

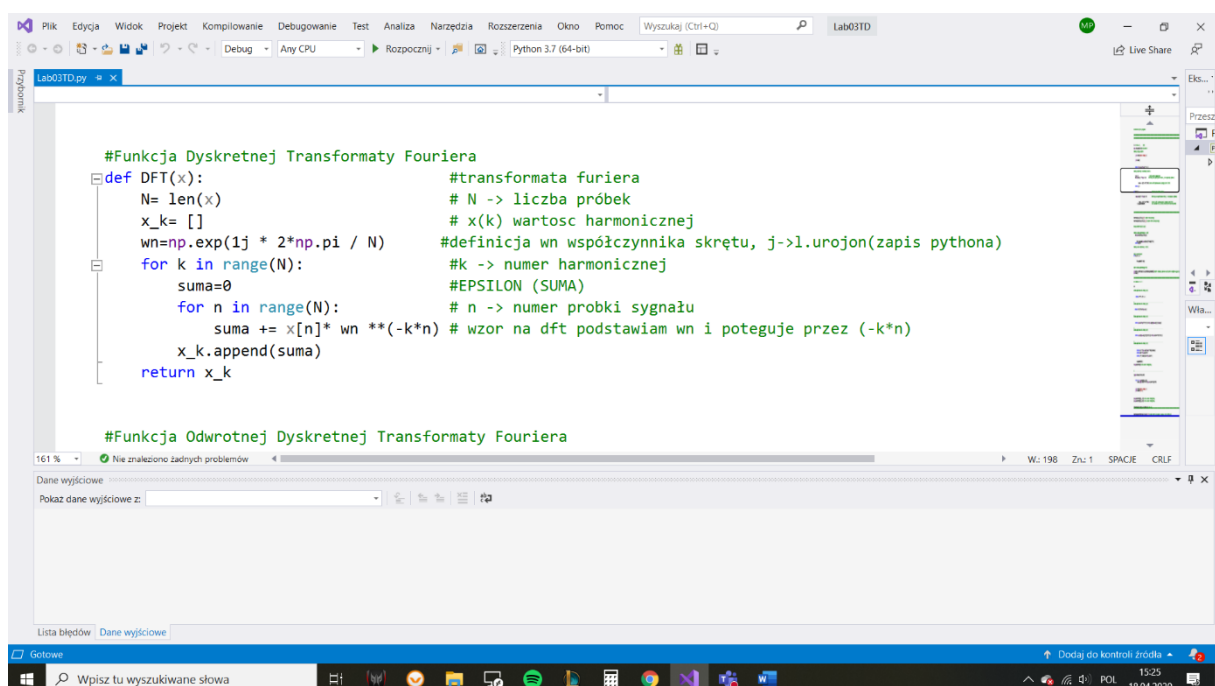
Sprawozdanie Lab03

Mateusz Proc 20C N1 nr 45123

Zadanie 1/.

Napisz funkcję realizującą Dyskretną Transformatę Fouriera.

Poniższa funkcja realizuje Dyskretną Transformatę Fouriera

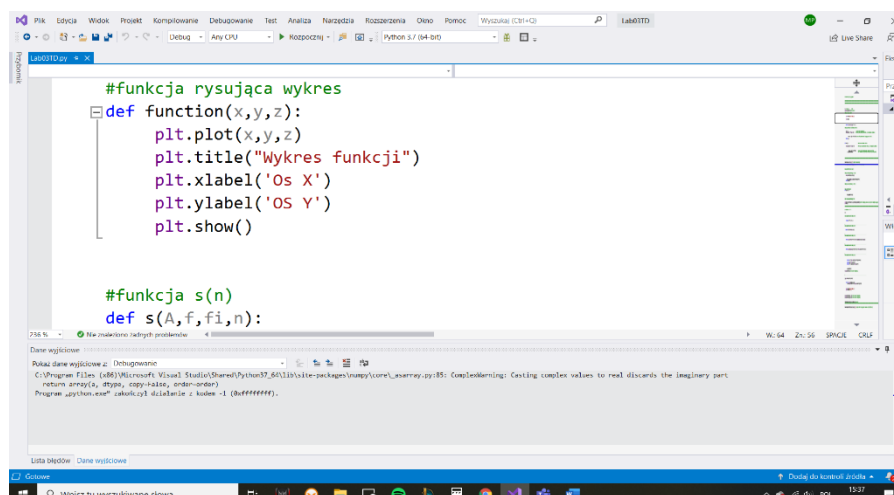


```
#Funkcja Dyskretniej Transformaty Fouriera
def DFT(x):
    N= len(x)
    x_k= []
    wn=np.exp(1j * 2*np.pi / N)
    for k in range(N):
        suma=0
        for n in range(N):
            suma += x[n]* wn **(-k*n)
        x_k.append(suma)
    return x_k

#Funkcja Odwrotnej Dyskretniej Transformaty Fouriera
```

The screenshot shows a Python IDE with a file named 'Lab03TD.py'. The code defines a function 'DFT(x)' that calculates the Discrete Fourier Transform. It uses 'len(x)' for the number of samples 'N', 'np.exp' for the complex exponential, and nested loops to calculate the sum for each frequency bin 'k'. The result is stored in a list 'x_k' and returned. A second function definition for the inverse DFT is also present but not fully visible. The IDE interface includes a menu bar, a toolbar, and a status bar at the bottom showing the system date and time.

Funkcja rysująca wykres.



