main

December 28, 2022

1

- 04-105

[ ]:
```
import matplotlib.pyplot as plt
import ipywidgets
import numpy as np
import pandas as pd
import scipy as sp
from scipy.io import wavfile
from scipy import signal
from scipy.signal import periodogram as periodogram_f
from scipy.fft import fftfreq, fftshift
from scipy.fft import fft, ifft, fft2, ifft2
import IPython
```

1.0.1 RC

10 . ,

.

:

$$U_{out} = U_{in}\frac{X_c}{\sqrt{R^2 + X_C^2}}\ ,X_C = \frac{1}{2\pi\nu C}$$

:

$$f_{cutoff} = \frac{1}{2\pi RC}$$

:

[ ]:
```
#                  ,

def plotAFR(Resis = 0.1,C = 100):
    fig,ax = plt.subplots(figsize = (16,9))
    plt.grid()
    plt.xlabel("     $ $")
    plt.ylabel("     ")
    plt.xlim(20,22000)
    plt.xticks(np.arange(20,22000,1000))
    plt.ylim(0,1)
    R = Resis*1000
    cutoff = 1/(2*np.pi * R* C)
    freq = np.linspace(20,22000,22000) #
    X_c = 1/(2*np.pi*freq*(C*np.power(0.1,9)))
    amp = X_c/np.sqrt(np.power(R,2)+np.power(X_c,2))
    plt.plot(freq,amp,color = 'blue')
ipywidgets.interact(plotAFR,Resis = (0.1,10,0.2),C = (0.1,500,5))
```

```
interactive(children=(FloatSlider(value=0.1, description='Resis', max=10.0,␣
 ↪min=0.1, step=0.2), FloatSlider(va…
```

[ ]:
```
<function __main__.plotAFR(Resis=0.1, C=100)>
```

<> $50\ nF$. , - $0.1\ \mu F$. LPF, HPF.

[ ]:
```
C: float = 0.1 * np.power(0.1,6) #
```

1.0.2

:

$$k = 1 + \frac{100}{4.7} = 22.3$$

<>,

$96\ dB$.

[ ]:
```
ampl_k = 22.3
```

-

[ ]:
```
def getAfr(freq: np.ndarray,R_hpf: float,R_lpf: float) -> np.ndarray:
    R1 = R_hpf*1000
    R2 = R_lpf*1000
    X_c = 1/(2*np.pi*freq*C)
    A_hpf = R1/np.sqrt(np.power(X_c,2)+np.power(R1,2))
    A_lpf = X_c/np.sqrt(np.power(X_c,2)+np.power(R2,2))
    A_result = A_hpf*A_lpf
    return A_result
```

[ ]:
```
def plotAFR(R_hpf = 1, R_lpf = 1):
    fig, ax = plt.subplots(figsize = (16,9))
    plt.grid()
    plt.xlabel("     $ $")
    plt.ylabel("     ")
    plt.xlim(20,22000)
    plt.xticks(np.arange(20,22000,1000))
    plt.ylim(0,int(ampl_k))
    ####
    R1 = R_hpf*1000
    R2 = R_lpf*1000
    freq = np.linspace(20,22000,20000)
    X_c = 1/(2*np.pi*freq*C)
    A_hpf = R1/np.sqrt(np.power(X_c,2)+np.power(R1,2))
    A_lpf = X_c/np.sqrt(np.power(X_c,2)+np.power(R2,2))
    A_result = ampl_k*A_hpf*A_lpf
    plt.plot(freq,A_hpf,label = "               ")
    plt.plot(freq,A_lpf,label = '               ')
    plt.plot(freq,A_result,label = '      ',color = "red")
    plt.title("       -                  ")
    plt.legend()
```

```
[ ]: ipywidgets.interact(plotAFR,R_hpf = (0.1,10,0.2),R_lpf = (0.1,10,0.2))
```

```
interactive(children=(FloatSlider(value=1.0, description='R_hpf', max=10.0,␣
↪min=0.1, step=0.2), FloatSlider(va…
```

```
[ ]: <function __main__.plotAFR(R_hpf=1, R_lpf=1)>
```

- :

$$F = \frac{1}{2\pi C\sqrt{R_{lpf}}\sqrt{R_{hpf}}}$$

```
[ ]: def getLpfPhase_shift(freq,R_lpf):
         phase_shift = - np.arctan(2*np.pi*freq*R_lpf*C*1000)
         return phase_shift
     def getHpfPhase_shift(freq,R_hpf):
         phase_shift = np.arctan(1/(2*np.pi*freq*R_hpf*C*1000))
         return phase_shift
     def getPhaseShift(freq,R_hpf,R_lpf):
         return getLpfPhase_shift(freq,R_lpf)+getHpfPhase_shift(freq,R_hpf)
     def getMaxAmpFreq(R_hpf,R_lpf):
         return 1/(2*np.pi*C*np.sqrt(R_hpf*1000)*np.sqrt(R_lpf*1000))
```

```
[ ]: def multiple_formatter(denominator=2, number=np.pi, latex='\pi'):
         def gcd(a, b):
             while b:
                 a, b = b, a%b
             return a
         def _multiple_formatter(x, pos):
             den = denominator
             num = int(np.rint(den*x/number))
             com = gcd(num,den)
             (num,den) = (int(num/com),int(den/com))
             if den==1:
                 if num==0:
```

```
                return r'$0$'
            if num==1:
                return r'$%s$'%latex
            elif num==-1:
                return r'$-%s$'%latex
            else:
                return r'$%s%s$'%(num,latex)
        else:
            if num==1:
                return r'$\frac{%s}{%s}$'%(latex,den)
            elif num==-1:
                return r'$\frac{-%s}{%s}$'%(latex,den)
            else:
                return r'$\frac{%s%s}{%s}$'%(num,latex,den)
    return _multiple_formatter

def PFR(R_hpf = 1,R_lpf = 1):
    fig, ax = plt.subplots(figsize = (16,9))
    plt.ylim(-np.pi/2,np.pi/2)
    plt.xlabel("     $ $")
    plt.ylabel("       $   $")
    plt.xlim(20,22000)
    plt.yticks(np.arange(-np.pi/2,np.pi/2+np.pi/12,np.pi/12))
    plt.xticks(np.arange(20,22000,1000))
    freq = np.linspace(20,22000,20000)
    plt.plot(freq,getHpfPhase_shift(freq,R_hpf),label = "            ")
    plt.plot(freq,getLpfPhase_shift(freq,R_lpf),label = "           ")
    plt.plot(freq,getPhaseShift(freq,R_hpf,R_lpf),label = "     ", color =␣
 ↪'red')
    plt.title("   -              ")
    ax.axhline(0, color='black', lw=2)
    maxAmp = getMaxAmpFreq(R_hpf,R_lpf)
    ax.axvline(maxAmp,color = 'black',lw = 2)
    ax.yaxis.set_major_locator(plt.MultipleLocator(np.pi / 2))
    ax.yaxis.set_minor_locator(plt.MultipleLocator(np.pi / 12))
    ax.grid(True)
    print("               : "+str(maxAmp))
    ax.yaxis.set_major_formatter(plt.FuncFormatter(multiple_formatter()))
    plt.legend()
ipywidgets.interact(PFR,R_hpf = (0.1,10,0.1),R_lpf = (0.1,10,0.1))
```

```
interactive(children=(FloatSlider(value=1.0, description='R_hpf', max=10.0,␣
↪min=0.1), FloatSlider(value=1.0, d…
```

