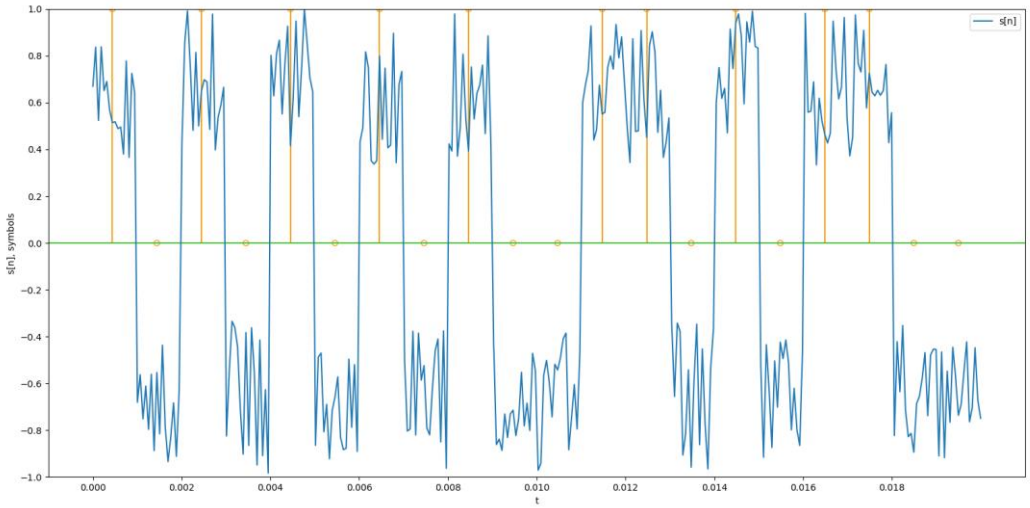
Ondřej Studnička (**xstudn00**)

*1. úkol*

Vzorkovací frekvence: **16 000Hz**
Délka signálu ve vzorcích: **32 000**
Délka signálu v sekundách: **2s**
Počet binárních symbolů: **2 000**

```
sound = wave.open('xstudn00.wav', 'rb')
vzorkovaci_frekvence = sound.getframerate()
delka_ve_vzorcich = sound.getnframes()
delka_v_sekundach = delka_ve_vzorcich/vzorkovaci_frekvence
pocet_binarnich_symbolu = int(delka_ve_vzorcich/16)
```

*2. úkol*

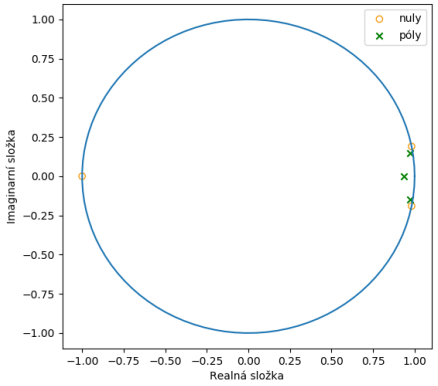



```
ret = []
while sound.tell() < sound.getnframes():
(decoded, ) = struct.unpack("<h", sound.readframes(1))
ret.append(decoded)
```

```
binary_numbers = []
for i in range(7,32000,16):
if ret[i] > 0:
binary_numbers.append(1)
else:
binary_numbers.append(0)
```

*3. úkol*

```
b = [0.0192, -0.0185, -0.0185, 0.0192]
a = [1, -2.8870, 2.7997, -0.9113]

z, p, k = tf2zpk(b, a)
is_stable = (p.size == 0) or np.all(np.abs(p) < 1)
```

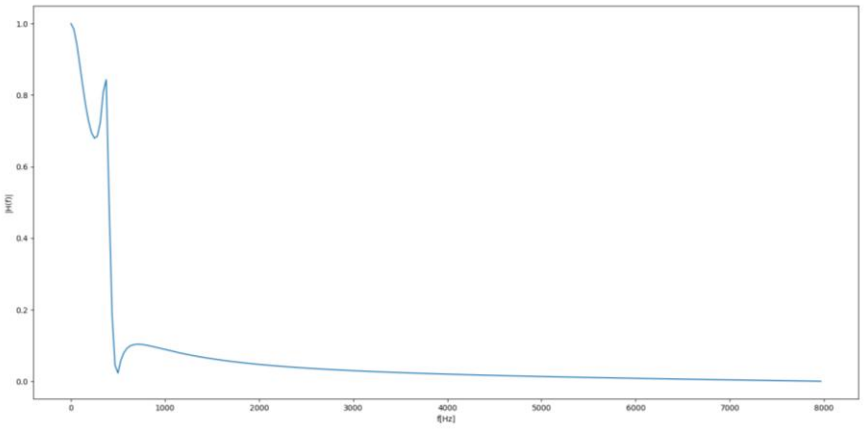



Filtr **je stabilní**.

*4. úkol*

**Dolní propust**
Mezní frekvence: **500Hz**
(vyčteno z grafu, lokální minimum na intervalu <0;1000>)

```
ret = []
for i in range(256):
i = i/256*vzorkovaci_frekvence/2
ret.append(i)

H = list(freqz(b,a,256))
plt.plot(ret, abs(H[i]))
```



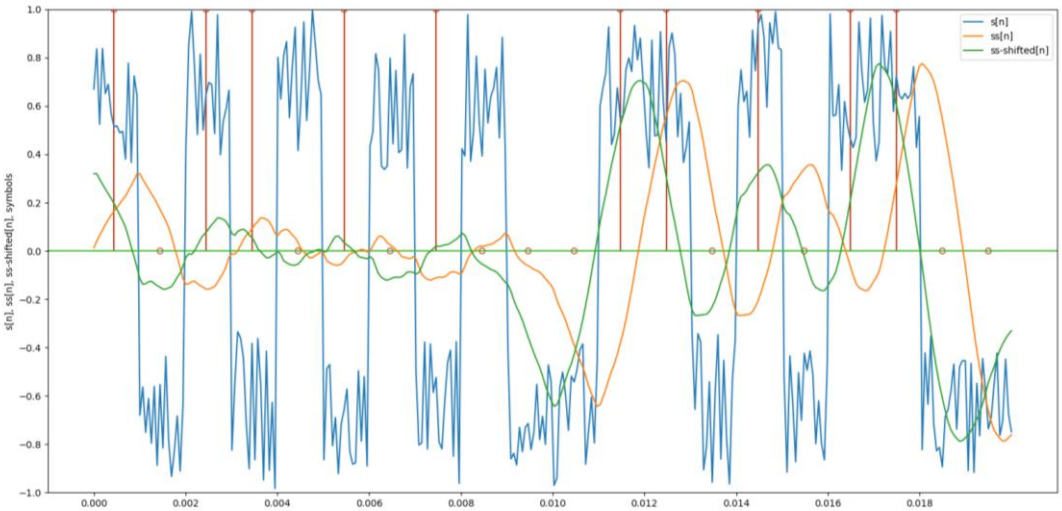
Ondřej Studnička (**xstudn00**)

*5. úkol*

Signál budu posouvat **o 15 vzorků doleva (-> předběhnutí)**.
Použil jsem metodu křížové korelace.

```
signalFiltered = lfilter(b, a, signal)

x = list(np.correlate(signal, signalFiltered, "full"))
y = (max(x))
z = x.index(y)
shift = len(signal) – z
```

*6. úkol*




Postup byl stejný jako u druhého úkolu.

```
signalShifted = signalFiltered[15:32000]
```

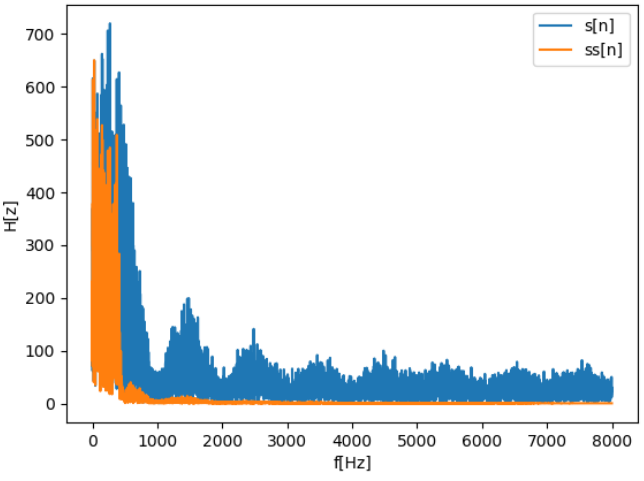
*7. úkol*

Chybovost: **5,45%**
Počet chyb: **109**

```
Získání binárních čísel probíhá stejně jako ve druhém úkolu.
counter = 0
for i in range(1999):
if binary_numbers[i] != binary_numbers_shifted[i]:
counter = counter + 1
```

*8. úkol*

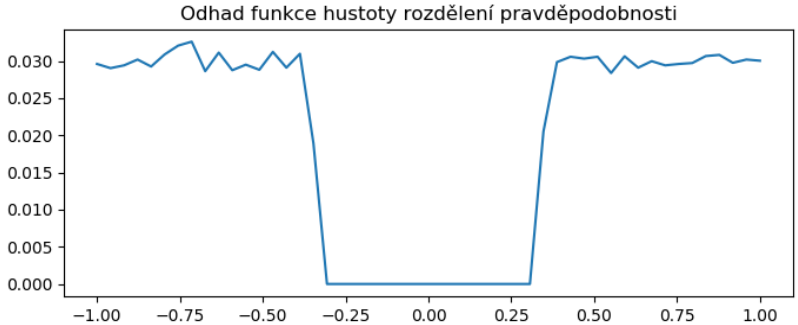
```
signalFFT = npf.fft(signal, 16000)
moduleSignalFFT = abs(signalFFT)
plt.plot(range(8000), moduleSignalFFT[0:8000],
label='s[n]')

signalFilteredFFT = npf.fft(signalFiltered, 16000)
modulesignalFilteredFFT = abs(signalFilteredFFT)
plt.plot(range(8000),
modulesignalFilteredFFT[0:8000], label='ss[n]')
```




*9. úkol*

Integrál vyšel 1 -> správně.

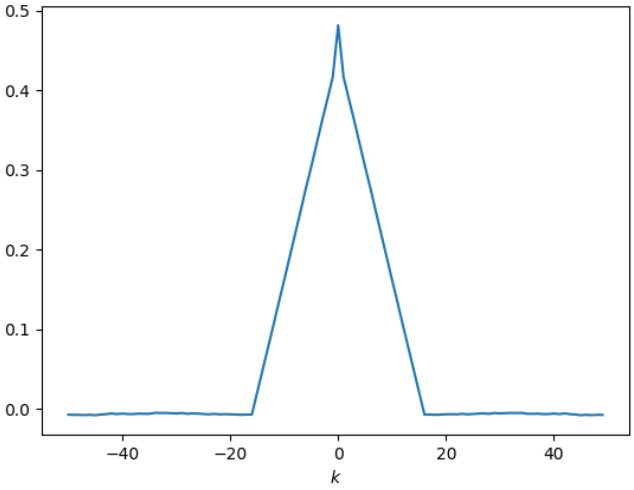
```
hist, _ = np.histogram(signal, n_aprx)
integral= np.sum(px)
```



Ondřej Studnička (**xstudn00**)

*10. úkol*

```
k = np.arange(-delka_ve_vzorcich+1,
delka_ve_vzorcich)
Rv = np.correlate(signal[:delka_ve_vzorcich:],
signal[:delka_ve_vzorcich:], 'full') / delka_ve_vzorcich

pocet_indexu = 50
pocatek = k.size // 2 - pocet_indexu
konec = k.size // 2 + pocet_indexu
x = k[pocatek:konec:]
y = Rv[pocatek:konec:]
plt.plot(x, y)
```



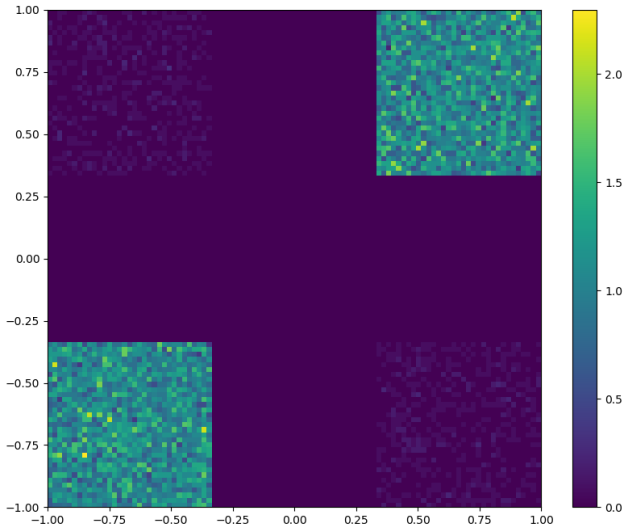

*11. úkol*

```
print(Rv[-1+delka_ve_vzorcich])
print(Rv[0+delka_ve_vzorcich])
print(Rv[15+delka_ve_vzorcich])
```

R[0] = 0.4815264367201469
R[1] = 0.4167632674189905
R[16] = -0.006860904450528325

*12. úkol*

```
x = np.linspace(min(signal), max(signal), 100)

px1x2, x1_edges, x2_edges = np.histogram2d(
signal[0:delka_ve_vzorcich-1]
,signal[1:delka_ve_vzorcich], x, normed=True)

X, Y = np.meshgrid(x1e_all[0], x2e_all[0])
im = plt.pcolormesh(X, Y, px1x2_all[0])

cbar = plt.colorbar(im)
```




*13. úkol*

```
int_all = []
binsize = np.abs(x1_edges[0] - x1_edges[1]) *
np.abs(x2_edges[0] - x2_edges[1])
integral = np.sum(px1x2 * binsize)
int_all.append(integral)
print(int_all)
```

Jedná se o správnou hodnotu. Proměnná int_all vychází 0.9999999999999947, což je v podstatě jedna -> správně.

*14. úkol*

```
binsize = np.abs(x1_edges[0] - x1_edges[1]) *
np.abs(x2_edges[0] - x2_edges[1])
bin_centers_x1 = x1_edges[:-1] + (x1_edges[1:] - x1_edges[:-1])
/ 2
bin_centers_x2 = x2_edges[:-1] + (x2_edges[1:] - x2_edges[:-1])
/ 2
x1x2 = np.outer(bin_centers_x1, bin_centers_x2)
R = np.sum(x1x2 * px1x2 * binsize)
R_all.append(R)
print(R_all)
```

(R[1]=) R_all vychází 0.4166899344648254. Při porovnání s R[1] z 11. úkolu zjistíme, že se na desetitisících začínají lišit. To je pravděpodobně způsobeno zaokrouhlováním.