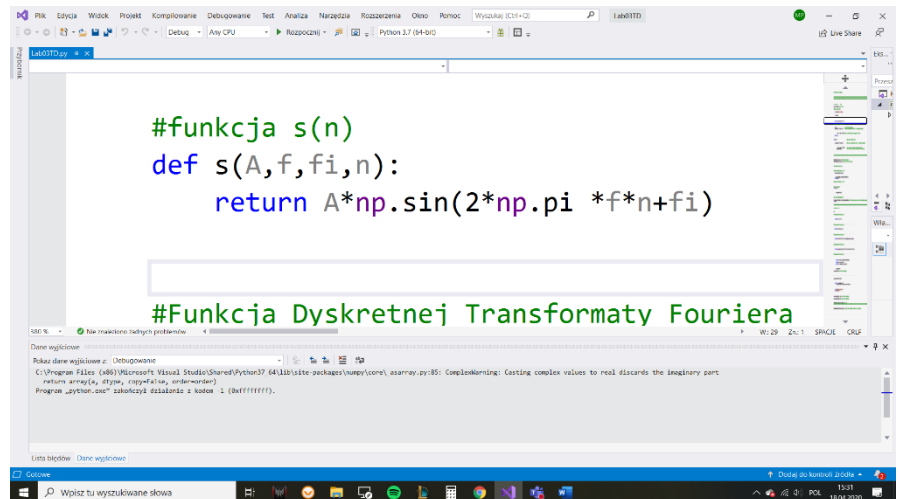
```
#funkcja rysująca wykres
def function(x,y,z):
    plt.plot(x,y,z)
    plt.title("Wykres funkcji")
    plt.xlabel('Os X')
    plt.ylabel('Os Y')
    plt.show()

#funkcja s(n)
def s(A,f,fi,n):
```

The screenshot shows a Python IDE with a file named 'Lab03TD.py'. The code defines a function 'function(x,y,z)' that uses 'plt.plot' to create a plot, 'plt.title' to set the title to 'Wykres funkcji', 'plt.xlabel' to set the x-axis label to 'Os X', 'plt.ylabel' to set the y-axis label to 'Os Y', and 'plt.show()' to display the plot. A second function definition for 's(A,f,fi,n)' is also present but not fully visible. The IDE interface includes a menu bar, a toolbar, and a status bar at the bottom showing the system date and time.

Zadanie 2/.

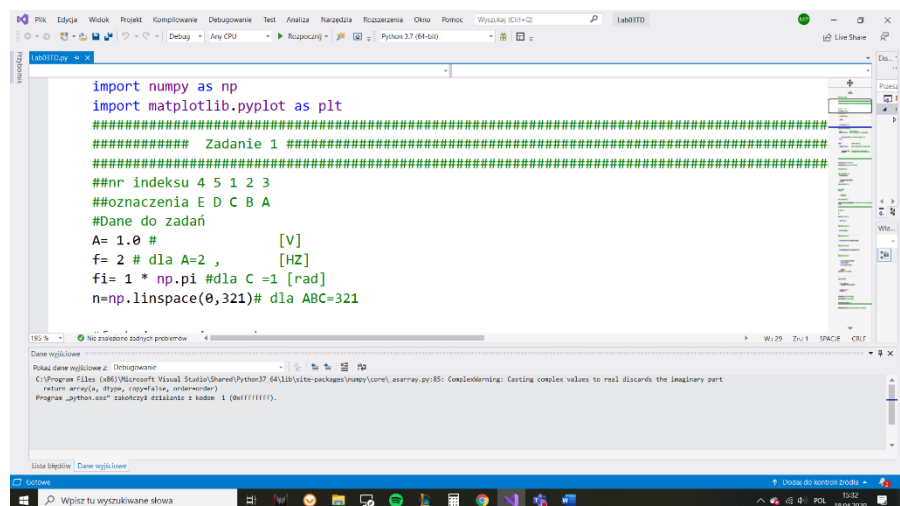
Używam funkcji z poprzednich zajęć s(n).



```
#funkcja s(n)
def s(A,f,fi,n):
    return A*np.sin(2*np.pi *f*n+fi)

#Funkcja Dyskretnej Transformaty Fouriera
```

Dla podanych parametrów.

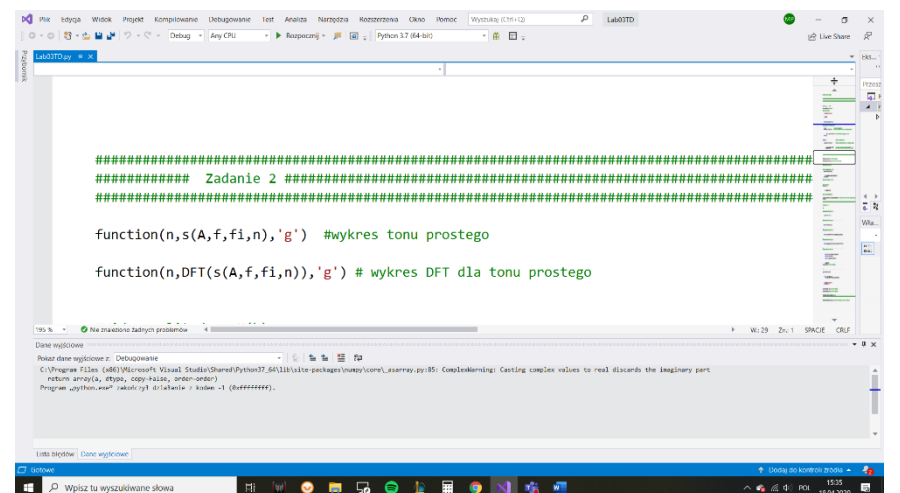


```
import numpy as np
import matplotlib.pyplot as plt

##### Zadanie 1 #####
#####nr indeksu 4 5 1 2 3#####
#####oznaczenia E D C B A#####
#Dane do zadań
A= 1.0 # [V]
f= 2 # dla A=2, [HZ]
fi= 1 * np.pi #dla C =1 [rad]
n=np.linspace(0,321)# dla ABC=321
```

Wyświetlenie wykresu dla tonu prostego z podanymi parametrami powyżej.

