Labolatorium PTI – ćwiczenie 1, przykład 3

Reprezentacja i przetwarzanie sygnałów

Nadpróbkowanie sygnałów audio z wykorzystaniem wybranych metod interpolacyjnych

Nadpróbkowanie sygnałów audio z wykorzystaniem wybranych metod interpolacyjnych

1. Wstęp

Nadpróbkowanie sygnałów polega na uzyskaniu próbek o większej częstotliwości próbkowania niż to wynika z pasma sygnału. Z twierdzenia o próbkowaniu wynika, że idealne odtworzenie sygnału jest możliwe jeśli częstotliwość próbkowania jest większa niż dwukrotna największa częstotliwość sygnału próbkowanego. W przypadku sygnałów dźwiękowych, próbkowanych na użytek późniejszego ich odsłuchiwania zakłada się, że pełen zakres słyszalnego pasma mieści się w granicach od 20 Hz do 20 kHz. Zatem minimalna częstotliwość próbkowania spełniająca ten warunek w przypadku sygnałów dźwiękowych to 40 kHz.

Nadpróbkowanie sygnałów dźwiękowych za pomocą metod interpolacyjnych stosuje się gdy do pozyskania próbek wykorzystano przetworniki analogowo-cyfrowe (A/D) o niskiej rozdzielczości (w dzisiejszych czasach oznacza to np. przetworniki 8-bitowe) i/lub o małej częstotliwości próbkowania (np. 8 kHz). Jedną z technik stosowaną do poprawy jakości takich próbek jest zastosowanie filtrów antyaliasingowych. Zwiększenie częstotliwości próbkowania na wejściu takich filtrów pozwala znacznie uprościć konstrukcję jak i polepszyć właściwości filtru antyaliasingowego, gdzie koszt obliczeniowy interpolacji jest znacznie niższy niż w przypadku zastosowania bardziej skomplikowanej konstrukcji filtru.

W zastosowaniach telekomunikacyjnych, cyfrowej telefonii VoIP (Voice over Internet Protocol – protokoły SIP, H.323, komunikatory Skype, Hangout) lub PSTN (Publiczna komutowana sieć telefoniczna – sygnalizacja ISDN) standardowym formatem reprezentacji dźwięku są dane o rozdzielczości 8-bitowej, próbkowane z częstotliwością 8 kHz (pierwsze standardy definiujące te protokoły komunikacyjne i ich implementacje powstawały w latach 80 i 90 ubiegłego wieku – RFC oraz wytyczne ITU-T). W celu realizacji przejścia połączeń głosowych pomiędzy tymi systemami, a więc zastosowania konwersji formatów kompresji

dźwięku również stosuje się metody interpolacyjne w celu zwiększenia rozdzielczości próbek, filtrowania (filtry antyaliasingowe, dolno-przepustowe) w celu polepszenia jakości reprezentowanego sygnału, a następnie wykonywana jest kompresja sygnału audio (np. do formatów G.723.1, G.726, G.729(A)(B)).

Przy przetwarzaniu dźwięku w systemach telekomunikacyjnych częstym problemem są również szumy pochodzące z części analogowej urządzeń końcowych systemu (terminali) – toru pomiędzy mikrofonem, a przetwornikiem analogowo-cyfrowym lub też szumy tła, czyli otoczenia rozmówcy, które rejestruje mikrofon. Do ich usunięcia stosuje się filtry dolno-przepustowe na wejście których wprowadza się nadpróbkowany metodami interpolacyjnymi sygnał po czym częstotliwość i rozdzielczość próbki po przetworzeniu przez filtr redukowana jest do standardowego formatu 8-bitów / 8 kHz.

2. Metody i opis eksperymentu

W testach wykorzystano dwa rodzaje próbek – tony DTMF generowane programowo w sposób symulujący próbkowanie z częstotliwością 8 kHz jako dane wejściowe dla metod interpolacyjnych oraz 44,1 kHz jako sygnał referencyjny dla próbek uzyskanych metodą interpolacji.

Drugi rodzaj próbek uzyskano dodając biały szum gaussowski do w/w 8 kHz i 44,1 kHz próbek sygnałów DTMF. Następnie powtórzono krok interpolacyjny na podstawie 8 kHz próbek zawierających szum i porównano wyniki z sygnałami referencyjnymi 44,1 kHz zakłóconymi białym szumem i bez zakłóceń.

W eksperymencie jako implementacji metod interpolacyjnych wykorzystano funkcję `interp1` pakietu MATLAB służącej do interpolacji danych 1-wymiarowych. Porównywane metody to interpolacja liniowa (`'linear'`), interpolacja wielomianami Hermita trzeciego stopnia (`'pchip'`) oraz wielomianami trzeciego stopnia `'spline'`.

Funkcja `'pchip'` (wielomian interpolacyjny Hermita trzeciego stopnia - Piecewise Cubic Hermite Interpolating Polynomial) jest wielomianem kawałkami trzeciego stopnia, który interpoluje (zachowuje) podane wartości i jednocześnie - określone pochodne w punktach interpolacji. Dwa punkty i dwie wartości pochodnych dane na końcach każdego interpolowanego przedziału (czyli w tzw. węzłach interpolacji) to cztery narzucone warunki na krzywą interpolacyjną. Pozwalają więc na użycie wewnątrz przedziałów (czyli pomiędzy interpolowanymi punktami/węzłami) wielomianów trzeciego stopnia. W przypadku podania jedynie wartości funkcji w węzłach, wymagane pochodne są wyliczane na podstawie punktów sąsiednich, tak aby zapewnić kształt krzywej możliwie bliski danym węzłom. W rezultacie wynikowa krzywa/funkcja interpolacyjna zachowuje w węzłach jedynie ciągłość C^1.

Interpolacja `'spline'` (w badanym przypadku również funkcjami trzeciego stopnia), w przeciwieństwie do metody `'pchip'`, korzysta bezpośrednio z wartości (pobranych próbek) w węzłach interpolacji i z definicji zapewnia ciągłość do drugiej pochodnej włącznie. Uzyskany kształt funkcji jest więc gładszy, ale przez to skoki (amplituda/zmienność) funkcji może być większa, a sama funkcja może być gorzej dopasowana do danych.

2.1. Interpolacja tonów DTMF

Tony DTMF powstają z nałożenia na siebie dwóch sinusoidalnych fal dźwiękowych o częstotliwościach przypisanych danemu przyciskowi – niskiej poniżej 1000 Hz, wysokiej powyżej 1200 Hz. Częstotliwości te prezentuje poniższa tabela 2.1.1. DTMF jest przykładem zastosowania modulacji MFSK (kluczowania wieloczęstotliwościowego).

