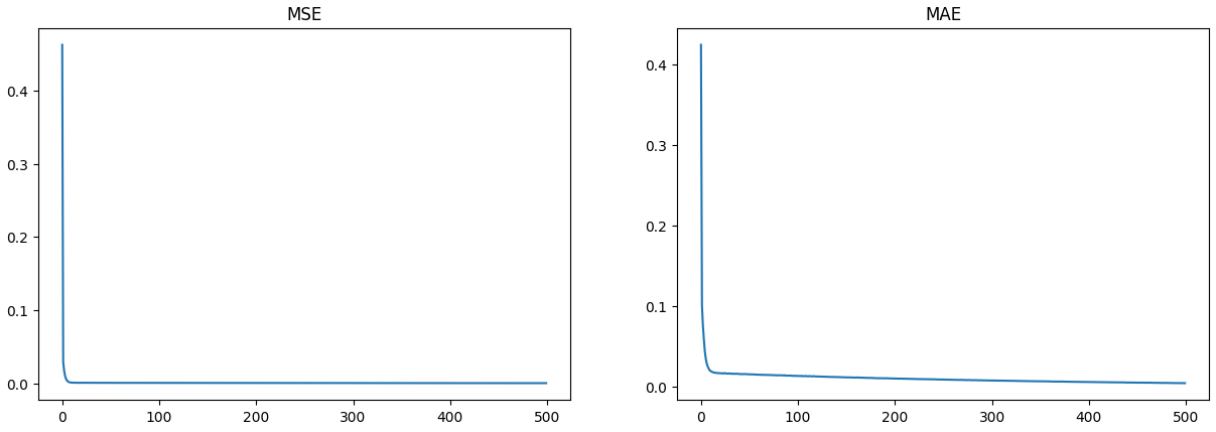
In [25]:
```
opt = keras.optimizers.SGD(learning_rate=0.1)
predictor.compile(loss='mse', optimizer=opt, metrics=['mae'])
```

Тренеруем модель

In [26]:
```
epochs = 500
time_start = time.time()
hist = predictor.fit(
    X,
    y,
    epochs=epochs,
    verbose=0,
    shuffle=True
    )
time_finish = time.time()
mse_loss, mae_loss = predictor.evaluate(X, y, verbose=0)

print(f'Fit time: {(time_finish - time_start):.{2}f}s')
print(f'Result MSE: {mse_loss}')
print(f'Result MAE: {mae_loss}')

fig, ax = plt.subplots(1, 2)
fig.set_figwidth(15)

ax[0].set_title('MSE')
ax[1].set_title('MAE')

ax[0].plot(range(epochs), hist.history['loss'])
ax[1].plot(range(epochs), hist.history['mae'])
```

```
Fit time: 9.40s
Result MSE: 3.779832331929356e-05
Result MAE: 0.004535430110991001
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x7f448047de50>]
```