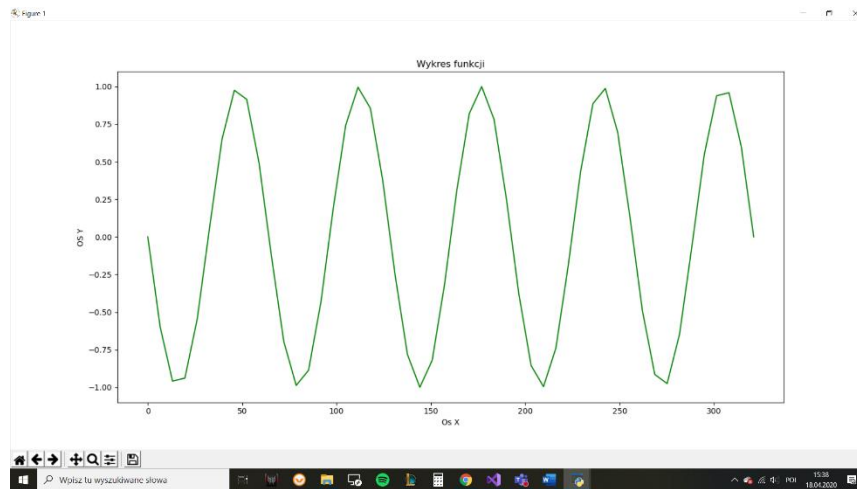
Oraz dla wykresu Dyskretnej transformaty Fouriera dla tonu prostego.



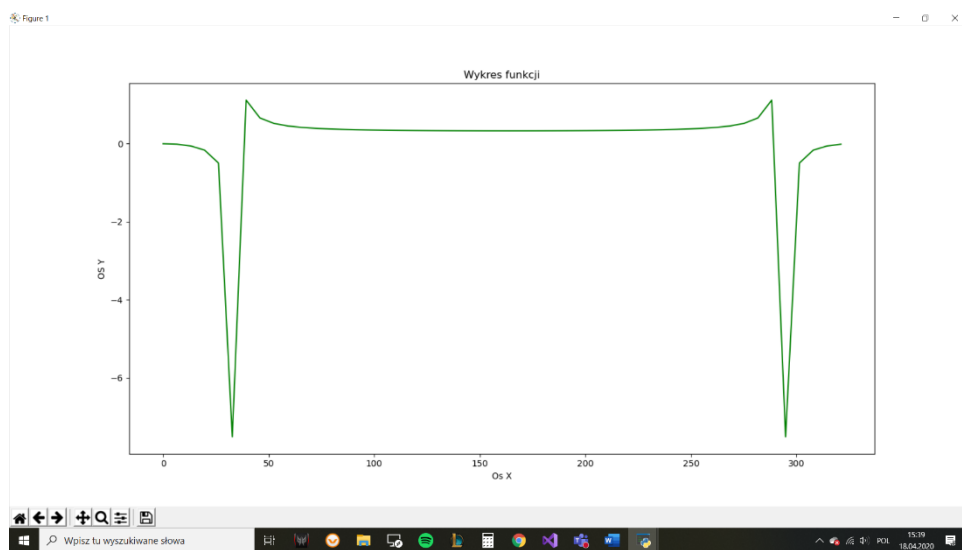
```
##### Zadanie 2 #####
#####nr indeksu 4 5 1 2 3#####
#####oznaczenia E D C B A#####

function(n,s(A,f,fi,n),'g') #wykres tonu prostego

function(n,DFT(s(A,f,fi,n)),'g') # wykres DFT dla tonu prostego
```



Wykres Dyskretnej Transformaty Fouriera dla tonu prostego



Obliczam widmo amplitudowe funkcja Mk.

The screenshot shows the Visual Studio Code editor with a Python file named 'Lab03TD.py'. The code defines a function `Mk()` that calculates the amplitude spectrum. It uses `np.real(DFT(s(A,f,fi,n)))` for the real part and `np.imag(DFT(s(A,f,fi,n)))` for the imaginary part. The amplitude spectrum `Mk` is calculated as `np.sqrt(Re**2+Im**2)`. The function `Mk()` initializes `Mk_tab` as an empty list and appends the calculated values for each frequency component.

```
#Widmo amplitudowe M(k)
#Re=np.real(DFT(s(A,f,fi,n)))
#Im=np.imag(DFT(s(A,f,fi,n)))
#Mk=np.sqrt(Re**2+Im**2)

#funkcja widma amplitudowego -> M(k)
def Mk():
    Re=np.real(DFT(s(A,f,fi,n)))
    Im=np.imag(DFT(s(A,f,fi,n)))
    Mk_tab=[]
    for i in range(0, len(Re)):
        Mk_tab.append(np.sqrt(Re[i]**2+Im[i]**2))
    return Mk_tab
```

The interface also shows a 'Dane wyjściowe' (Output) window with the following content:

```
Pokaż dane wyjściowe z: Debugowanie
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\lib\site-packages\numpy\core\_asarray.py:85
return array(a, dtype, copy=False, order=None)
```

Obliczam wartość amplitudy w skali decybelowej funkcją Mk_prim.

