In [ ]:

```
Пупов Нікіта
Варіант 11
```

In [ ]:

```
from itertools import chain
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from functools import reduce
```

In [12]:

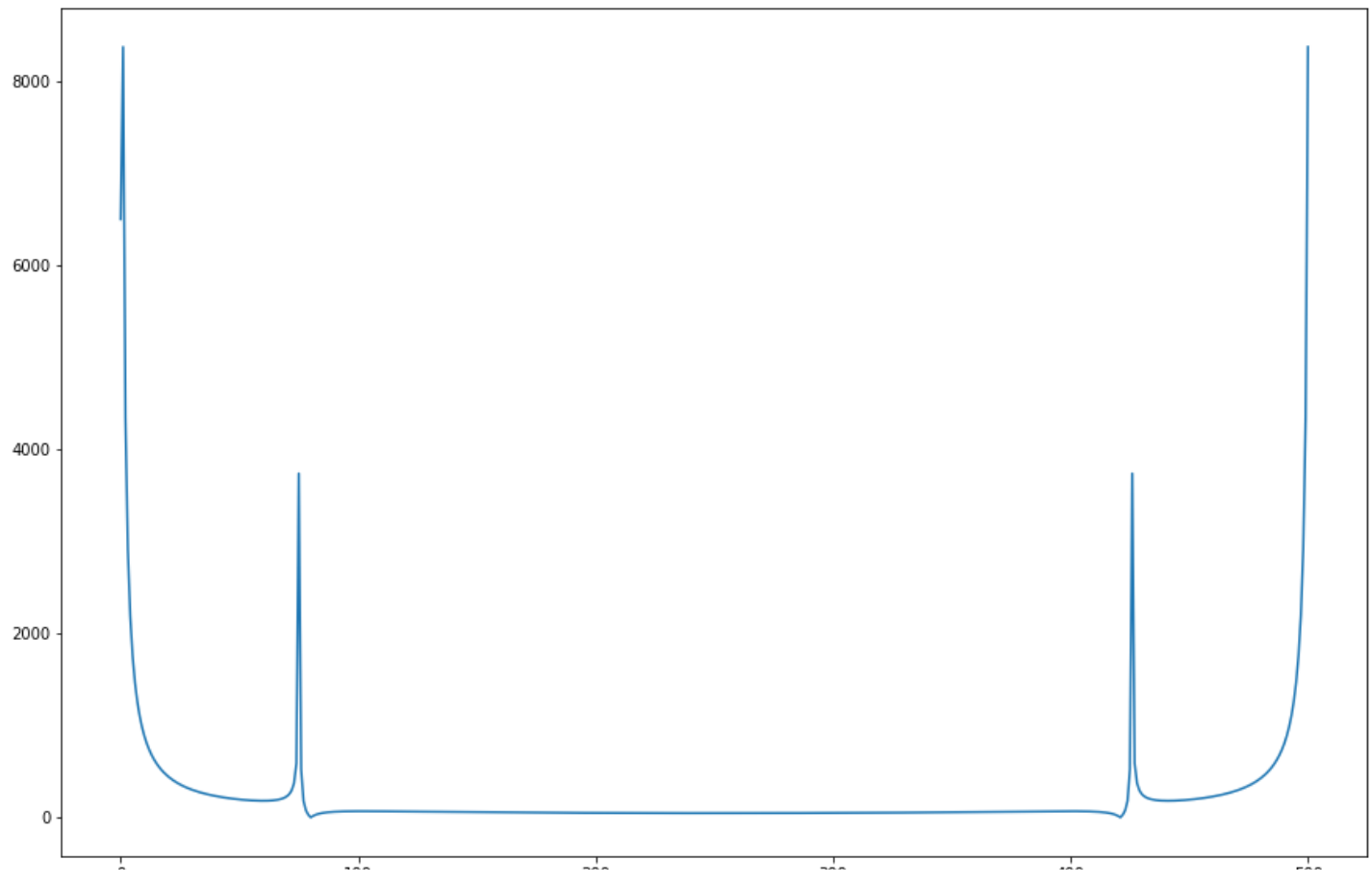
```
def plot_graph(X, y, color=None):
 plt.figure(figsize=(15, 10))

 if color is None:
    plt.plot(X, y)
 else:
    plt.plot(X, y, color=color)

 plt.show()
```

In [ ]:

```
T = 5
y = np.fromfile('f11.txt', sep=' ')
X = np.linspace(0, T, len(y))
N = len(y)
plot_graph(X, y)
```