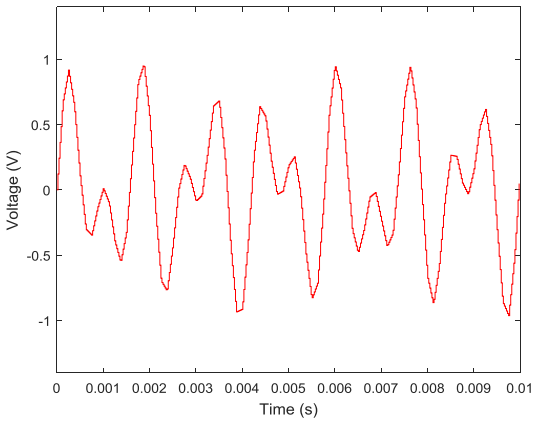
	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

2.1.1 Częstotliwości tonów DTMF dla poszczególnych klawiszy

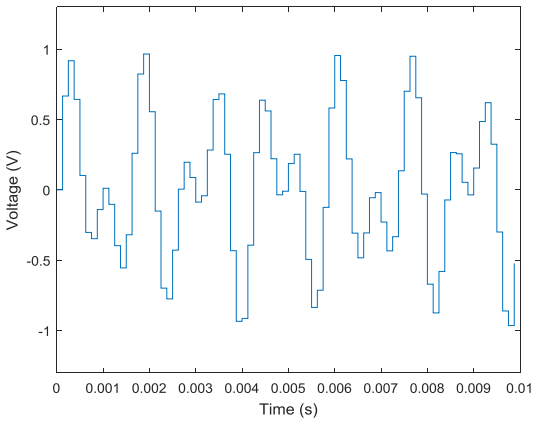
W eksperymencie wygenerowano próbki 8 kHz (wejściowe dla metod interpolacyjnych) i 44.1 kHz (próbki referencyjne) nakładając na nie kilka tonów DTMF – kolejno dla przycisku '1' (Rys. 2.1.1, 2.1.3), a następnie jednocześnie '15' (Rys. 2.1.7, 2.1.9), '159' (Rys. 2.1.13, 2.1.15) oraz '159D' (Rys. 2.1.19, 2.1.21). Uzyskano w ten sposób próbki zawierające kolejno 2, 4, 6, 8 sinusoidalnych fal o różnych częstotliwościach stopniując w ten sposób złożoność sygnału interpolowanego.

Do weryfikacji poprawności wygenerowanych tonów DTMF wykorzystano funkcję `goertzel` dostępną w pakiecie MATLAB „Signal Processing Toolbox" implementującą algorytm Goertzela obliczający w tym przypadku 8 punktów (697Hz, 770Hz, 852Hz, 941Hz, 1209Hz, 1336Hz, 1477Hz, 1633Hz) z 205-punktowej DFT (Dyskretna Transformata Fouriera) przy częstotliwości próbkowania 8 kHz (Rys. 2.1.5, 2.1.11, 2.1.17, 2.1.23).

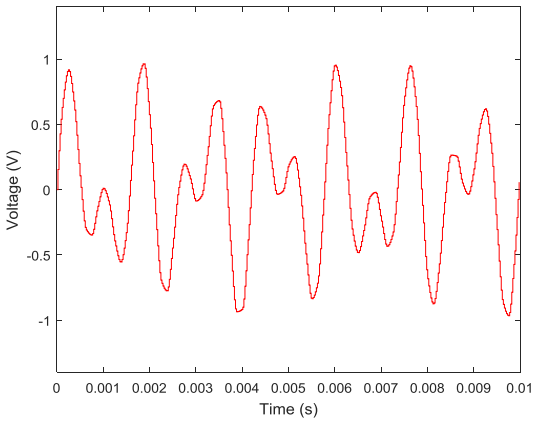
Błędy poszczególnych metod interpolacji dla zadanych 8 kHz próbek tonów DTMF zebrano w tabeli 2.1.2. Jako miarę jakości uzyskanych sygnałów wykorzystano wartość pierwiastka kwadratowego z błędu średniokwadratowego wyliczanego względem wygenerowanych referencyjnych próbek tonów DTMF z częstotliwością próbkowania 44,1 kHz.



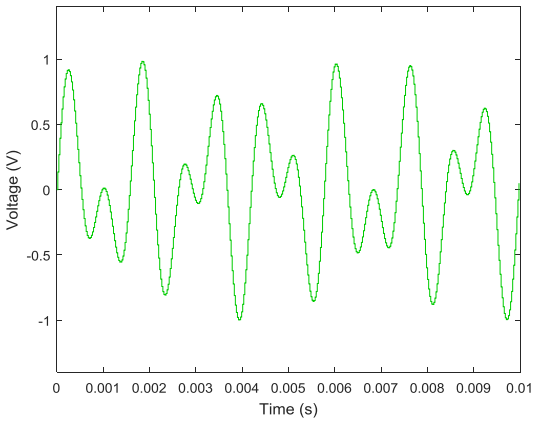
2.1.1 sygnał DTMF '1' int. liniowa - 44.1 kHz



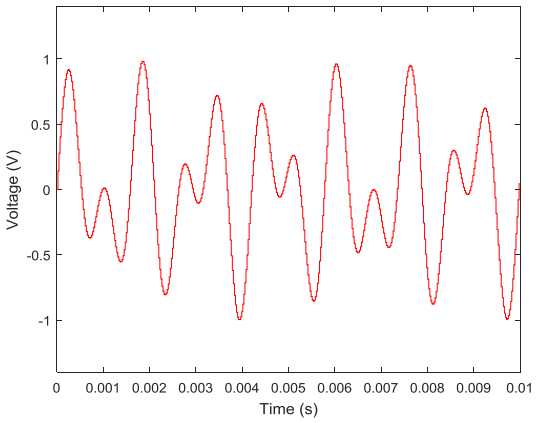
2.1.2 sygnał DTMF ‘1’ - 8 kHz



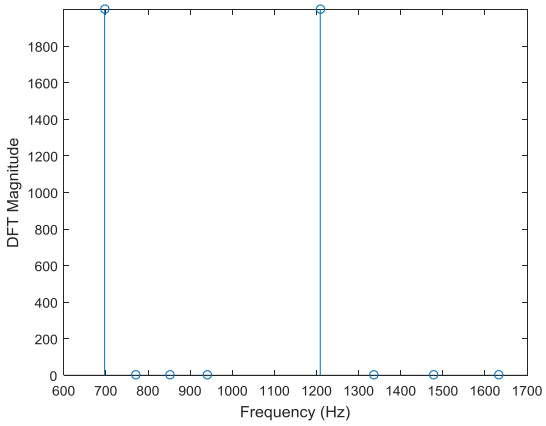
2.1.3 sygnał DTMF '1' int. pchip - 44.1 kHz



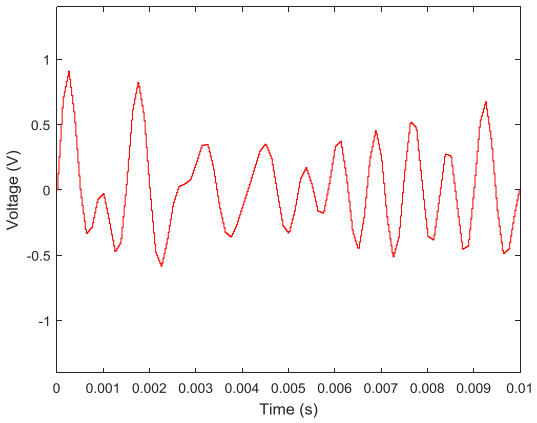
2.1.4 sygnał DTMF ‘1’ – 44.1 kHz



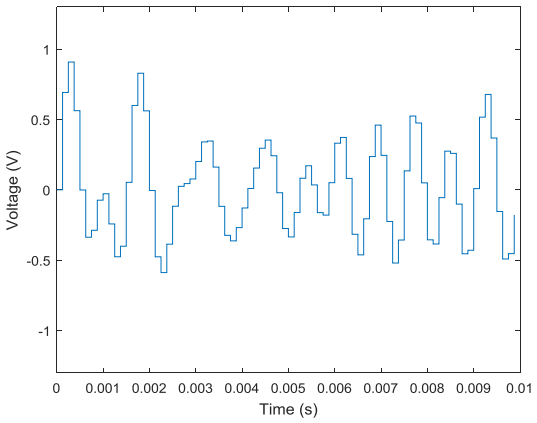
2.1.5 sygnał DTMF '1' int. spline - 44.1 kHz



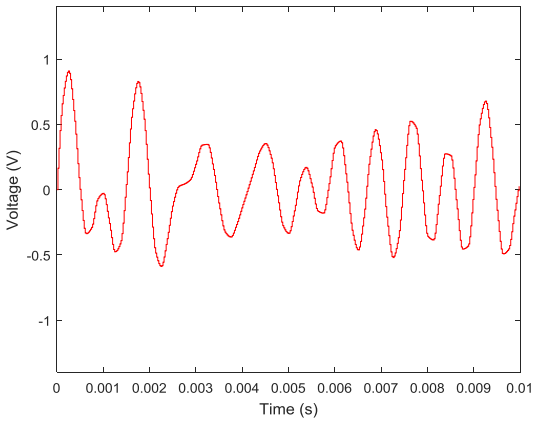
2.1.6 prążki DTMF ‘1’ – alg. Goertzela



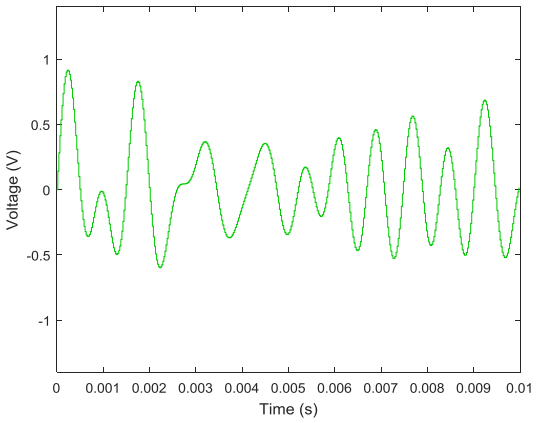
2.1.7 sygnał DTMF '15' int. liniowa - 44.1 kHz



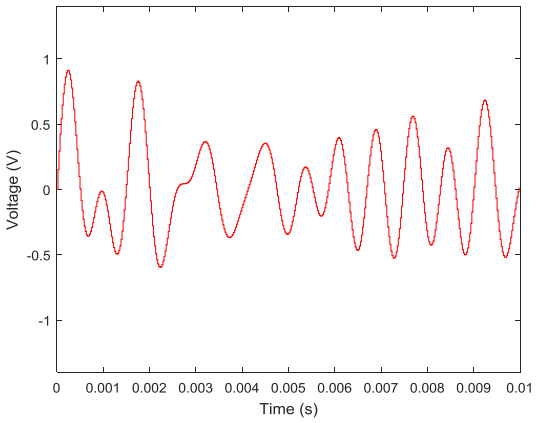
2.1.8 sygnał DTMF ‘15’ - 8 kHz



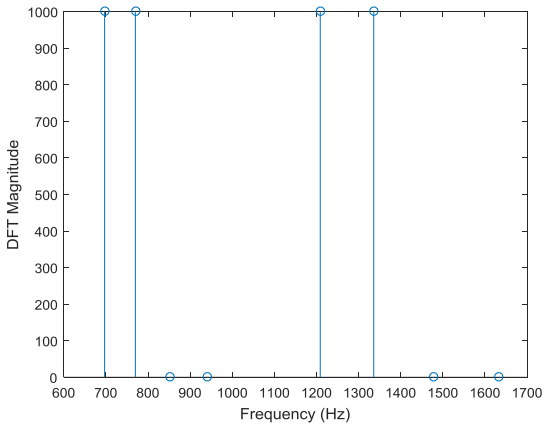
2.1.9 sygnał DTMF '15' int. pchip - 44.1 kHz



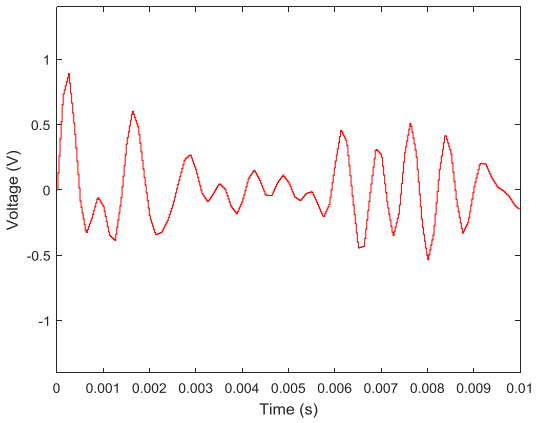
2.1.10 sygnał DTMF ‘15’ – 44.1 kHz



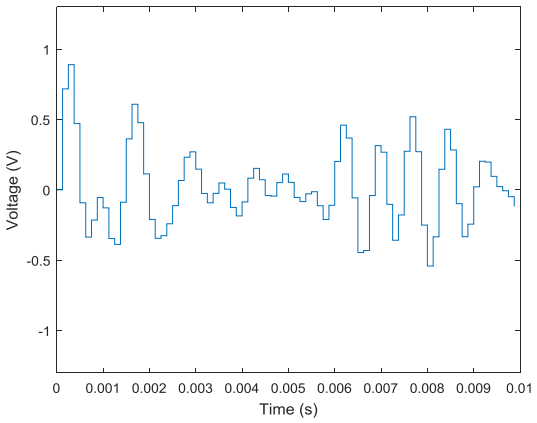
2.1.11 sygnał DTMF '15' int. spline - 44.1 kHz



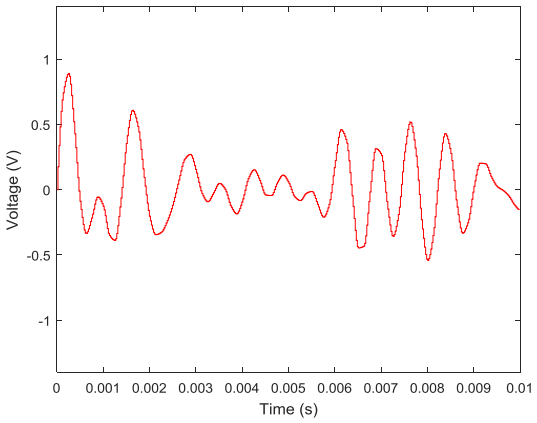
2.1.12 prążki DTMF ‘15’ – alg. Goertzela



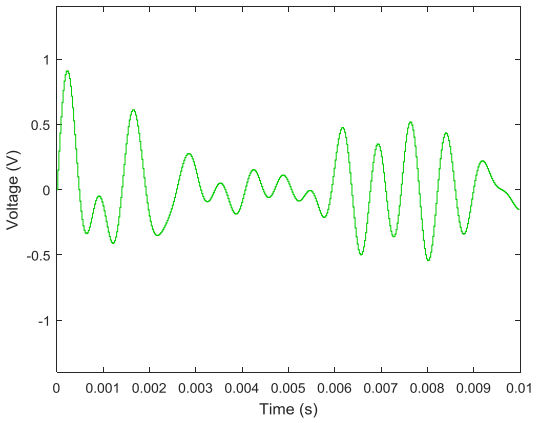
2.1.13 sygnał DTMF '159' int. liniowa - 44.1 kHz



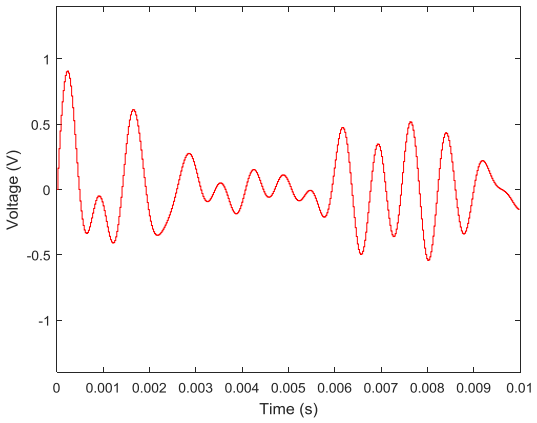
2.1.14 sygnał DTMF ‘159’ - 8 kHz



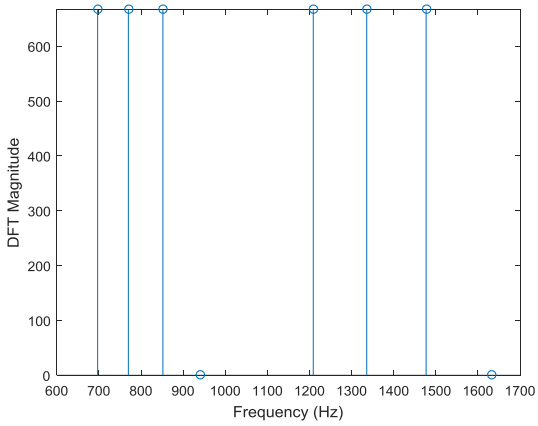
2.1.15 sygnał DTMF '159' int. pchip - 44.1 kHz



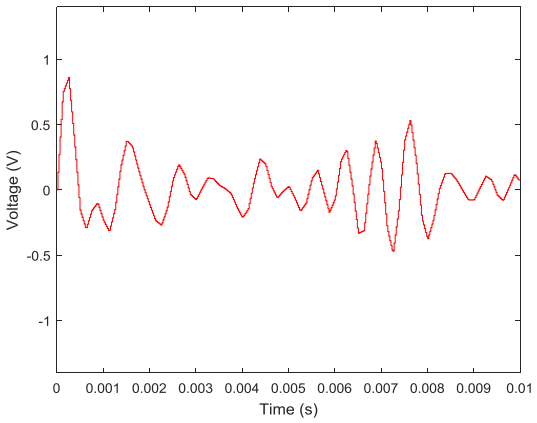
2.1.16 sygnał DTMF ‘159’ – 44.1 kHz



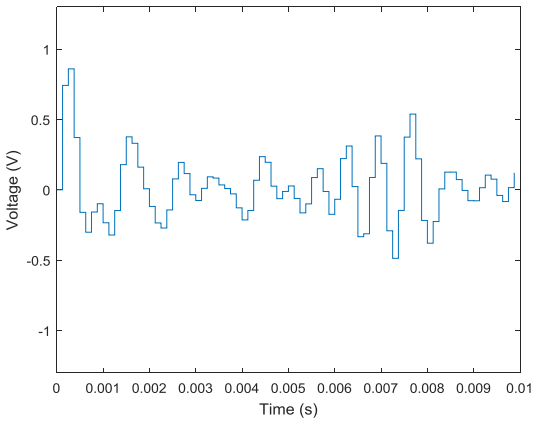
2.1.17 sygnał DTMF '159' int. spline - 44.1 kHz



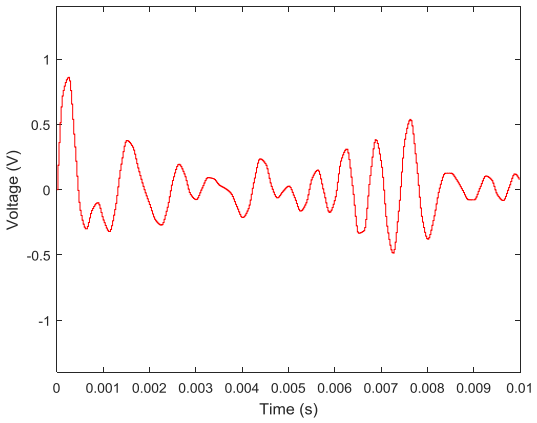
2.1.18 prążki DTMF ‘159’ – alg. Goertzela



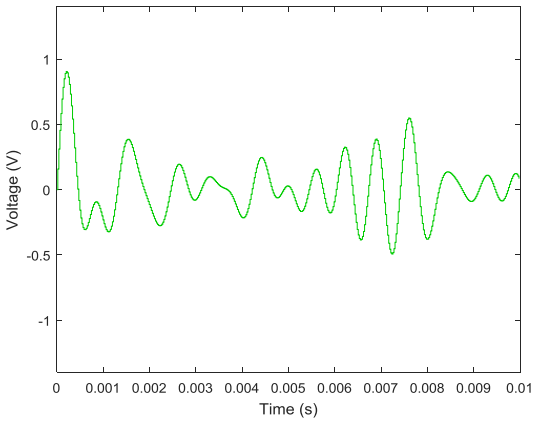
2.1.19 sygnał DTMF '159D' int. liniowa - 44.1 kHz



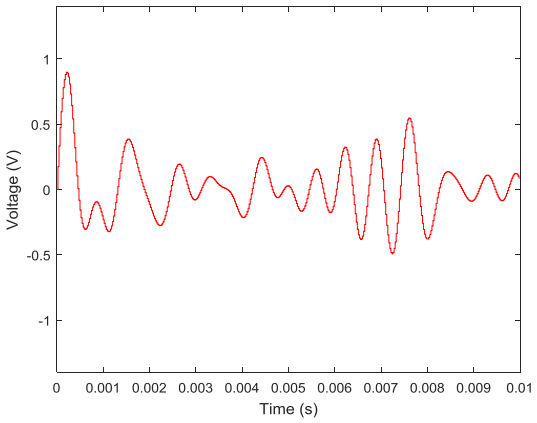
2.1.20 sygnał DTMF ‘159D’ - 8 kHz



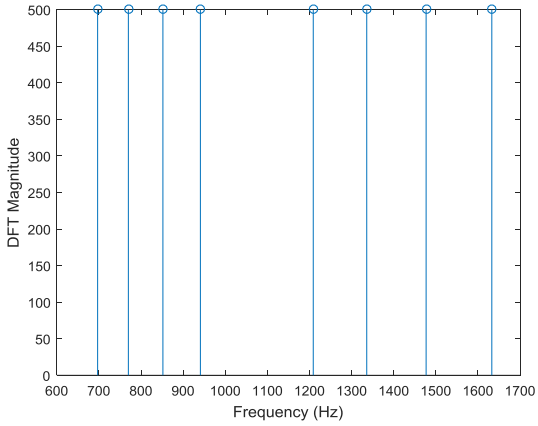
2.1.21 sygnał DTMF '159D' int. pchip - 44.1 kHz



2.1.22 sygnał DTMF ‘159D’ – 44.1 kHz



2.1.23 sygnał DTMF '159D' int. spline - 44.1 kHz



2.1.24 prążki DTMF ‘159D’ – alg. Goertzela

	‘1’	‘15’	‘159’	‘159D’
linear	0.030107	0.0237	0.021678	0.021146
pchip	0.016335	0.013248	0.012208	0.012136
spline	0.0006312	0.00059647	0.00066989	0.00081932

2.1.2 Błąd interpolacji tonów DTMF - sqrt(MSE)

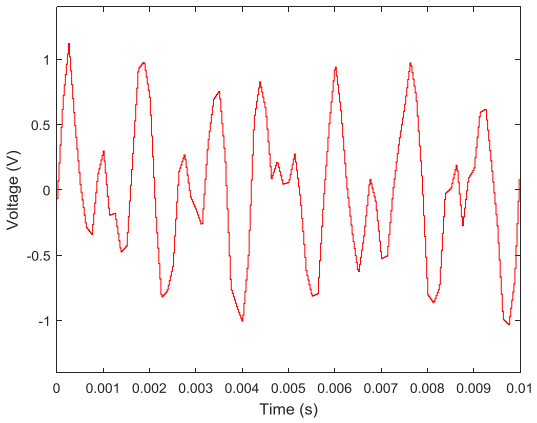
2.2. Interpolacja tonów DTMF na tle białego szumu

W tym eksperymencie do wygenerowanych w pkt. 2.1 tonów DTMF dodano biały szum gaussowski na poziomie SNR = 10. Celem tej operacji było zaburzenie ciągłości interpolowanych sygnałów, co powinno mieć znaczący wpływ na metody `'pchip'` oraz `'spline'`, które zachowują ciągłość do pierwszej pochodnej krzywej interpolacyjnej przez co krzywa ta może znacząco odbiegać od sygnału wejściowego. Drugim celem było sprawdzenie na ile tak wygenerowane krzywe (na podstawie próbek 8 kHz zawierających dodatkowo biały szum) będą odległe od referencyjnych sygnałów nie zawierających zakłóceń – innymi słowy na ile interpolacje `'pchip'` i `'spline'` mogą zadziałać jako częściowy filtr usuwający szum z sygnału audio.

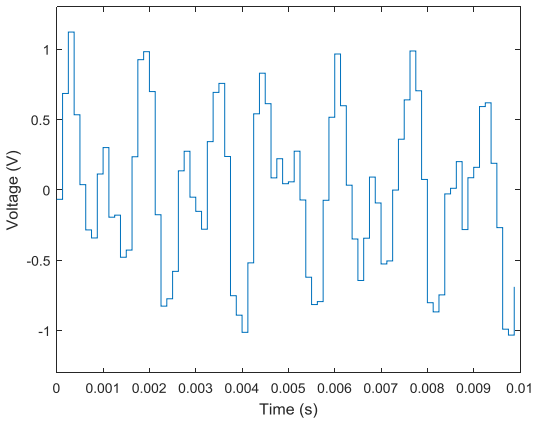
Do wygenerowania białego szumu gaussowskiego wykorzystano funkcję `awgn` dostępną w pakiecie MATLAB „Communications System Toolbox”.

Błędy poszczególnych metod interpolacji dla zadanych 8 kHz próbek tonów DTMF zawierających biały szum zebrano w tabelach na końcu tego podrozdziału. Jako miarę jakości uzyskanych sygnałów wykorzystano wartość pierwiastka kwadratowego z błędu średniokwadratowego wyliczanego względem wygenerowanych referencyjnych próbek tonów DTMF z częstotliwością próbkowania 44,1 kHz zawierających szum (tabela 2.2.1) i próbek wygenerowanych w pkt. 2.1 nie zawierających szumu (tabela 2.2.2).

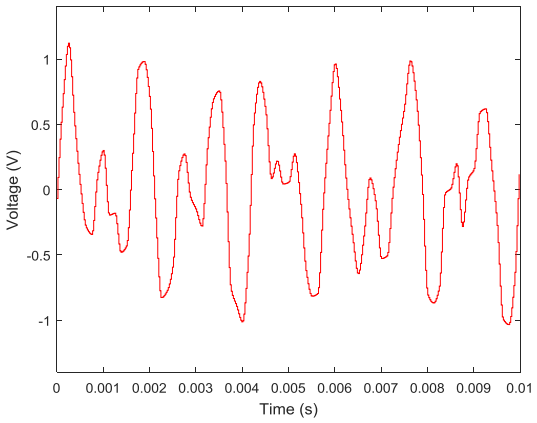
Dodatkowo w tabeli 2.2.3 zebrano wartość SNR dla nowo wygenerowanych próbek poszczególnymi metodami interpolacyjnymi względem sygnałów nie zawierających szumu.



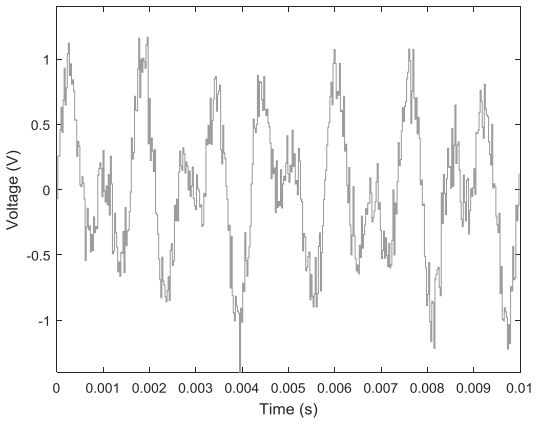
2.2.1 sygnał N+DTMF '1' int. liniowa - 44.1 kHz



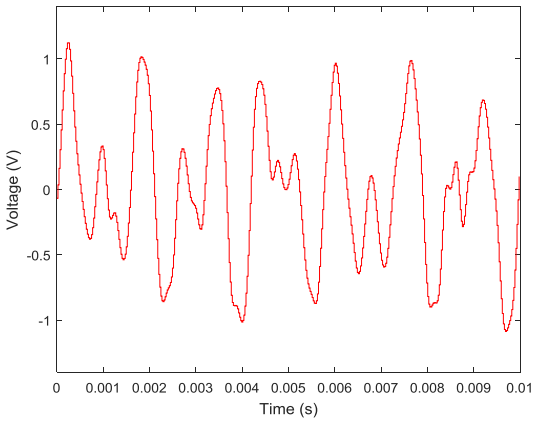
2.2.2 sygnał N+DTMF ‘1’ - 8 kHz



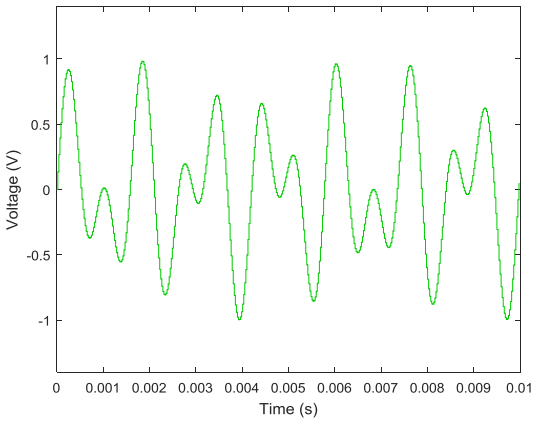
2.2.3 sygnał N+DTMF '1' int. pchip - 44.1 kHz



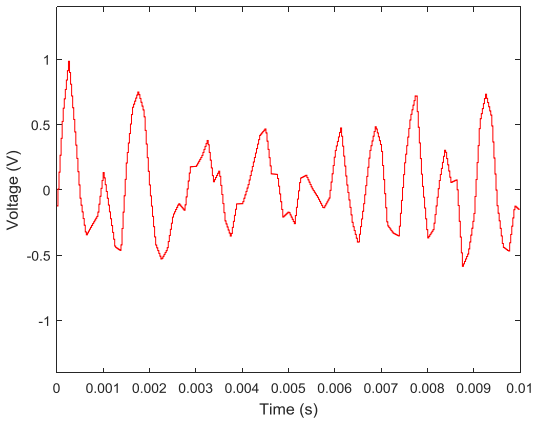
2.2.4 sygnał N+DTMF ‘1’ – 44.1 kHz



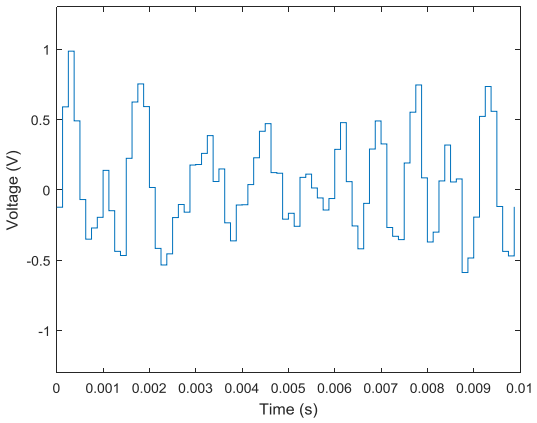
2.2.5 sygnał N+DTMF '1' int. spline - 44.1 kHz



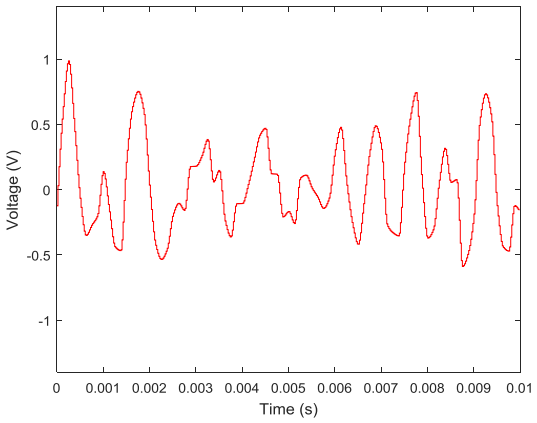
2.2.6 sygnał DTMF ‘1’ – 44.1 kHz



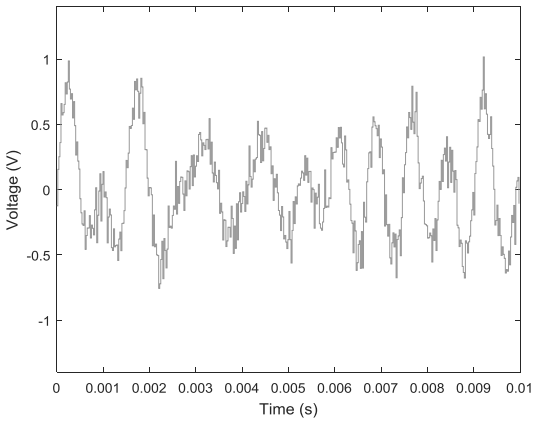
2.2.7 sygnał N+DTMF '15' int. liniowa - 44.1 kHz



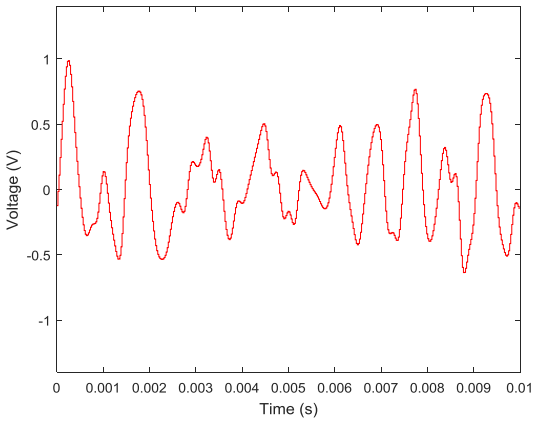
2.2.8 sygnał N+DTMF ‘15’ - 8 kHz



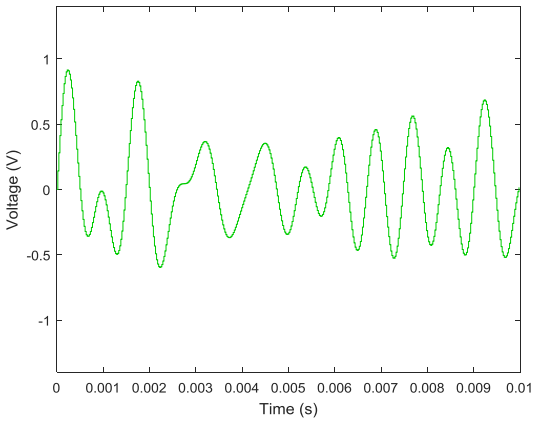
2.2.9 sygnał N+DTMF '15' int. pchip - 44.1 kHz



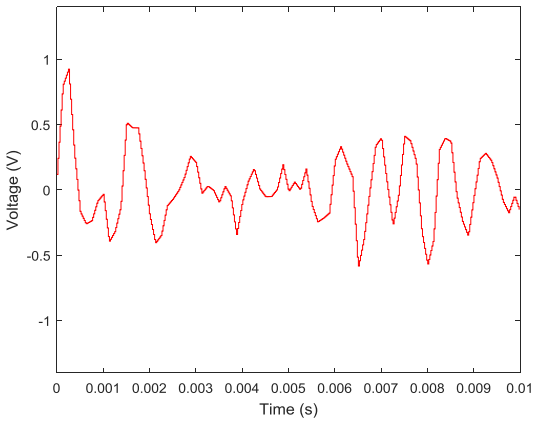
2.2.10 sygnał N+DTMF ‘15’ – 44.1 kHz



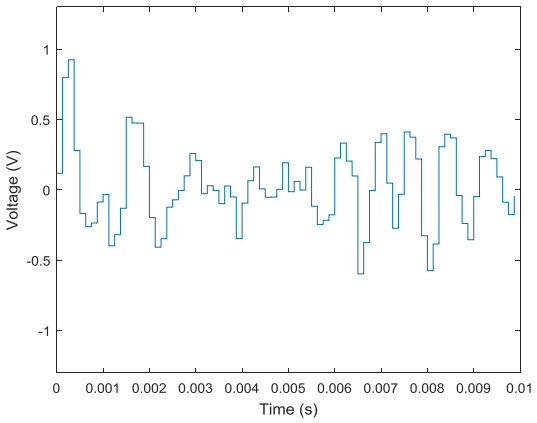
2.2.11 sygnał N+DTMF '15' int. spline - 44.1 kHz



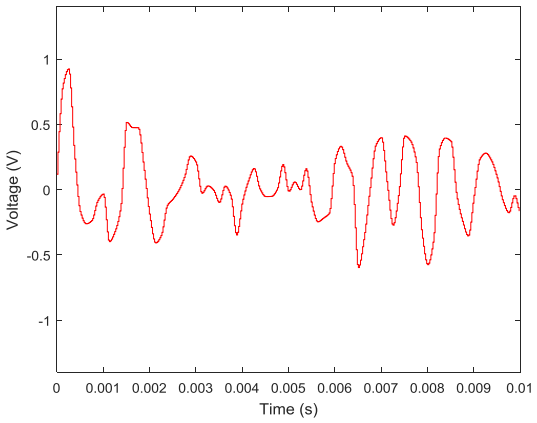
2.2.12 sygnał DTMF ‘15’ – 44.1 kHz



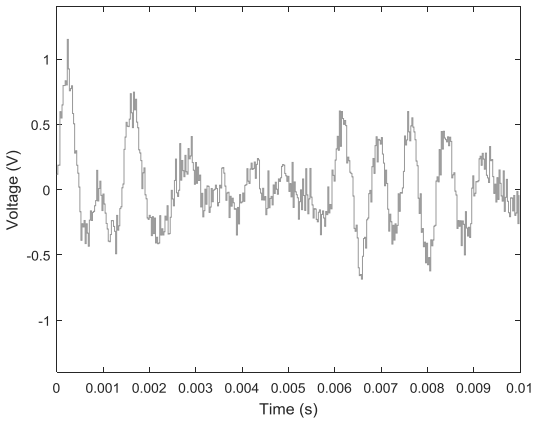
2.2.13 sygnał N+DTMF '159' int. liniowa - 44.1 kHz



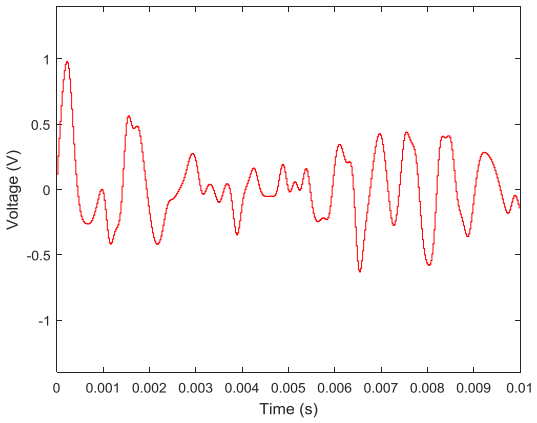
2.2.14 sygnał N+DTMF ‘159’ - 8 kHz



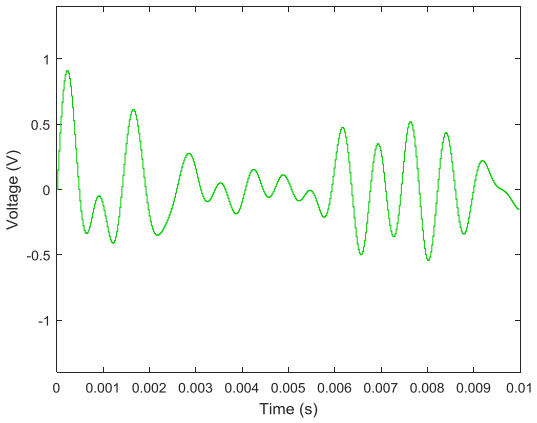
2.2.15 sygnał N+DTMF '159' int. pchip - 44.1 kHz