The screenshot shows the Visual Studio Code editor with the same Python file 'Lab03TD.py'. The code defines a function `Mk_prim(x)` that calculates the decibel amplitude. It uses `10*np.log10(x)` to convert the amplitude to decibels.

```
return Mk_tab

#funkcja skali decybelowej -> M'(k)
def Mk_prim(x): # M'(k)
    return 10*np.log10(x)
```

The interface also shows the same 'Dane wyjściowe' (Output) window as in the previous screenshot.

Wyznaczam skalę
częstotliwości funkcją Sk_cz.

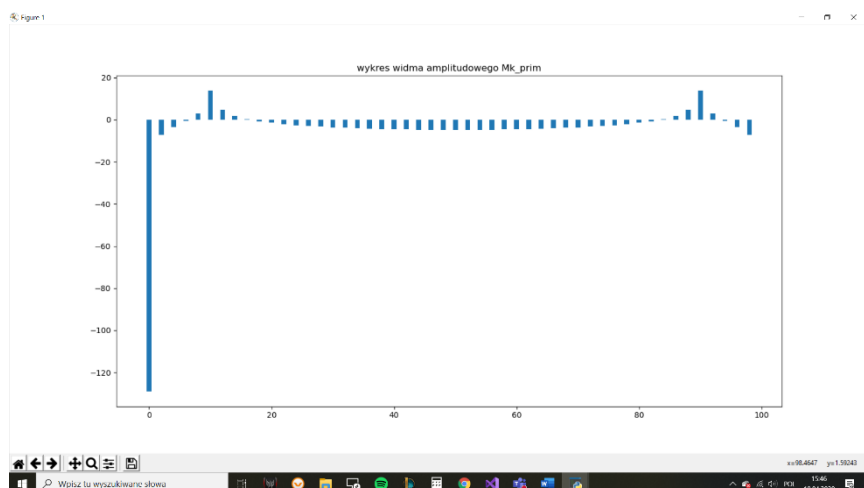
```
#Skala czestotliwosci
def sk_cz(x,fs):
    N=len(x)
    fk=[]
    for k in range(N):
        fk.append(k*(fs/N))
    return fk
```

Wykreślam widmo
amplitudowe Mk_prim.

```
#wykres widma amplitudowego M'(k)

plt.title('wykres widma amplitudowego Mk_prim')
plt.bar(sk_cz(DFT(s(A,f,fi,n)),100),Mk_prim(Mk())) #wykres s
plt.show()
```

Wykres.



Zadanie 3/.

Parametry do funkcji z poprzednich zajęć.

```
##### Zadanie 3 #####
#nr indeksu 4 5 1 2 3
#           E D C B A

a=3
b=2
c=1

#widmo amplitudowe dla funkcji x(t)
```

Widmo amplitudowe dla funkcji $x(t)$.

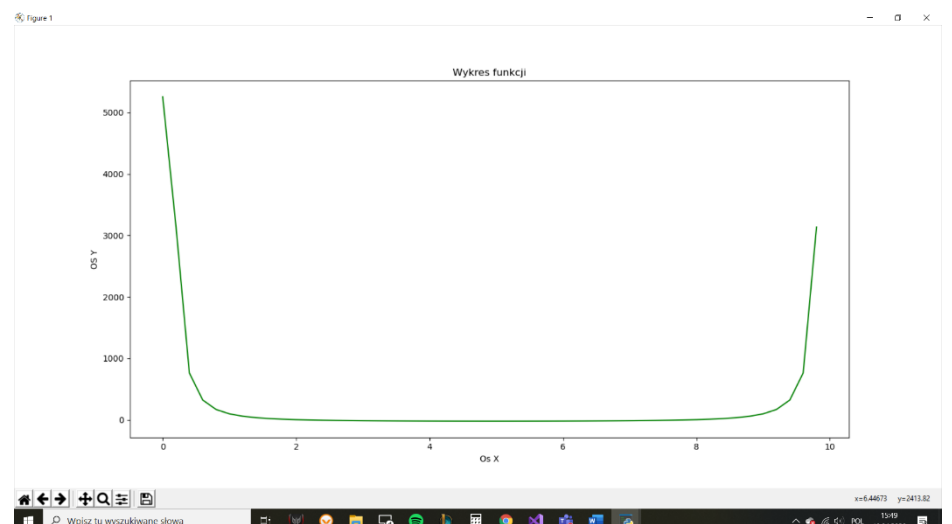
```
b=2
c=1

#widmo amplitudowe dla funkcji x(t)
t_x=np.linspace(-10,10,50)

def x(t):
    return a*t**2 + b*t + c

function(sk_cz(x(t_x),10),DFT(x(t_x)),'g')
```

Wykres.



Widmo amplitudowe dla funkcji $y(t)$.