In [14]:

```
abs_ft = np.abs(np.fft.fft(y))
plot_graph(range(N), abs_ft)
```



In [15]:

```
abs_ft_half = abs_ft[:N // 2]

maximums, = np.where(np.logical_and.reduce(
 np.array([abs_ft_half > np.roll(abs_ft_half, i) for i in
chain(range(-5, 0), range(1, 6))]),
 axis=0
))

maximums = maximums[(maximums >= 5) & (maximums < N // 2 - 5)]

#we have 2 maximums, therefore k = 6
tuple(maximums)
```

Out[15]:

```
(75, 102)
```

In [23]:

```
f_max1 = maximums[0] / T
f_max1
f_max2 = maximums[1] / T
f_max2
```

Out[23]:

```
20.4
```

In [58]:

```
values = pd.DataFrame(
 np.stack((X ** 3, X ** 2, X, np.sin(2 * np.pi * f_max1 * X),
           np.sin(2 * np.pi * f_max2 * X), np.ones(N)), axis=1),
 columns=['a1', 'a2', 'a3', 'a4', 'a5', 'a6']
)
values.head()
```

Out[58]:

	a1	a2	a3	a4	a5	a6
0	0.000000	0.0000	0.00	0.000000	0.000000	1.0
1	0.000001	0.0001	0.01	0.809017	0.958522	1.0
2	0.000008	0.0004	0.02	0.951057	0.546394	1.0
3	0.000027	0.0009	0.03	0.309017	-0.647056	1.0
4	0.000064	0.0016	0.04	-0.587785	-0.915241	1.0

In [ ]:

```
Квадратична похибка:
(f(xi)-fi)^2
Мінімізуємо квадратичну похибку
((f(x1)-f1)^2)+...)'= 0

2(f(x1)-f1)*(x1^3) + .... = 0 |a1

2(f(x1)-f1)*(x1^2) + .... = 0 |a2

2(f(x1)-f1)*(x1) + .... = 0 |a3

2(f(x1)-f1)*(sin a4 f_max1) + .... = 0 |a4

2(f(x1)-f1)*(sin a5 with f_max2) + .... = 0 |a5

2(f(x1)-f1) + .... = 0 |a6
```