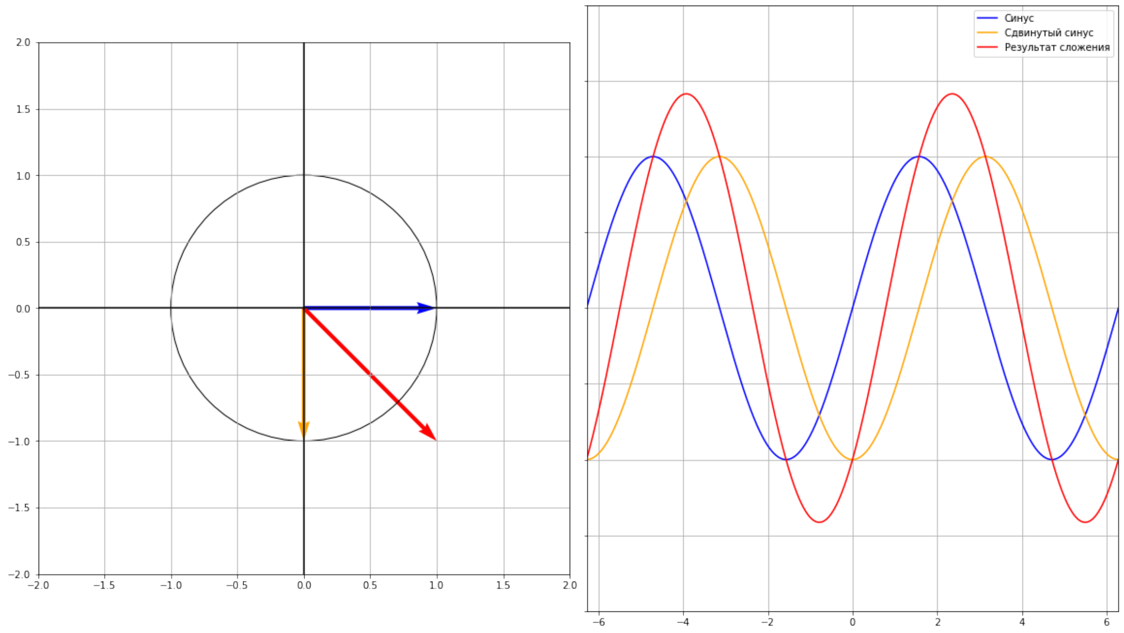
```
[ ]: <function __main__.PFR(R_hpf=1, R_lpf=1)>
```

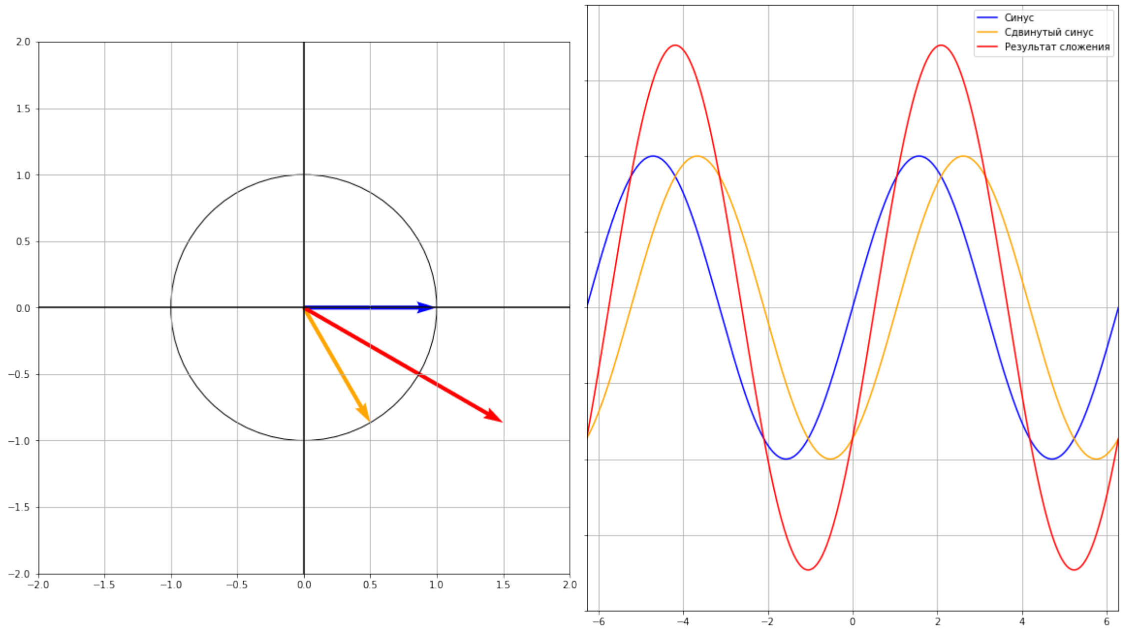
[ ]:

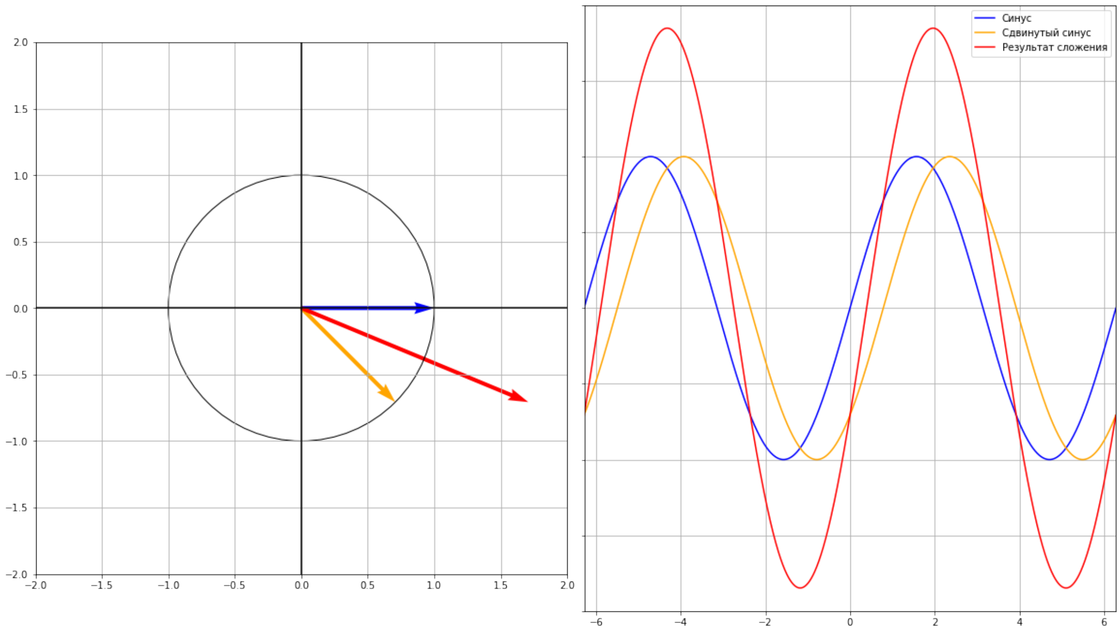
```
def plotShift(shift: float):
    fig = plt.figure(figsize = (16,9),constrained_layout = True)
    gs = fig.add_gridspec(1, 2, hspace=0, wspace=0)
    (ax1,ax2) = gs.subplots(sharex='col', sharey='row')
    eyeCircle = plt.Circle((0, 0), 1, color='black',fill = False)
    ax1.set_xlim(-2,2)
    ax1.set_ylim(-2,2)
    ax1.axvline(0,color = 'black')
    ax1.axhline(0,color = 'black')
    ax1.grid(True)
    # ax2.set_aspect('equal')
    ax1.set_aspect('equal')
    ax1.add_patch(eyeCircle)
    V = np.array([[1,0], [np.cos(shift),np.sin(shift)], [1+np.cos(shift),np.
↪sin(shift)]])
    origin = np.array([[0, 0, 0],[0, 0, 0]]) # origin point

    ax1.quiver(*origin, V[:,0], V[:,1],␣
↪color=['b','orange','r'],scale_units='xy', scale=1)
    ax2.set_xlim(-np.pi*2,2*np.pi)
    ax2.set_ylim(-2,2)
    xAxes = np.linspace(-np.pi*2,np.pi*2,1000)
    ax2.plot(xAxes,np.sin(xAxes),color = 'blue',label = "   ")
    ax2.plot(xAxes,np.sin(xAxes+shift),color = 'orange',label = "         ")
    ax2.plot(xAxes,np.sin(xAxes+shift)+np.sin(xAxes),color = 'red',label =␣
↪'          ')
    ax2.legend()
    ax2.grid(True)
    plt.show()
```

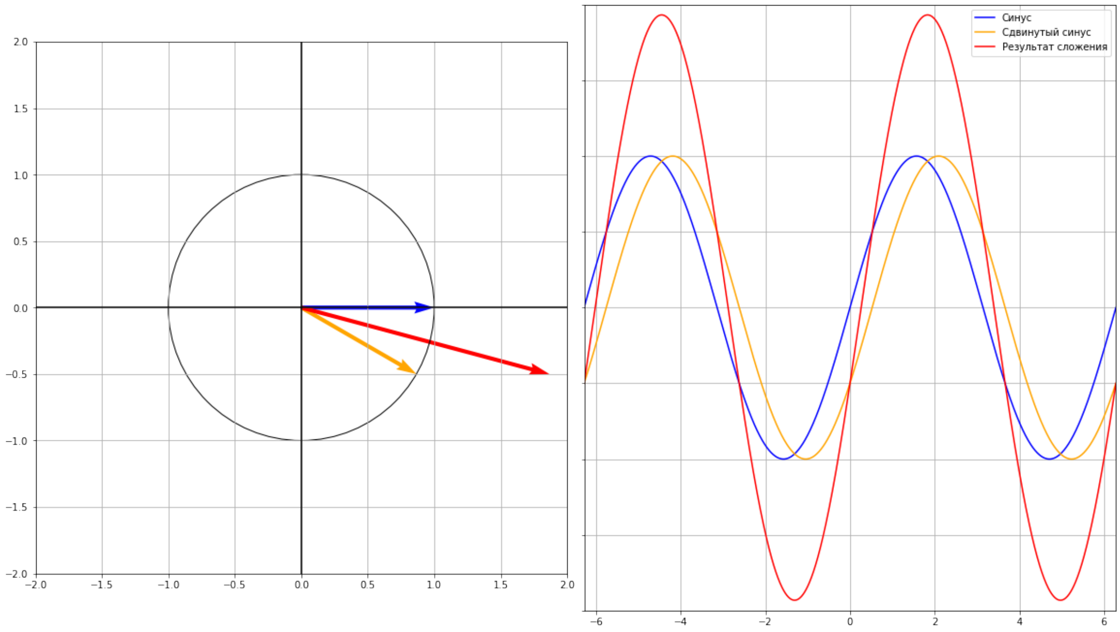
[ ]:

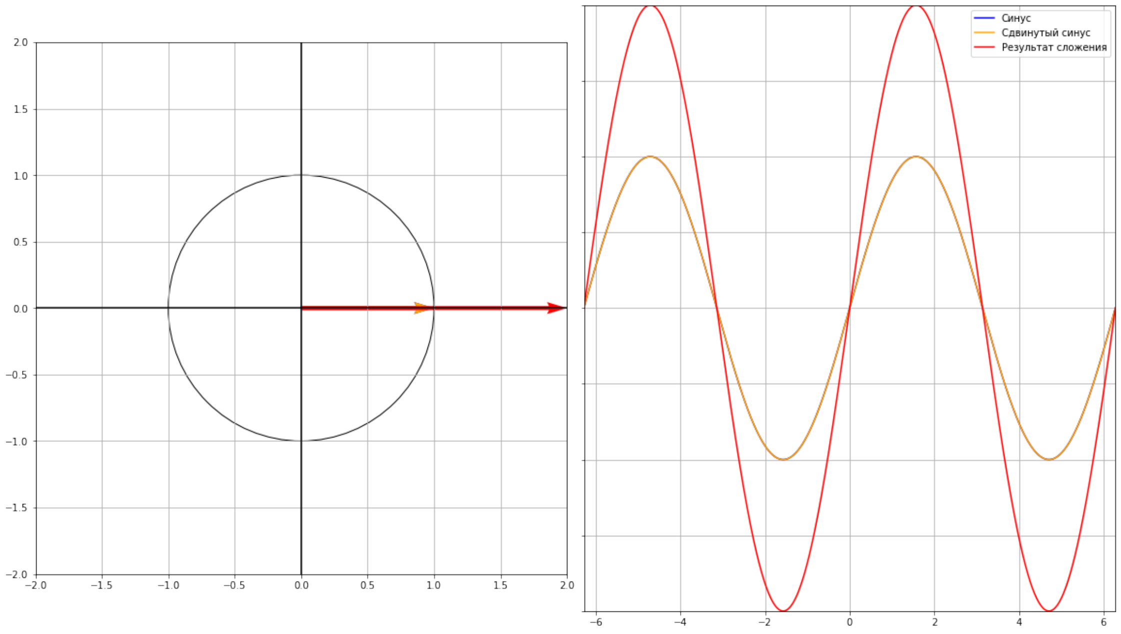
```
shifts = [3*np.pi/2,5*np.pi/3,7*np.pi/4,11*np.pi/6,0,np.pi/6,np.pi/4,np.pi/3,np.
↪pi/2]
for i in shifts:
    plotShift(i)
```

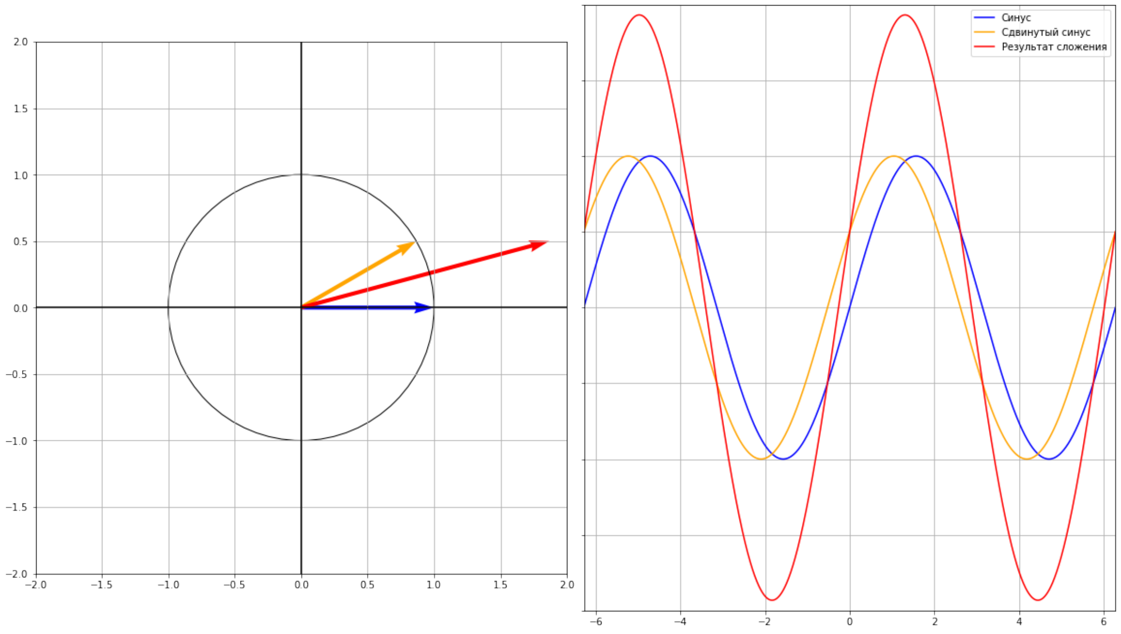

Синус
Сдвинутый синус
Результат сложения
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
−6
−4
−2
0
2
4
6

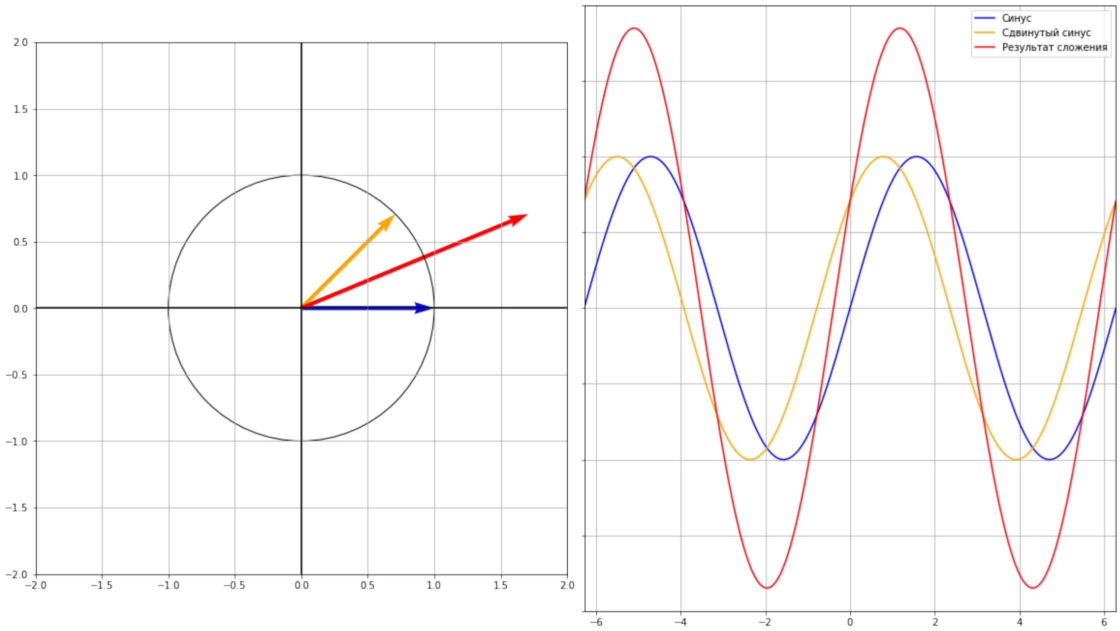

Синус
Сдвинутый синус
Результат сложения
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
−6
−4
−2
0
2
4
6

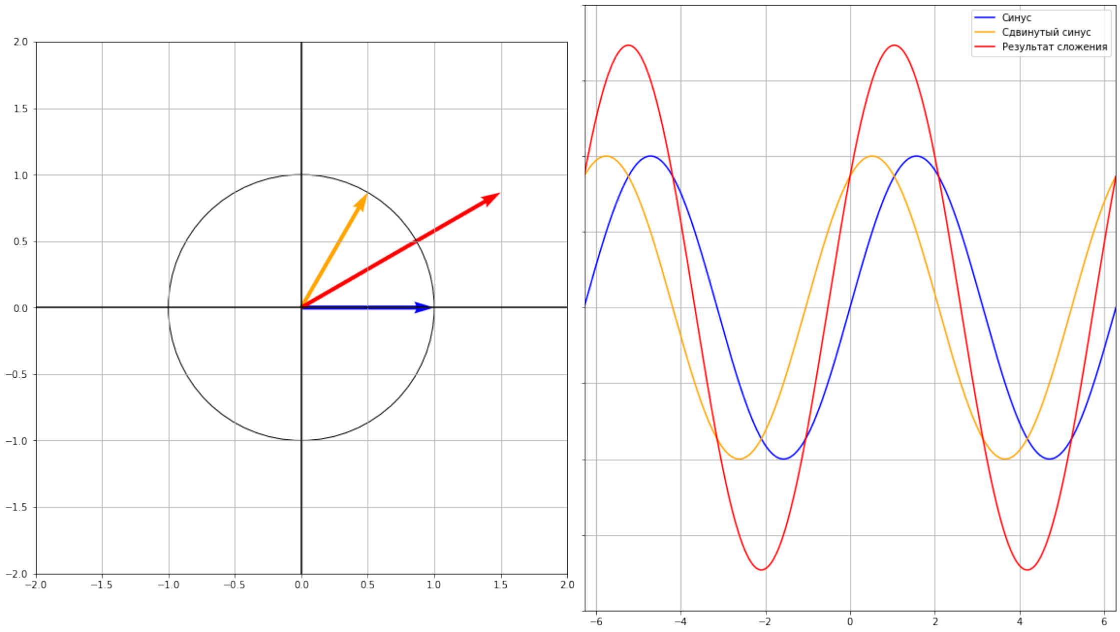

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Синус
Сдвинутый синус
Результат сложения
−6
−4
−2
0
2
4
6

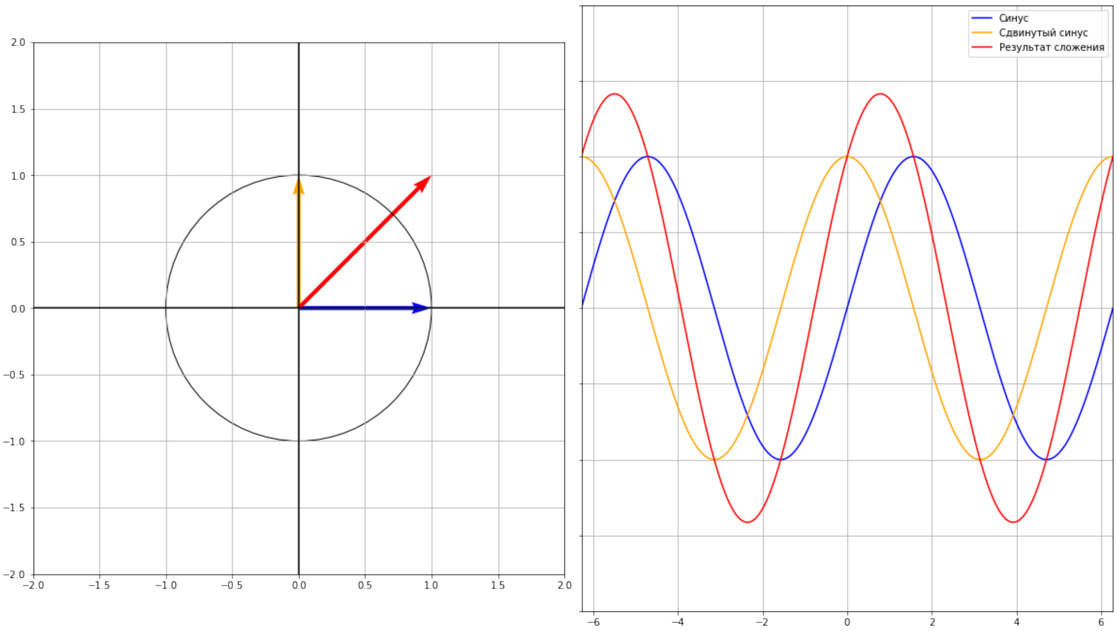

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Синус
Сдвинутый синус
Результат сложения
−6
−4
−2
0
2
4
6


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Синус
Сдвинутый синус
Результат сложения
−6
−4
−2
0
2
4
6


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Синус
Сдвинутый синус
Результат сложения
−6
−4
−2
0
2
4
6


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
Синус
Сдвинутый синус
Результат сложения
−6 −4 −2 0 2 4 6


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0
Синус
Сдвинутый синус
Результат сложения
−6 −4 −2 0 2 4 6



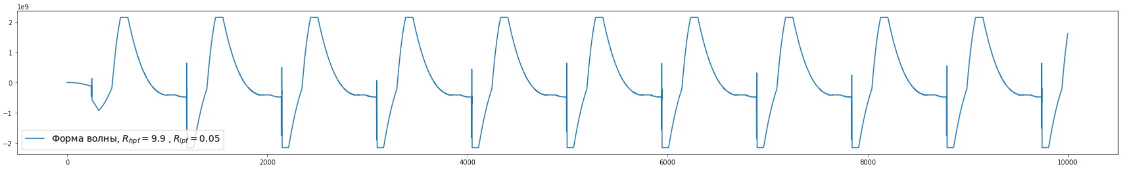
1.0.3 1 - <>

1.1 -

[ ]:
```
rate, data = sp.io.wavfile.read('DR0000_0229.wav')
data=data[:,0]
time = np.arange(0, len(data), 1)/rate
R_hpf,R_lpf = 9.9,0.05
```

```
C:\Users\     \AppData\Local\Temp\ipykernel_15004\1130674698.py:1:
WavFileWarning: Chunk (non-data) not understood, skipping it.
  rate, data = sp.io.wavfile.read('DR0000_0229.wav')
```

[ ]:
```
plt.figure(figsize=(30,4))
plt.plot(data[0:10000],label = "       , $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf} =␣
  ↪"+str(R_lpf)+"$")
plt.legend(fontsize = 14)
plt.show()
```



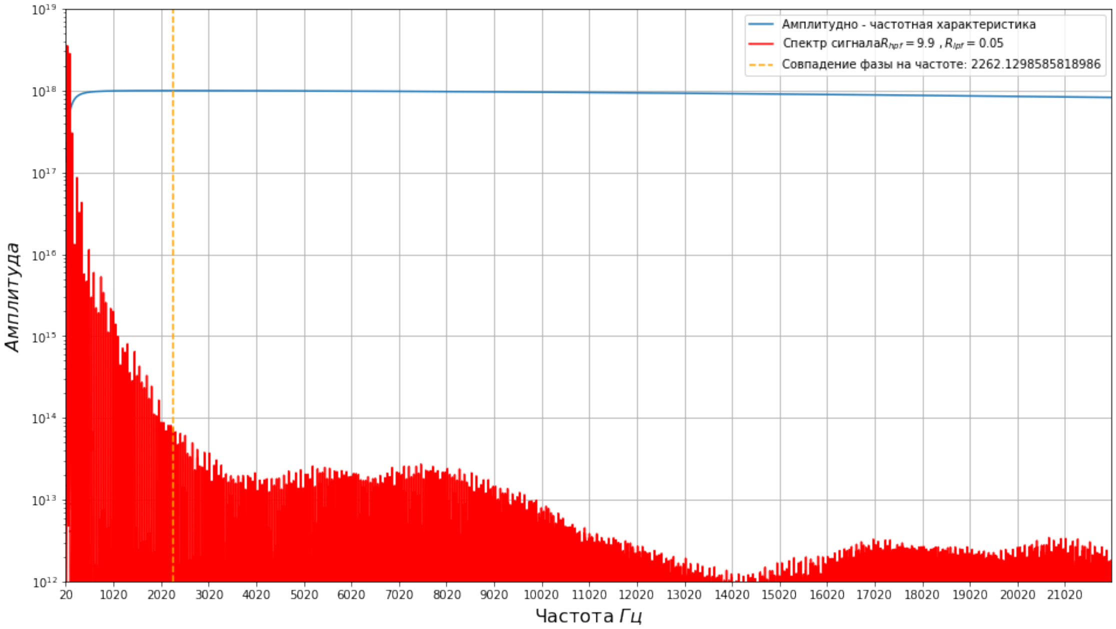
[ ]:
```
IPython.display.Audio("DR0000_0229.wav")
```

```
[ ]: <IPython.lib.display.Audio object>
```

```
[ ]: periodogram = np.abs(fft(data))**2 / (rate * len(data))
     frequencies = fftfreq(len(data), d=1/rate)
     frequencies
```

```
[ ]: array([ 0.        ,  0.06624271,  0.13248543, …, -0.19872814,
            -0.13248543, -0.06624271])
```

```
[ ]: freq = np.linspace(22,22000,20000)
     amp = 1e18
     plt.figure(figsize=(16,9))
     plt.plot(freq,getAfr(freq,R_hpf,R_lpf)*amp,label = '       -               ')
     plt.plot(fftshift(frequencies), fftshift(periodogram), color='red',label = "   ␣
      ↪    $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf} = "+str(R_lpf)+"$")
     plt.xlim(22,22000)
     plt.axvline(x=getMaxAmpFreq(R_hpf=R_hpf,R_lpf=R_lpf),
                 label = "               :␣
      ↪"+str(getMaxAmpFreq(R_hpf=R_hpf,R_lpf=R_lpf)),
                 linestyle = 'dashed',
                 color = 'orange')
     plt.ylim((1e12,1e19))
     plt.xlabel('     $ $', fontsize = 16)
     plt.ylabel('$     $',fontsize = 16)
     plt.legend()
     plt.grid()
     plt.semilogy()
     plt.xticks(np.arange(20,22000,1000))
     plt.show()
```

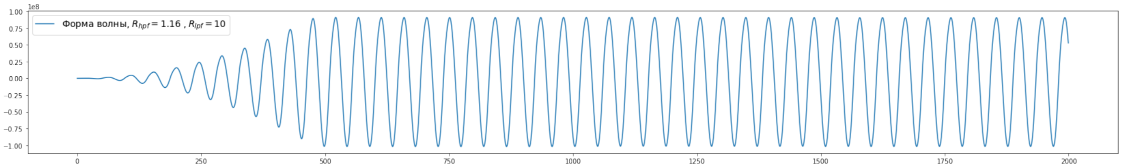


1.2 - -

[ ]:
```
rate, data = sp.io.wavfile.read('DR0000_0230.wav')
data=data[:,0]
time = np.arange(0, len(data), 1)/rate
R_hpf,R_lpf = 1.16,10
```

```
C:\Users\     \AppData\Local\Temp\ipykernel_15004\3860120285.py:1:
WavFileWarning: Chunk (non-data) not understood, skipping it.
  rate, data = sp.io.wavfile.read('DR0000_0230.wav')
```

[ ]:
```
plt.figure(figsize=(30,4))
plt.plot(data[:2000],label = "       , $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf} =␣
  ↪"+str(R_lpf)+"$")
plt.legend(fontsize = 14)
plt.show()
```



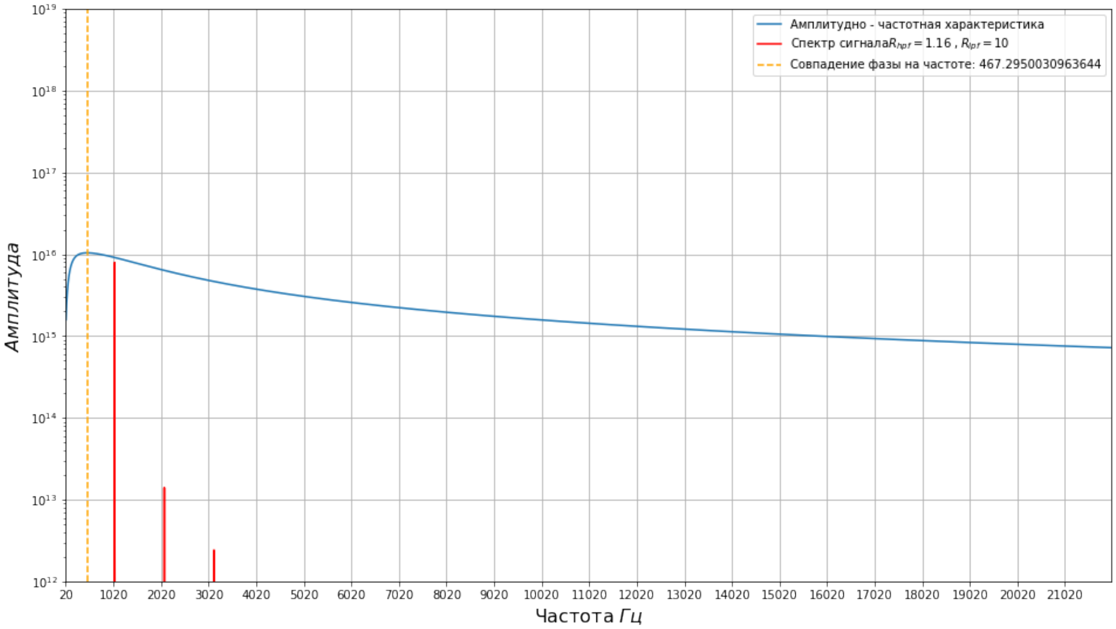
[ ]:
```
IPython.display.Audio("DR0000_0230.wav")
```

```
[ ]: <IPython.lib.display.Audio object>
```

```
[ ]: periodogram = np.abs(fft(data))**2 / (rate * len(data))
     frequencies = fftfreq(len(data), d=1/rate)
     frequencies
```

```
[ ]: array([ 0.        ,  0.1393534,  0.2787068, …, -0.4180602, -0.2787068,
            -0.1393534])
```

```
[ ]: freq = np.linspace(22,22000,20000)
     amp = 1e17
     plt.figure(figsize=(16,9))
     plt.plot(freq,getAfr(freq,R_hpf,R_lpf)*amp,label = '       -               ')
     plt.plot(fftshift(frequencies), fftshift(periodogram), color='red',label = "   ␣
      ↪    $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf} = "+str(R_lpf)+"$")
     plt.axvline(x=getMaxAmpFreq(R_hpf=R_hpf,R_lpf=R_lpf),
                 label = "               :␣
      ↪"+str(getMaxAmpFreq(R_hpf=R_hpf,R_lpf=R_lpf)),
                 linestyle = 'dashed',
                 color = 'orange')
     plt.xlim(22,22000)
     plt.ylim((1e12,1e19))
     plt.xlabel('    $ $', fontsize = 16)
     plt.ylabel('$    $',fontsize = 16)
     plt.legend()
     plt.grid()
     plt.semilogy()
     plt.xticks(np.arange(20,22000,1000))
     plt.show()
```



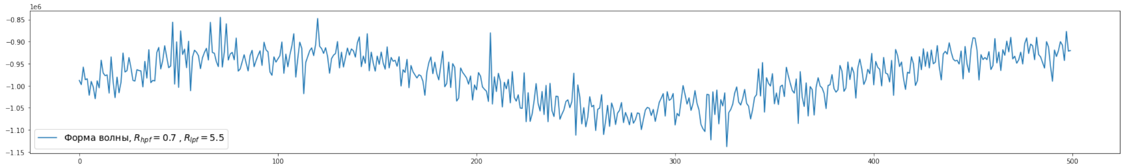
$-\pi/6$

1.0.4 1.3 - +

```
[ ]: rate, data = sp.io.wavfile.read('DR0000_0232.wav')
     data=data[:,0]
     time = np.arange(0, len(data), 1)/rate
     R_hpf,R_lpf = 0.7,5.5
```

```
C:\Users\     \AppData\Local\Temp\ipykernel_15004\3387931043.py:1:
WavFileWarning: Chunk (non-data) not understood, skipping it.
  rate, data = sp.io.wavfile.read('DR0000_0232.wav')
```

```
[ ]: plt.figure(figsize=(30,4))
     plt.plot(data[2000:2500],label = "       , $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf}␣
       ↪= "+str(R_lpf)+"$")
     plt.legend(fontsize = 14)
     plt.show()
```



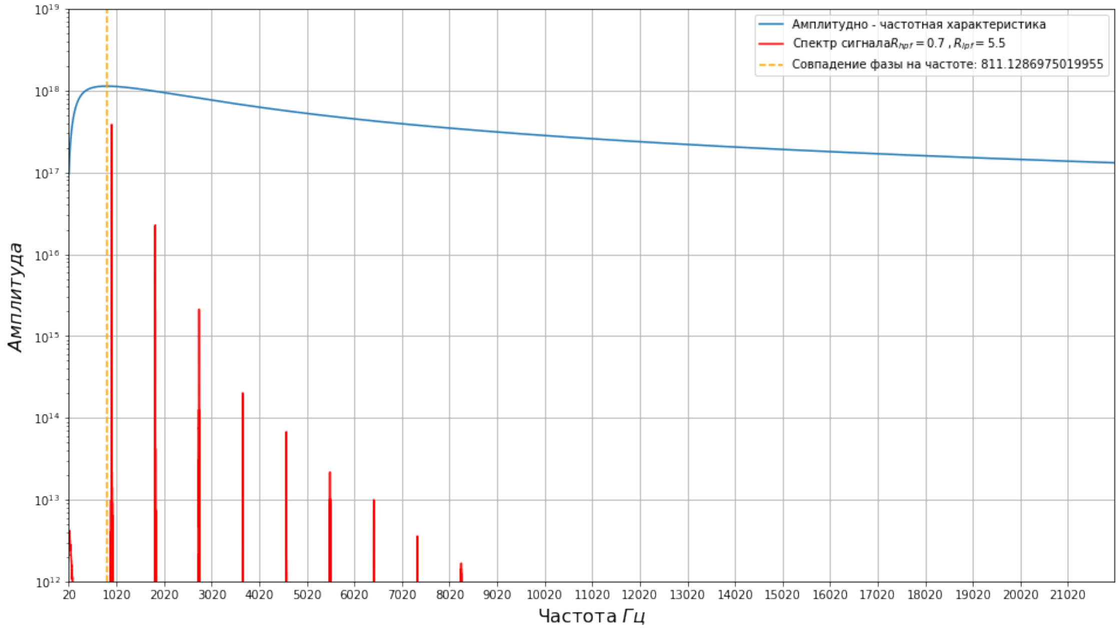
```
[ ]: IPython.display.Audio("DR0000_0232.wav")
```

```
[ ]: <IPython.lib.display.Audio object>
```

```
[ ]: periodogram = np.abs(fft(data))**2 / (rate * len(data))
     frequencies = fftfreq(len(data), d=1/rate)
     frequencies
```

```
[ ]: array([ 0.        ,  0.10212418,  0.20424837, …, -0.30637255,
            -0.20424837, -0.10212418])
```

```
[ ]: freq = np.linspace(22,22000,20000)
     amp = 1e19
     plt.figure(figsize=(16,9))
     plt.plot(freq,getAfr(freq,R_hpf,R_lpf)*amp,label = '       -               ')
     plt.plot(fftshift(frequencies), fftshift(periodogram), color='red',label = "    ⊔
      ↪    $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf} = "+str(R_lpf)+"$")
     plt.axvline(x=getMaxAmpFreq(R_hpf=R_hpf,R_lpf=R_lpf),
                 label = "               :⊔
      ↪"+str(getMaxAmpFreq(R_hpf=R_hpf,R_lpf=R_lpf)),
                 linestyle = 'dashed',
                 color = 'orange')
     plt.xlim(22,22000)
     plt.ylim((1e12,1e19))
     plt.xlabel('     $ $', fontsize = 16)
     plt.ylabel('$     $',fontsize = 16)
     plt.legend()
     plt.grid()
     plt.semilogy()
     plt.xticks(np.arange(20,22000,1000))
     plt.show()
```



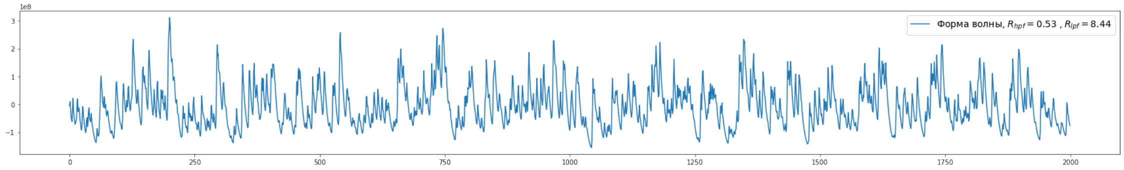
1.0.5 2 -

2.1 - <>

[ ]:
```
rate, data = sp.io.wavfile.read('DR0000_0237.wav')
data=data[:,0]
time = np.arange(0, len(data), 1)/rate
R_hpf,R_lpf = 0.53,8.44
```

```
C:\Users\     \AppData\Local\Temp\ipykernel_15004\912230510.py:1:
WavFileWarning: Chunk (non-data) not understood, skipping it.
  rate, data = sp.io.wavfile.read('DR0000_0237.wav')
```

[ ]:
```
plt.figure(figsize=(30,4))
plt.plot(data[500:2500],label = "        , $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf}␣
  ↪= "+str(R_lpf)+"$")
plt.legend(fontsize = 14)
plt.show()
```



[ ]:

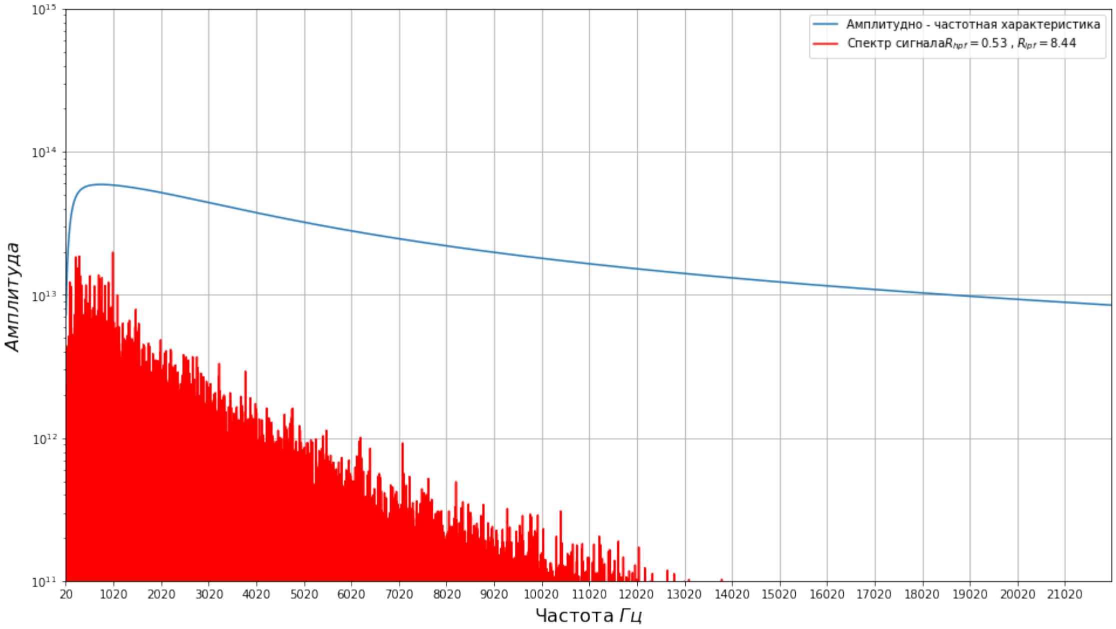
```
IPython.display.Audio("DR0000_0237.wav")
```

[ ]:

```
<IPython.lib.display.Audio object>
```

[ ]:

```
periodogram = np.abs(fft(data))**2 / (rate * len(data))
frequencies = fftfreq(len(data), d=1/rate)
freq = np.linspace(22,22000,20000)
amp = 1e15
plt.figure(figsize=(16,9))
plt.plot(freq,getAfr(freq,R_hpf,R_lpf)*amp,label = '       -               ')
plt.plot(fftshift(frequencies), fftshift(periodogram), color='red',label = "    ␣
 ↪   $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf} = "+str(R_lpf)+"$")
plt.xlim(22,22000)
plt.ylim((1e11,1e15))
plt.xlabel('     $ $', fontsize = 16)
plt.ylabel('$     $',fontsize = 16)
plt.legend()
plt.grid()
plt.semilogy()
plt.xticks(np.arange(20,22000,1000))
plt.show()
```



2.2 -

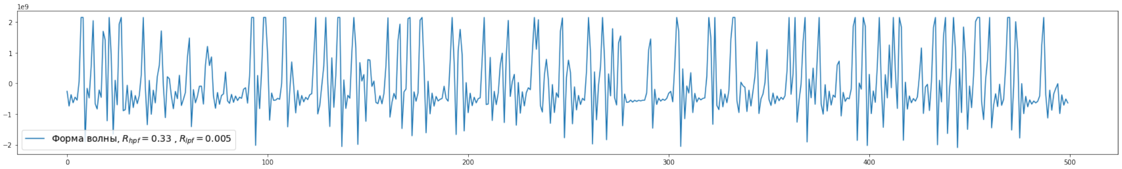
[ ]:

```
rate, data = sp.io.wavfile.read('DR0000_0238.wav')
data=data[:,0]
time = np.arange(0, len(data), 1)/rate
```

```
R_hpf,R_lpf = 0.33,0.005
```

```
C:\Users\     \AppData\Local\Temp\ipykernel_15004\1410607889.py:1:
WavFileWarning: Chunk (non-data) not understood, skipping it.
  rate, data = sp.io.wavfile.read('DR0000_0238.wav')
```

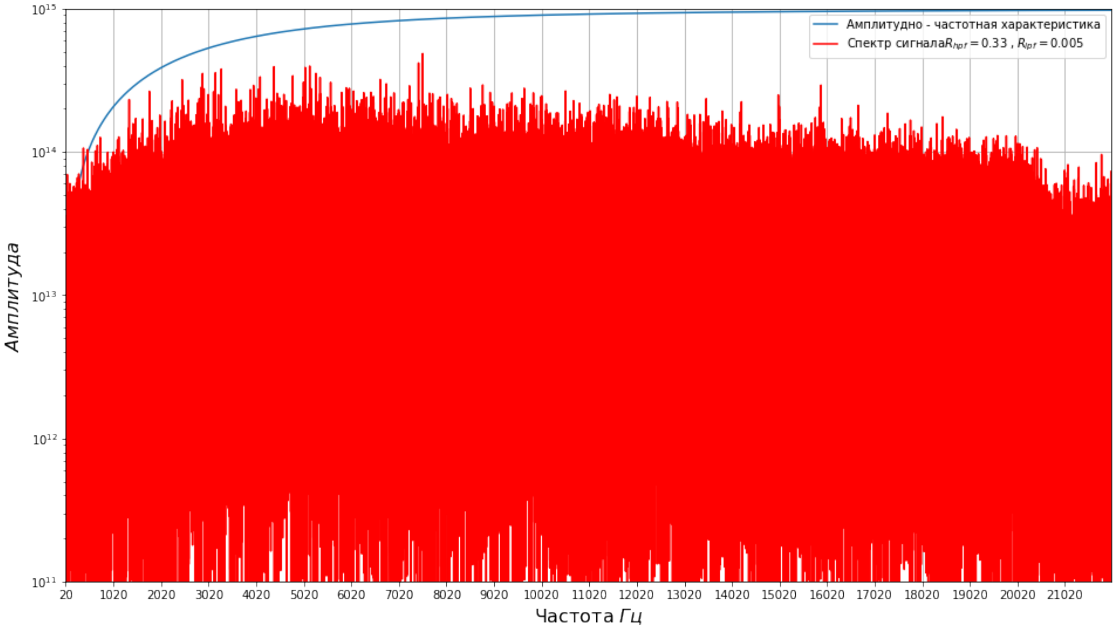
```
[ ]: plt.figure(figsize=(30,4))
     plt.plot(data[500:1000],label = "        , $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf}␣
      ↪= "+str(R_lpf)+"$")
     plt.legend(fontsize = 14)
     plt.show()
```



```
[ ]: IPython.display.Audio("DR0000_0238.wav")
```

```
[ ]: <IPython.lib.display.Audio object>
```

```
[ ]: periodogram = np.abs(fft(data))**2 / (rate * len(data))
     frequencies = fftfreq(len(data), d=1/rate)
     freq = np.linspace(22,22000,20000)
     amp = 1e15
     plt.figure(figsize=(16,9))
     plt.plot(freq,getAfr(freq,R_hpf,R_lpf)*amp,label = '       -               ')
     plt.plot(fftshift(frequencies), fftshift(periodogram), color='red',label = "    ␣
      ↪     $R_{hpf} = "+str(R_hpf)+"\ , R_{lpf} = "+str(R_lpf)+"$")
     plt.xlim(22,22000)
     plt.ylim((1e11,1e15))
     plt.xlabel('     $ $', fontsize = 16)
     plt.ylabel('$     $',fontsize = 16)
     plt.legend()
     plt.grid()
     plt.semilogy()
     plt.xticks(np.arange(20,22000,1000))
     plt.show()
```


Амплитудно - частотная характеристика
Спектр сигнала$R_{hpf} = 0.33$, $R_{lpf} = 0.005$
10^{15}
10^{14}
10^{13}
10^{12}
10^{11}
Амплитуда
20
1020
2020
3020
4020
5020
6020
7020
8020
9020
10020
11020
12020
13020
14020
15020
16020
17020
18020
19020
20020
21020
Частота *Гц*