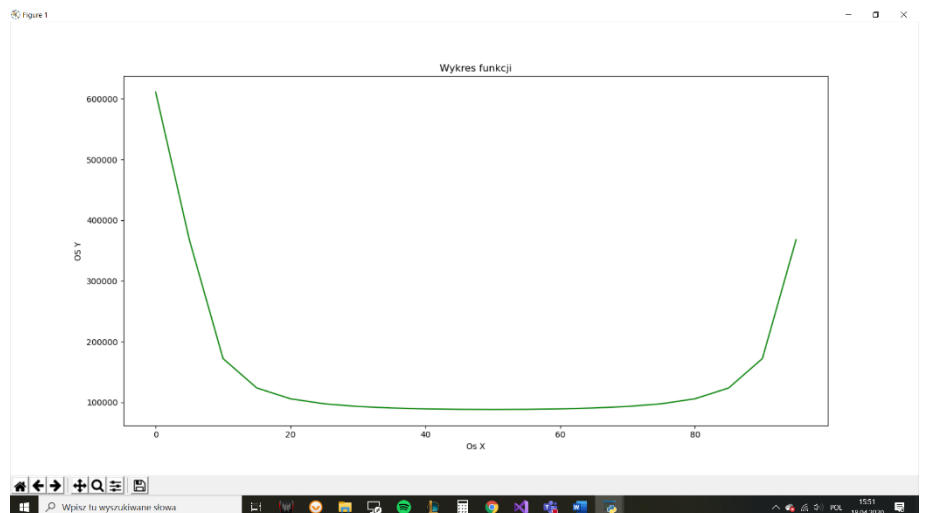
```
#widmo amplitudowe dla funkcji y(t)
t_y=np.linspace(-10,1,20)

def y(t):
    return 2*x(t)**2+12*np.cos(t)

function(sk_cz(y(t_y),100),DFT(y(t_y)),'g')

#-----widmo amplitudowe dla funkcji y(t)-----
```

Wykres.



Widmo amplitudowe dla funkcji $z(t)$.

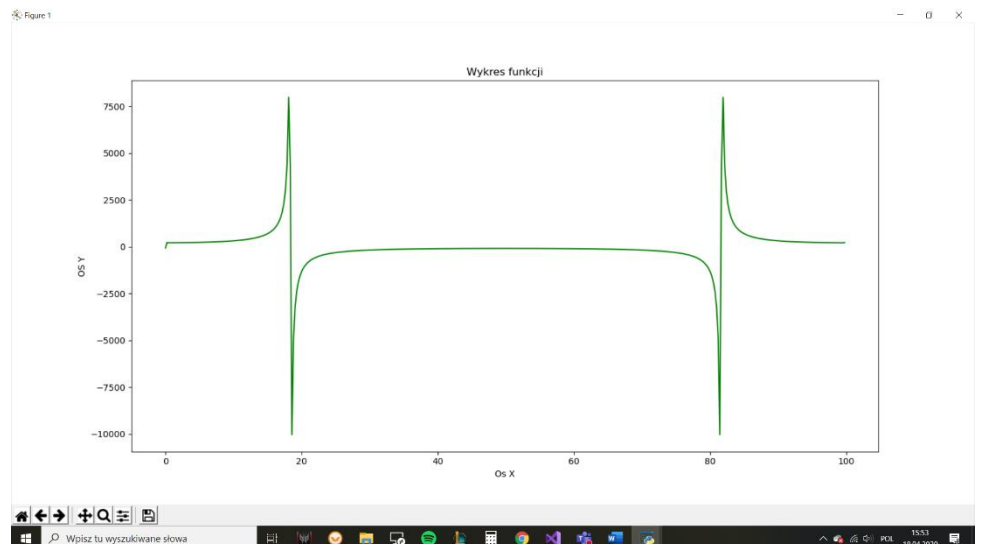
```
Lab03TD.py
#widmo amplitudowe dla funkcji z(t)
t_z=np.linspace(-10,1,420)

def z(t):
    return np.sin(2*m.pi*7*t)*x(t)-0.2*np.log10(np.abs(y(t)))+(m.pi)

function(sk_cz(z(t_z),100),DFT(z(t_z)),'g')

#-----widmo amplitudowe dla funkcji u(t)-----
```

Wykres.



Widmo amplitudowe dla funkcji $u(t)$.

```
#-----  
#widmo amplitudowe dla funkcji u(t)  
t_u=np.linspace(-10,1,420)  
  
def u(t):  
    return np.sqrt(np.abs(y(t)*y(t)*z(t)))-1.8*np.sin(0.4*t*z(t)*x(t))  
  
function(sk_cz(u(t_u),100),DFT(u(t_u)), 'g')  
  
#-----  
#widmo amplitudowe dla funkcji u(t)
```

Wykres.



Widmo amplitudowe dla funkcji $v(t)$.

```
#widmo amplitudowe dla funkcji v(t)
t_v=np.linspace(0,1,420)

def v(t):
    if (t<0.22)and(t>=0):
        return (1-7*t)*np.sin((2*m.pi*t*10)/(t+0.04))
    if(t>=0.22)and(t<0.7):
        return 0.63*t*np.sin(125*t)
    if(t<=1)and (t>=0.7):
        return (t**(-0.662))+0.77*np.sin(8*t)

o=[]
for i in t_v:
    o.append(v(i))

v1=np.abs(DFT(o)) # obl wartosc bezwzgle dna
function(sk_cz(o,100),v1,'g')
```

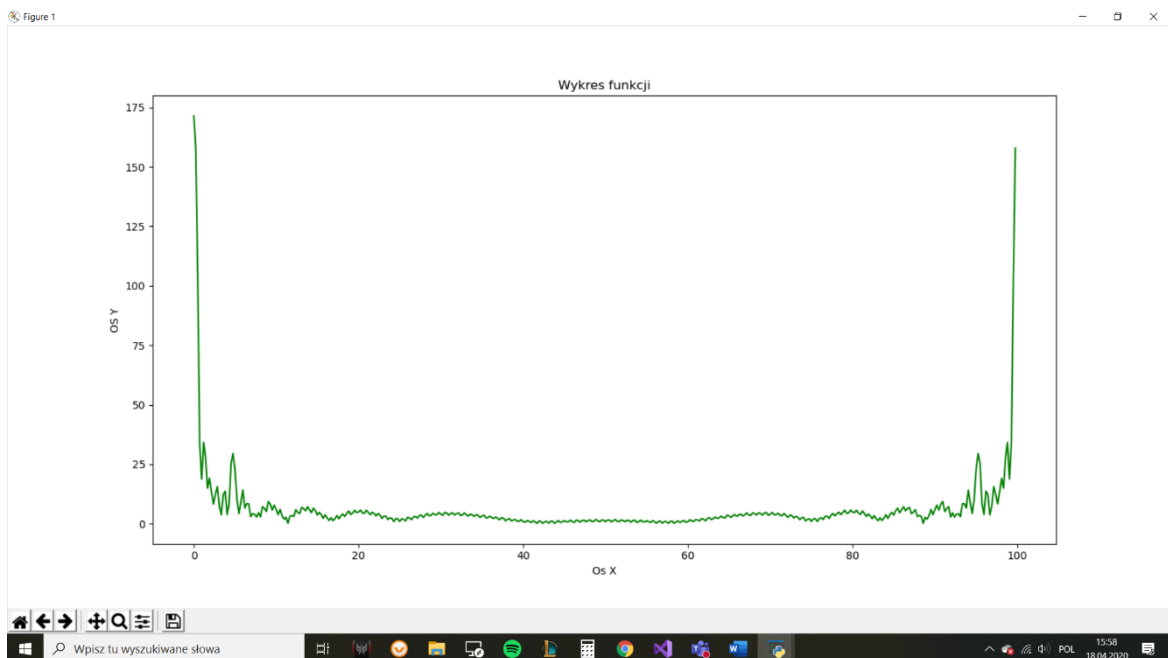
Wyrażenie kontrolne 1

Nazwa	Wartość	Typ
Dodaj element do oglądania		

Dane wyjściowe

```
return array(a, dtype, copy=False, order=order)
```

Wykres.



Widmo amplitudowe dla funkcji $p(t)$.

```
Lab03TD.py
#widmo amplitudowe dla funkcji p(t)
t_p=np.linspace(-10,1,420)

def p(t,n):
    sum = 0
    for n in np.arange(1, n+1):
        sum += ((np.cos(12*t*n**2)+np.cos(16*t*n))/n**2)
    return sum

def P_function(x,y):
    plt.plot(x,y)
    plt.title("Wykres funkcji ")
    plt.xlabel("Os X")
    plt.ylabel("Os Y")
    plt.show()

p1=np.abs(DFT(p(t_p, 2))) # obl wartosc bezwzgladna
p2=np.abs(DFT(p(t_p, 4))) # obl wartosc bezwzgladna
p3=np.abs(DFT(p(t_p, 32))) # obl wartosc bezwzgladna

P_function(sk_cz(p(t_p,2),100),p1)
P_function(sk_cz(p(t_p,4),100),p2)
P_function(sk_cz(p(t_p,32),100),p3) # AB -32
#####
```

91 % Nie znaleziono żadnych problemów

Wyrażenie kontrolne 1

Nazwa	Wartość	Typ
Dodaj element do oglądania		

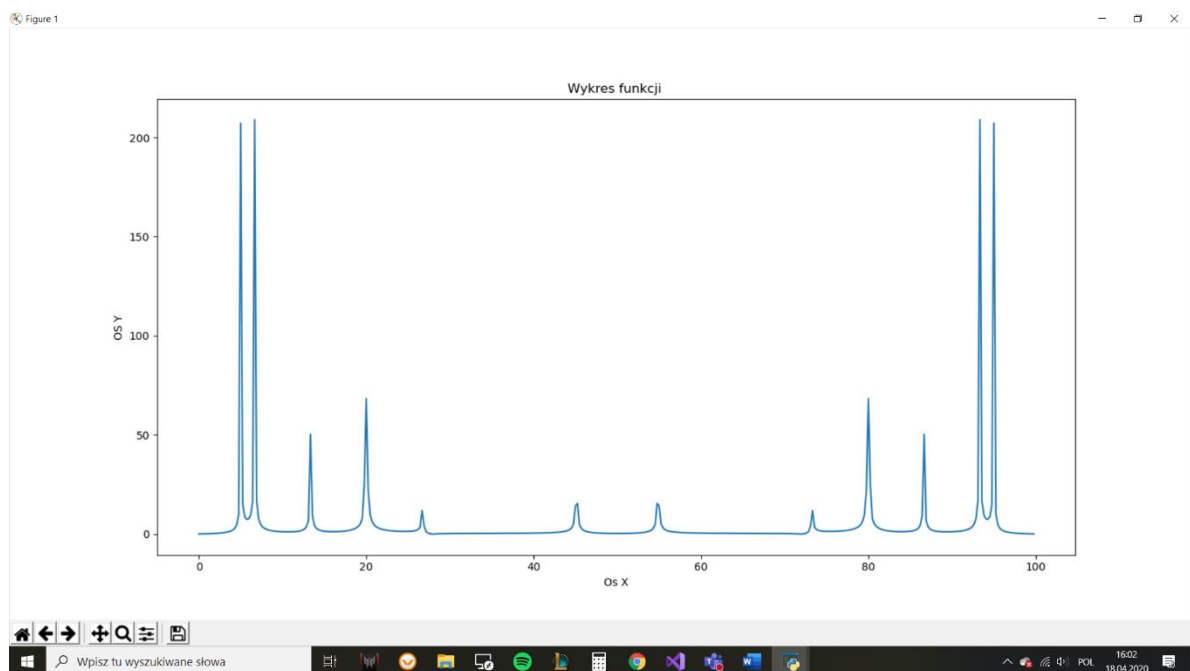
Dane wyjściowe

Pokaz dane wyjściowe z: Debugowanie

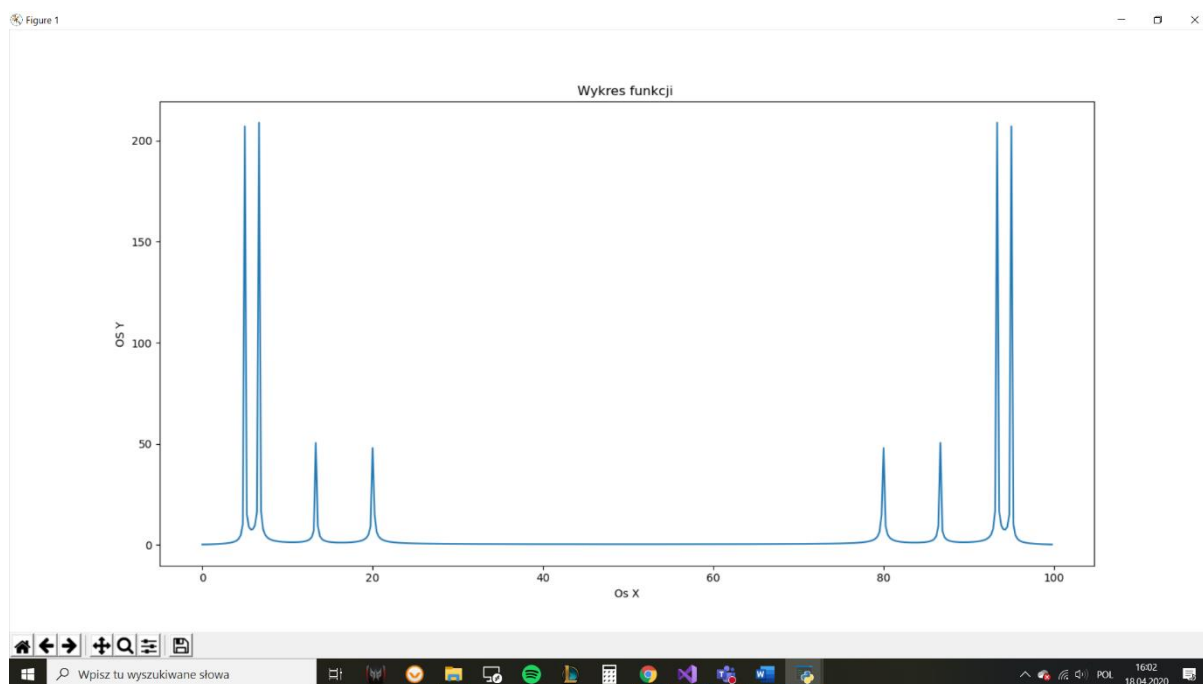
```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\lib\site-packages\numpy\core\_asarray.py:85
return array(a, dtype, copy=False, order=order)
```

Wykresy.

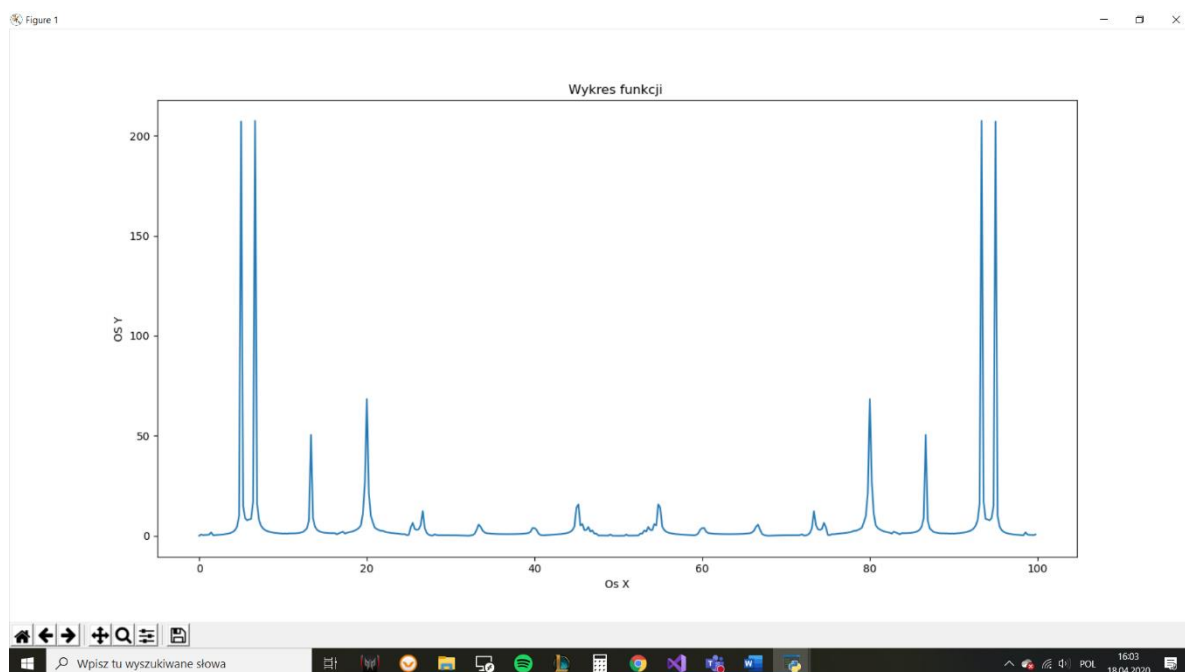
1/.



2/.

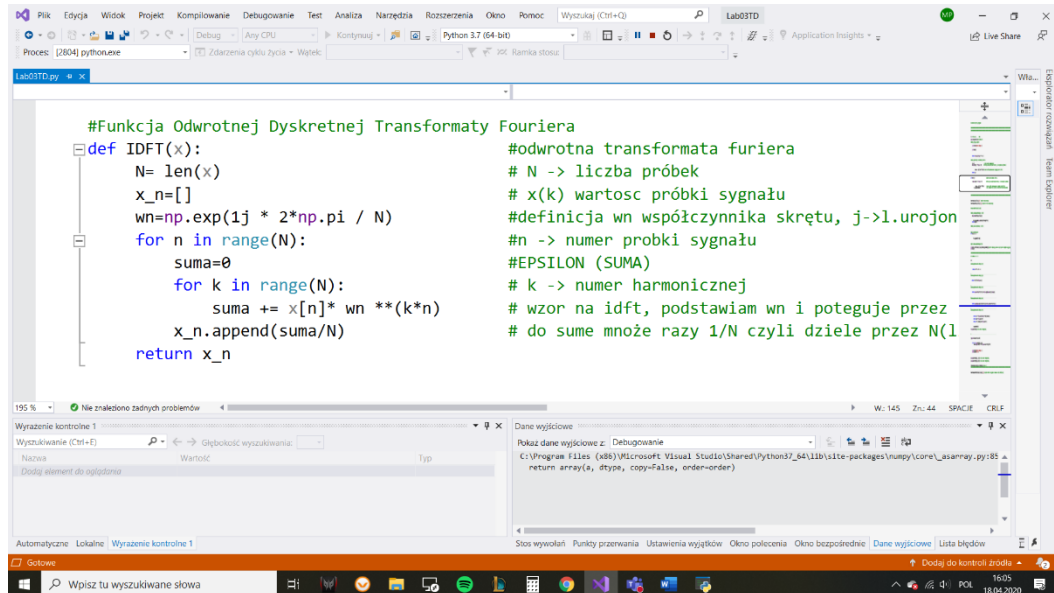


3/.



Zadanie 4/.

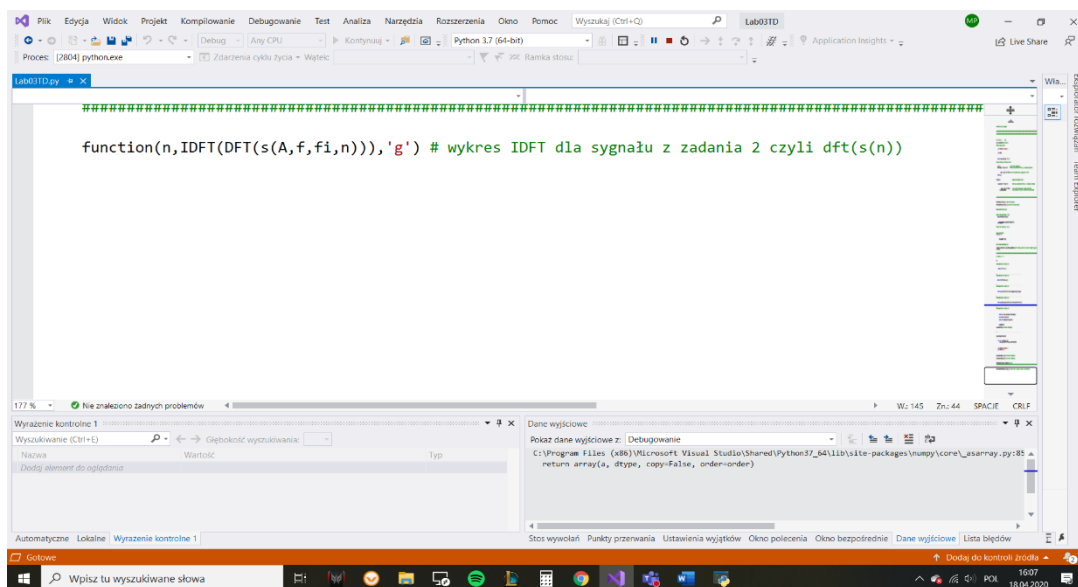
Funkcja realizująca Odwrotną Dyskretną Transformatę Fouriera.



```
#Funkcja Odwrotnej Dyskretnej Transformaty Fouriera
def IDFT(x):
    N= len(x)
    x_n=[]
    wn=np.exp(1j * 2*np.pi / N)
    for n in range(N):
        suma=0
        for k in range(N):
            suma += x[k]* wn **(k*n)
        x_n.append(suma/N)
    return x_n
```

The screenshot shows the Visual Studio Code editor with a Python file named 'Lab03TD.py'. The code implements the IDFT function. Comments in Polish explain the variables: N is the number of samples, x(k) is the signal value, wn is the complex exponential coefficient, n is the sample number, EPSILON (SUMA) is the sum, k is the harmonic number, and the formula for IDFT is given. The code uses NumPy for complex numbers and array operations. The interface also shows a 'Dane wyjściowe' (Output) window with the NumPy array definition and a 'Wyrażenie kontrolne 1' (Test 1) window.

Zweryfikuj poprawność jej działania odwracając sygnał z dziedziny częstotliwości do dziedziny czasu (wykorzystaj sygnał użyty w zadaniu drugim).



```
function(n,IDFT(DFT(s(A,f,fi,n))), 'g') # wykres IDFT dla sygnału z zadania 2 czyli dft(s(n))
```

The screenshot shows the same Visual Studio Code environment. The code now includes a function call to test the IDFT implementation. The function 'function(n, IDFT(DFT(s(A, f, fi, n))), 'g')' is intended to plot the IDFT of the signal from the previous task. The interface shows the same 'Dane wyjściowe' and 'Wyrażenie kontrolne 1' windows.

Wykres.