In [50]:

```
def first_funk(xArr, fArr):
    a1 = 0;
    a2 = 0;
    a3 = 0;
    a4 = 0;
    a5 = 0;
    a6 = 0;
    minus = 0;
    for x,f in zip(xArr, fArr):
        a1 += 2*pow(x, 6)
        a2 += 2*pow(x, 5)
        a3+=2*pow(x, 4)
        a4+=2*pow(x,3)*np.sin(2 * np.pi * f_max1 * x)
        a5+=2*pow(x,3)*np.sin(2 * np.pi * f_max2 * x)
        a6+=2*pow(x,3)
        minus+=2*pow(x, 3)*f

    return [[a1, a2, a3, a4, a5, a6], minus]

def second_funk(xArr, fArr):
    a1 = 0;
    a2 = 0;
    a3 = 0;
    a4 = 0;
    a5 = 0;
    a6 = 0;
    minus = 0;
    for x,f in zip(xArr, fArr):
        a1 += 2*pow(x, 5)
        a2 += 2*pow(x, 4)
        a3+=2*pow(x, 3)
        a4+=2*pow(x,2)*np.sin(2 * np.pi * f_max1 * x)
        a5+=2*pow(x,2)*np.sin(2 * np.pi * f_max2 * x)
        a6+=2*pow(x,2)
        minus+=2*pow(x,2)*f

    return [[a1, a2, a3, a4, a5, a6], minus]

def third_funk(xArr, fArr):
    a1 = 0;
    a2 = 0;
    a3 = 0;
    a4 = 0;
    a5 = 0;
    a6 = 0;
    minus = 0;
    for x,f in zip(xArr, fArr):
        a1 += 2*pow(x, 4)
        a2 += 2*pow(x, 3)
        a3+=2*pow(x, 2)
        a4+=2*pow(x,1)*np.sin(2 * np.pi * f_max1 * x)
        a5+=2*pow(x,1)*np.sin(2 * np.pi * f_max2 * x)
        a6+=2*pow(x,1)
        minus+=2*pow(x,1)*f

    return [[a1, a2, a3, a4, a5, a6],minus]

def fourth_funk(xArr, fArr):
    a1 = 0;
    a2 = 0;
    a3 = 0;
    a4 = 0;
    a5 = 0;
    a6 = 0;
    minus = 0;
    for x,f in zip(xArr, fArr):
        a1 += 2*pow(x, 3) * np.sin(2 * np.pi * f_max1 * x)
        a2 += 2*pow(x, 2) * np.sin(2 * np.pi * f_max1 * x)
        a3+=2*pow(x, 1) * np.sin(2 * np.pi * f_max1 * x)
```

```
        a4+=2*np.sin(2 * np.pi * f_max1 * x) * np.sin(2 * np.pi * f_max1 * x)
        a5+=2*np.sin(2 * np.pi * f_max2 * x) * np.sin(2 * np.pi * f_max1 * x)
        a6+=2 * np.sin(2 * np.pi * f_max1 * x)
        minus+=2*np.sin(2 * np.pi * f_max1 * x)*f

    return [[a1, a2, a3, a4, a5, a6], minus]

def fifth_funk(xArr, fArr):
    a1 = 0;
    a2 = 0;
    a3 = 0;
    a4 = 0;
    a5 = 0;
    a6 = 0;
    minus = 0;
    for x,f in zip(xArr, fArr):
        a1 += 2*pow(x, 3) * np.sin(2 * np.pi * f_max2 * x)
        a2 += 2*pow(x, 2) * np.sin(2 * np.pi * f_max2 * x)
        a3+=2*pow(x, 1) * np.sin(2 * np.pi * f_max2 * x)
        a4+=2*np.sin(2 * np.pi * f_max1 * x) * np.sin(2 * np.pi * f_max2 * x)
        a5+=2*np.sin(2 * np.pi * f_max2 * x) * np.sin(2 * np.pi * f_max2 * x)
        a6+=2 * np.sin(2 * np.pi * f_max2 * x)
        minus+= 2*np.sin(2 * np.pi * f_max2 * x)*f

    return [[a1, a2, a3, a4, a5, a6], minus]

def sixth_funk(xArr, fArr):
    a1 = 0;
    a2 = 0;
    a3 = 0;
    a4 = 0;
    a5 = 0;
    a6 = 0;
    minus = 0;
    for x,f in zip(xArr, fArr):
        a1 += 2*pow(x, 3)
        a2 += 2*pow(x, 2)
        a3+=2*pow(x, 1)
        a4+=2*np.sin(2 * np.pi * f_max1 * x)
        a5+=2*np.sin(2 * np.pi * f_max2 * x)
        a6+=2
        minus+=2*f

    return [[a1, a2, a3, a4, a5, a6], minus]
```

In [51]:

```
A = np.stack((
    first_funk(X,y)[0],
    second_funk(X,y)[0],
    third_funk(X,y)[0],
    fourth_funk(X,y)[0],
    fifth_funk(X,y)[0],
    sixth_funk(X,y)[0]),
    axis=1
)

res = [
    first_funk(X,y)[1],
    second_funk(X,y)[1],
    third_funk(X,y)[1],
    fourth_funk(X,y)[1],
    fifth_funk(X,y)[1],
    sixth_funk(X,y)[1]
]

print(A)
print(res)
```

```
[[ 2.24779911e+06  5.23963542e+05  1.25625833e+05 -2.45319171e+02
  -1.67575441e+02  3.13751250e+04]
 [ 5.23963542e+05  1.25625833e+05  3.13751250e+04 -4.90652626e+01
```

```
  -3.35156508e+01   8.35835000e+03]
 [ 1.25625833e+05   3.13751250e+04   8.35835000e+03 -9.81305253e+00
  -6.70313016e+00   2.50500000e+03]
 [-2.45319171e+02 -4.90652626e+01 -9.81305253e+00   5.00000000e+02
  -4.76063633e-13   3.71822669e-13]
 [-1.67575441e+02 -3.35156508e+01 -6.70313016e+00 -4.76063633e-13
   5.00000000e+02   9.19703200e-13]
 [ 3.13751250e+04   8.35835000e+03   2.50500000e+03   3.71822669e-13
   9.19703200e-13   1.00200000e+03]]
[1508140.114532113, 336448.68048993306, 73539.60242610003, -7715.878093333481, -147.46630
99824698, 12996.774710000002]
```

In [ ]:

Розв'яємо рівняння і отримуємо коефіієнти

In [56]:

```
answ = np.linalg.solve(A, res)
tuple(answ.round(4))
```

Out[56]:

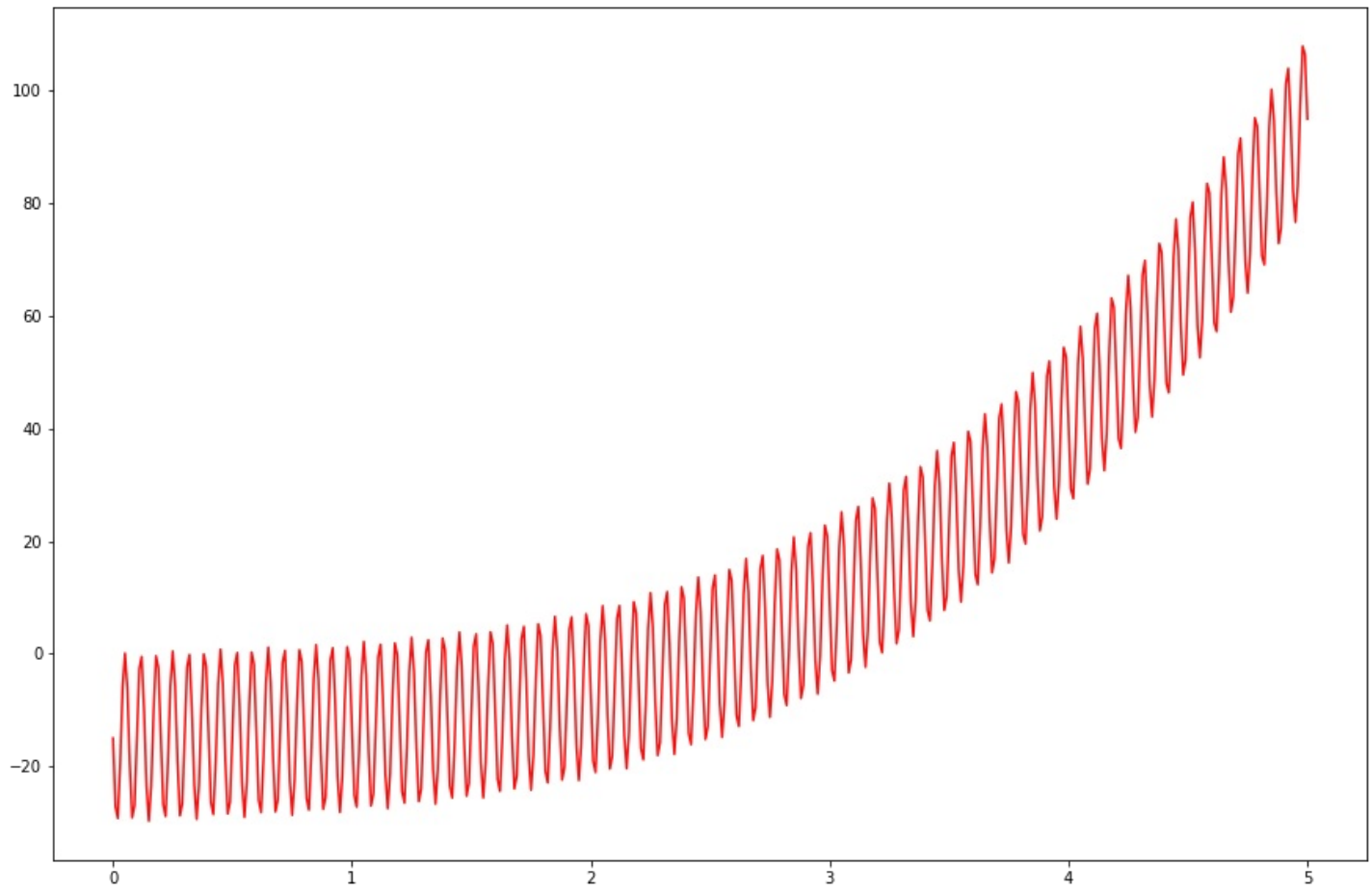
```
(1.0, -1.0, 2.0, -15.0, -0.0, -15.0)
```

In [ ]:

Підставляємо отримані коефіцінти у функцію, обчислюємо її значення:

In [61]:

```
func_approximated = np.vectorize(
 lambda t: answ[0] * t ** 3 + answ[1] * t ** 2 + answ[2] * t + answ[3] * np.sin(2 * np.
pi *
f_max1 * t) + answ[4] * np.sin(2 * np.pi * f_max2 * t) + answ[5]
)
y_approximated = func_approximated(X)
plot_graph(X, y_approximated)
```



In [ ]:

Обчислюємо квадратичну похибку функції

In [62]:

```
np.sum((y - y_approximated) ** 2)
```

Out[62]:

```
3.3864563698559513e-07
```