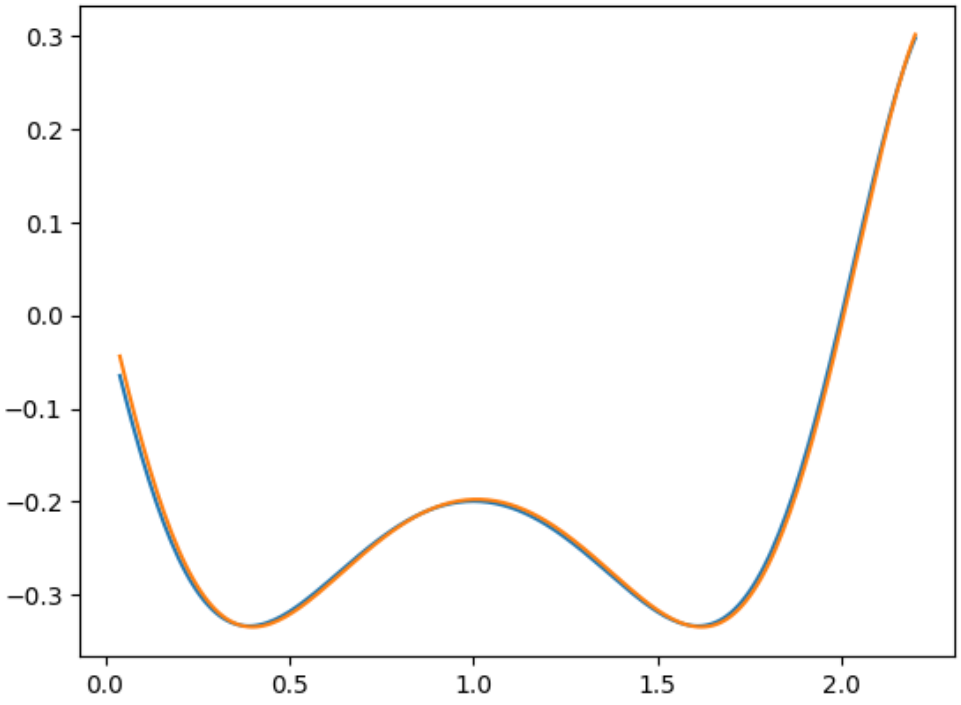
Рисуем сигнал

In [27]:
```
my_denoized = predictor.predict(X).flatten()
```

```
7/7 [==============================] - 0s 762us/step
```

In [28]:
```
plt.plot(x_points[D:], y)
plt.plot(x_points[D:], my_denoized)
```

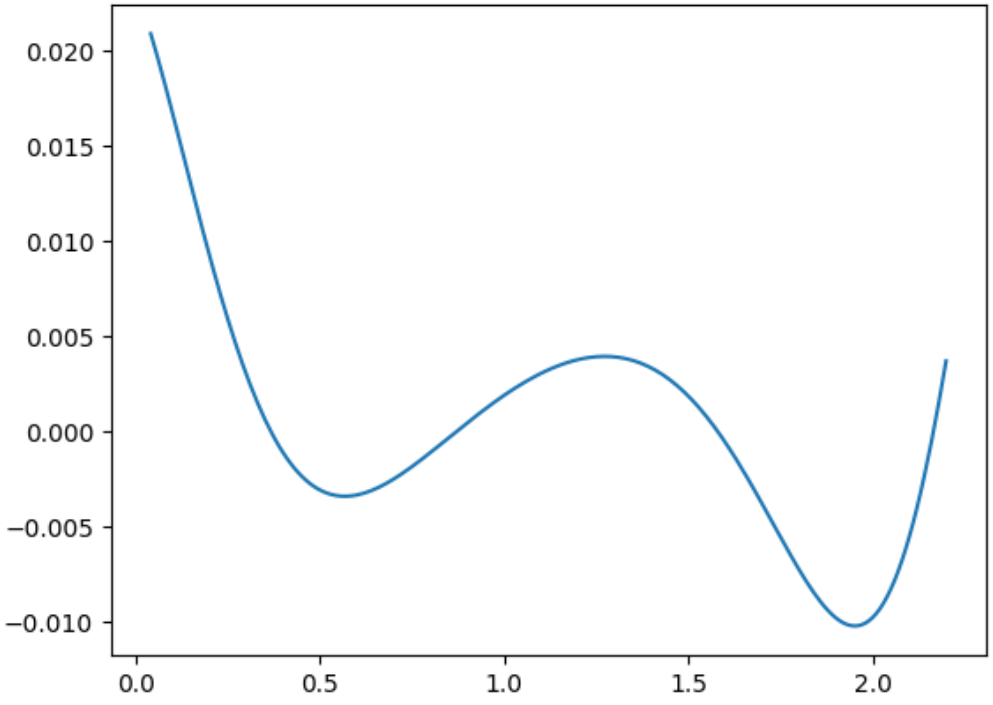
Out[28]:
```
[<matplotlib.lines.Line2D at 0x7f44804290d0>]
```



Находим абсолютное отконение

In [29]:
```
errors = my_denoized - y
plt.plot(x_points[D:], errors)
```

Out[29]:
```
[<matplotlib.lines.Line2D at 0x7f44803974c0>]
```



In [ ]: