	<pre>bt(ekg1.index, ekg1['9'], color='khaki') bt(ekg1.index, ekg1['10'], color='lime') bt(ekg1.index, ekg1['11'], color='coral') bt(ekg1.index, ekg1['12'], color='pink') cle('ekg1.txt') bbel('Czas[s]') bbel('Wartość') cm(0,ekg1_time) cname = input("Podaj nazwe pliku do zapisu: ") cavefig(file_name + ".jpg") bw() ekg1.txt</pre>
1000 -	
ekg100 samplir ekg100 ekg100 plt.fic plt.plc plt.tit plt.xla	<pre>i</pre>
plt.yli # file_ # plt.s plt.sho	m(0,800) m(-3,2.0) name = input("Podaj nazwe pliku do zapisu: ") savefig(file_name + ".jpg") w() ekg100.txt
-1 -1 -2 -	100 200 300 400 500 600 700
ekg_noisamplir ekg_noiekg_noiekg_noi plt.fig plt.yla plt.xla plt.xla plt.xla	<pre>s.getcwd() sse = pd.read_csv(str1+'\\ekg_noise.txt', names=['Czas', 'Wartosc amplitudy'], sep='\s+') sg_frequency_ekg_noise = 360 # czestotliwosc probkowania sse['Czas'] = ekg_noise.index/sampling_frequency_ekg_noise # ustalenie punktów na osi czasu sse = ekg_noise.set_index('Czas') sse_time = ((ekg_noise.shape[0])/sampling_frequency_ekg_noise) sure(figsize=(20,10)) st(ekg_noise.index, ekg_noise['Wartosc amplitudy']) ste('ekg_noise.txt') stel('Czas[s]') stel('Czas[s]') stel('Wartość') sm(0, ekg_noise_time) sm(-1.0,1.5)</pre>
	name = input("Podaj nazwe pliku do zapisu: ") savefig(file_name + ".jpg") w() ekg_noise.txt
-0.5 -1.0 Cwicze	1 2 3 4 Czas[s] 5 6 7 8
Celem cv 1) Wyger f = 50 length N = np. fs = 48 sinus = plt.fic plt.plc plt.tit	viczenia jest praktyczne wypróbowanie funkcji numpy.fft i numpy.ifft do wyznaczania prostej i odwrotnej transformaty Fouriera neruj ciąg próbek odpowiadający fali sinusoidalnej o częstotliwości 50 Hz i długości 65536. = 65_536 arange (length)
plt.xla plt.axh plt.sho	ubel('Numer próbki') uline(y=0, color='black')
-0.25 - -0.50 - -1.00 -	o 10000 20000 30000 40000 50000 60000 acz dyskretną transformatę Fouriera tego sygnału i przedstaw jego widmo amplitudowe na wykresie w zakresie częstotliwości
fs/2], gd: widmo = f = np. plt.fic plt.plc plt.tit plt.yla plt.xla plt.xla plt.sho	<pre>zie fs oznacza częstotliwość próbkowania. = np.abs(np.fft.rfft(sinus)) / (length/2) fft.rfftfreq(length, 1/fs) gure(figsize=(20,5)) et(f, widmo) ele('Widmo fali sinusoidalnej') ebel('Wartość') ebel('Częstotliwość[Hz]') em (0, sampling_frequency_ekg1)</pre>
	zóo 400 Gzęstotliwość[Hz] 600 800 Deruj ciąg próbek mieszaniny dwóch fal sinusoidalnych (tzn. ich kombinacji liniowej) o częstotliwościach 50 i 60 Hz. Wykonaj z punktu 2 dla tego sygnału.
<pre>f1 = 50 f2 = 60 fs = 48 N = np. y1 = np y2 = np widmo = plt.fig plt.sub plt.plc plt.tit</pre>	# ustawienie częstotliwości 50 Hz # ustawienie częstotliwości 60 Hz # usta
plt.xla plt.yla plt.suk plt.plc plt.xli plt.tit plt.xla	<pre>abel('Numer próbki') abel('Wartość') aplot(2, 1, 2) at(f, widmo) am(0, 1000) ale('Widmo amplitudowe') abel('Częstotliwość[Hz]') abel('Wartość')</pre>
0.5 - 0.0	0 10000 20000 30000 40000 50000 60000 Numer pröbki Widmo amplitudowe
0.2 - 0.0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 -	zoo 400 Częstotliwość[Hz] 600 800 rz eksperymenty dla różnych czasów trwania sygnałów, tzn. dla różnych częstotliwości próbkowania. 60_000 arange (length) 2. sin (2 * np.pi * N * f1/fs) 2. sin (2 * np.pi * N * f2/fs)
plt.fig plt.suk plt.plc plt.tit plt.xla plt.yla plt.suk plt.plc plt.xli plt.tit plt.xla	<pre>inp.abs(np.fft.rfft(y1 + y2)) / (length/2) fure(figsize=(20,20)) pplot(4, 1, 1) pt(N,y1 + y2) cle('Kombinacja liniowa') cle('Numer próbki') cle('Wartość') pplot(4, 1, 2) pt(f, widmo) m(0, 1000) cle('Widmo amplitudowe') cle('Widmo amplitudowe') cle('Częstotliwość[Hz]')</pre>
fs = 24 N = np. y1 = np y2 = np widmo = plt.suk plt.plc plt.tit plt.xla	<pre>abel('Wartość') i0_000 arange(length) i.sin(2 * np.pi * N * f1/fs) i.sin(2 * np.pi * N * f2/fs) i.np.abs(np.fft.rfft(y1 + y2)) / (length/2) inplot(4, 1, 3) int(N,y1 + y2) ile('Kombinacja liniowa') ibel('Numer próbki') ibel('Wartość')</pre>
plt.plc plt.xli plt.tit plt.xla	<pre>pplot(4, 1, 4) pt(f, widmo) m(0, 1000) m(0, 1000) ple('Widmo amplitudowe') plet('Częstotliwość[Hz]') plet('Wartość') pw()</pre> <pre>Kombinacja liniowa</pre>
-0.5 -1.0 -1.5 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0	0 10000 20000 30000 40000 50000 60000 Numer próbki Widmo amplitudowe
2.0 - 1.5 - 1.0 - 0.5 1.0 1.5 2.0 1.0 2.0	200 400 Częstotiwośc(Hz] 600 800 Częstotiwośc(Hz] Kombinacja liniowa 0 10000 20000 30000 40000 50000 60000 Numer próbki Widmo amplitudowe
0.6 - 0.5 - 0.5 - 0.1 - 0.0 -	zóo 4óo Gzęstotliwość[Hz] 6óo 8óo nie 3. viczenia jest obserwacja widma sygnału EKG
ekg100 samplir ekg100 ekg100 plt.fic plt.plc plt.tit plt.xla plt.yla	ać sygnał ekg100.txt i ocenić go wizualnie na wykresie s.getcwd() = pd.read_csv(str1+'\\ekg100.txt', names=['1']) tg_frequency_ekg100 = 360 # czestotliwosc probkowania 'Czas'] = ekg100.index/sampling_frequency_ekg100 # ustalenie punktów na osi czasu = ekg100.set_index('Czas') pure(figsize=(20,5)) tt(ekg100.index, ekg100['1']) tle('ekg100.txt') tbel('Czas[s]') tbel('Czas[s]') tbel('Wartość') m(0.800)
plt.yli plt.sho	m(0,800) m(-3,2.0) w() ekg100.txt
gdzie fs o x = np. y = np. plt.fic plt.plc plt.xli plt.yli plt.tit	nczyć jego dyskretną transformatę Fouriera i przedstawić widmo amplitudowe sygnału w funkcji częstotliwości w zakresie [0, fs oznacza częstotliwość próbkowania. fft.fftfreq(ekg100.size, 1/sampling_frequency_ekg100) abs(np.fft.fft(ekg100['1']))/(ekg100.size//2) pure(figsize=(20,5)) abs(x, y) am(0,sampling_frequency_ekg100/2) am(0,0.015) cle('Widmo amplitudowe sygnału ekg100') abel('Częstotliwość [Hz]')
0.014 - 0.012 - 0.010 - 0.006 - 0.004 - 0.002 - 0.000	Widmo amplitudowe sygnału ekg100
Ćwicze Celem ćw wybrać r scipy.sig 1) Wczyt sygnału. str1=os	viczenia jest praktyczne wypróbowanie działania filtrów w celu wyeliminowania niepożądanych zakłóceń z sygnału EKG. Proszę odzaj filtra do eksperymentowania, np. Butterwortha lub Czebyszewa. Do filtracji wykorzystać gotowe funkcje z biblioteki nal aj sygnał ekg noise.txt i zauważ zakłócenia nałożone na sygnał. Wykreślić częstotliwościową charakterystykę amplitudową
ekg_noi widmo = frequer plt.fig plt.suk plt.plc plt.xla plt.xla plt.yla	noise = 360 # częstotliwość próbkowania se = pd.read_csv(str1+'\\ekg_noise.txt', names=['Czas', 'Wartosc amplitudy'], sep='\s+').set_index(np.abs(np.fft.rfft(ekg_noise['Wartosc amplitudy'] - ekg_noise['Wartosc amplitudy'].mean())) / (ekg ncy = np.fft.rfftfreq(ekg_noise.size, 1/fs_ekg_noise) # częstotliwościowa charakterystyka amplitude nure(figsize=(20,10)) nplot(2, 1, 1) nt(ekg_noise) nle('ekg_noise') nbel('Czas[s]') nbel('Wartość') nplot(2, 1, 2) nt(frequency, widmo)
plt.plc plt.tit plt.xla	ot(frequency, widmo) Ele('Częstotliwościowa charakterystyka amplitudowa sygnału') Obel('Częstotliwość [Hz]') Obel('Wartość')
-0.75	Częstotliwościowa charakterystyka amplitudowa sygnału Czas[s] Czestotliwościowa charakterystyka amplitudowa sygnału 0 25 50 75 100 125 150 175 Czestotliwość [Hz]
paramet Można to	j filtr dolnoprzepustowy o częstotliwości granicznej 60 Hz w celu redukcji zakłóceń pochodzących z sieci zasilającej. Wyznacz cy filtra, wykreśl jego charakterystykę (zależność tłumienia od częstotliwości), przebieg sygnału po filtracji oraz jego widmo. eż wyznaczyć różnicę między sygnałem przed i po filtracji i widmo tej różnicy. docy_bound = 60 # częstotliwość graniczna porth_filter = sig.butter(6, frequency_bound, 'low', output = 'sos', fs = sampling_frequency_ekg_noted = sig.sosfilt(butterworth_filter, ekg_noise['Wartosc amplitudy']) # Infinite impulse response pure(figsize=(20,15)) polot(3, 1, 1) pot(filtered)
butterw filtered plt.fig plt.suk plt.plc	<pre>cle('Sygnał po filtrze dolnoprzepustowym Butterwortha') dbel('Czas[s]') dbel('Wartość') sig.butter(6, frequency_bound/(sampling_frequency_ekg_noise/2), 'low') sig.freqz(b,a) # charakterystyka częstotliwościowa filtra sampling_frequency_ekg_noise / (2*np.pi) snp.log10(abs(h)) splot(3, 1, 2) st(x,y) cle('Zalezność tłumienia od częstotliwości') dbel('Częstotliwość [Hz]')</pre>
butterw filtered plt.fig plt.suk plt.plc plt.tit plt.xla plt.yla b,a = s w,h = s x = w*s y = 20*	.bel('Amplituda [dB]')
butterwifilters plt.fig plt.suk plt.yla b,a = s w,h = s x = w*s y = 20* plt.suk plt.plc plt.tit plt.xla plt.yla plt.suk plt.plc plt.tit plt.xla plt.yla plt.suk plt.plc plt.tit plt.xla plt.yla osa osa osa osa osa osa osa o	<pre>cplot(3, 1, 3) ot(filtered - ekg_noise['Wartosc amplitudy']) cle('Różnica miedzy sygnałem przed i po filtracji') dbel('Czas[s]') dbel('Wartość')</pre>
butterweitlerer plt.fig plt.suk plt.plc plt.tit plt.xla plt.yla b,a = s w,h = s x = w*s y = 20* plt.suk plt.plc plt.tit plt.xla plt.yla plt.tit plt.xla plt.yla plt.suk plt.suk plt.yla plt.suk	<pre>oplot(3, 1, 3) ot(filtered - ekg_noise['Wartosc amplitudy']) cle('Różnica miedzy sygnałem przed i po filtracji') dbel('Czas[s]') dbel('Wartość') ow()</pre>
butterwe filtered plt.fig plt.suk plt.plc plt.tit plt.xla plt.yla plt.suk plt.plc plt.tit plt.xla plt.yla plt.suk plt.plc plt.tit plt.xla plt.yla plt.sho	pplot(3, 1, 3) pt (filtered - ekg_noise['Wartosc amplitudy']) pte('Różnica miedzy sygnałem przed i po filtracji') ptel('Czas[s]') ptel('Wartość') pw() Sygnał po filtrze dolnoprzepustowym Butterwortha
butterwe filters plt. suk plt. plt. yla	plot(3, 1, 3) r(filtered = ekg poise' Wartosc amplitudy')) rel(Woorlds miedry aygnalem przed i po filtracji')
butterwe filters plt. suk plt. plt. yla	modet(3, 1, 3) (filtered - ekg noise; "Nactosc amplitudy") let (Notacian miedzy asymniem pozed i po filtrac(i*) mel (Ozaz (s)*) Sygnal po filtrac dohrsprzeputowym Buttowortha you was a second to mineral political politi