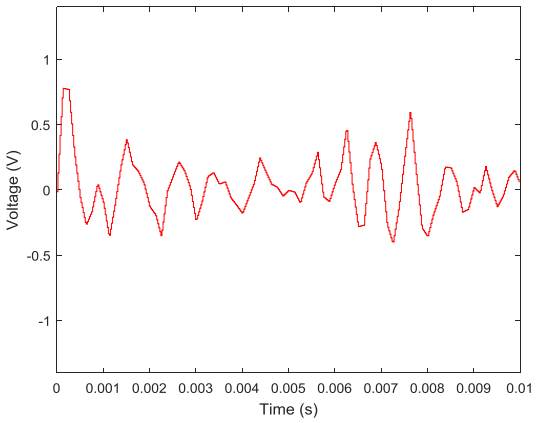
2.2.16 sygnał N+DTMF ‘159’ – 44.1 kHz



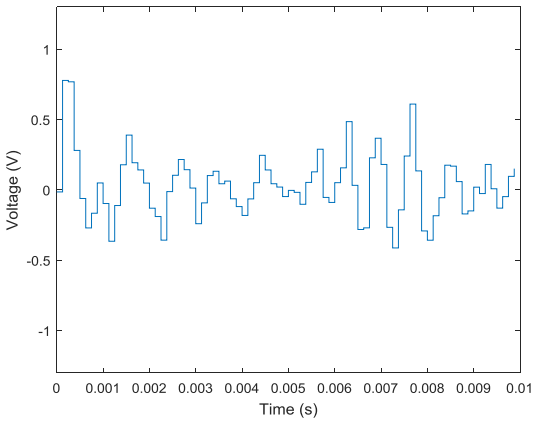
2.2.17 sygnał N+DTMF '159' int. spline - 44.1 kHz



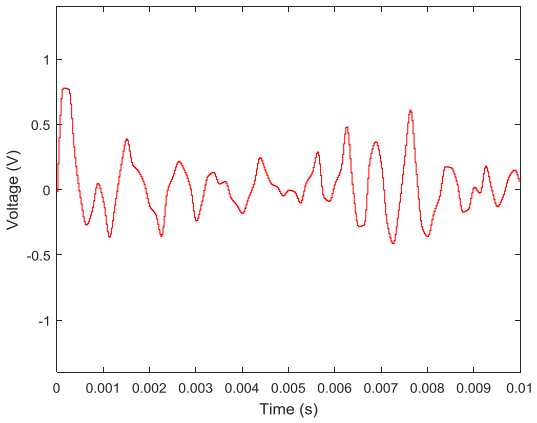
2.2.18 sygnał DTMF ‘159’ – 44.1 kHz



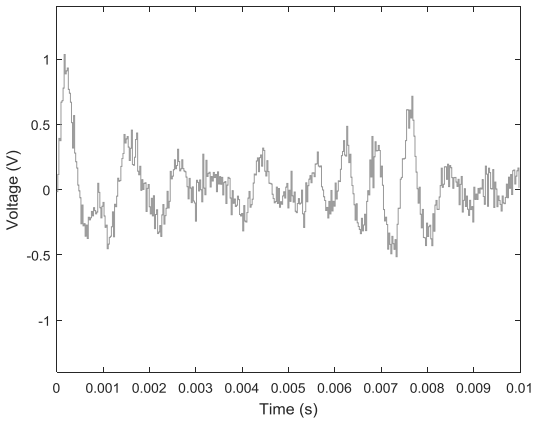
2.2.19 sygnał N+DTMF '159D' int. liniowa - 44.1 kHz



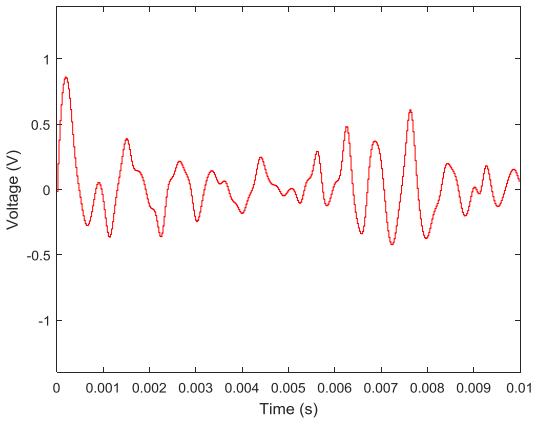
2.2.20 sygnał N+DTMF ‘159D’ - 8 kHz



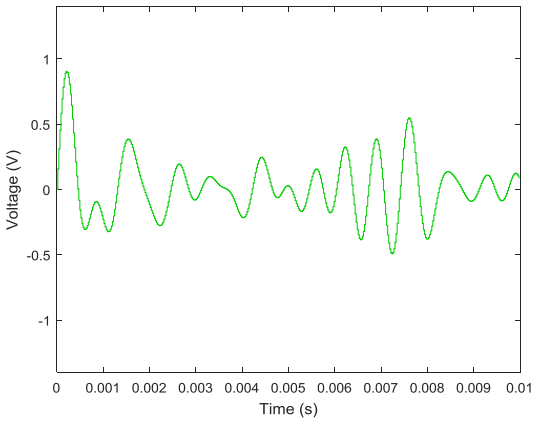
2.2.21 sygnał N+DTMF '159D' int. pchip - 44.1 kHz



2.2.22 sygnał N+DTMF ‘159D’ – 44.1 kHz



2.2.23 sygnał N+DTMF '159D' int. spline - 44.1 kHz



2.2.24 sygnał DTMF ‘159D’ – 44.1 kHz

	‘1’	‘15’	‘159’	‘159D’
linear	0.18649	0.13029	0.10826	0.094144
pchip	0.19304	0.13469	0.11127	0.096599
spline	0.19689	0.13746	0.11331	0.098429

2.2.1 Błąd interpolacji tonów DTMF względem sygnału zawierającego szum - sqrt(MSE)

	‘1’	‘15’	‘159’	‘159D’
linear	0.13327	0.09391	0.079331	0.068963
pchip	0.14343	0.10092	0.084141	0.072897
spline	0.14877	0.10465	0.086865	0.075287

2.2.2 Błąd interpolacji tonów DTMF względem sygnału nie zawierającego szumu - sqrt(MSE)

	‘1’	‘15’	‘159’	‘159D’
linear	11.402	11.357	11.287	11.201
pchip	10.78	10.766	10.73	10.72
spline	10.466	10.461	10.443	10.451

2.2.3 Wartość SNR dla interpolowanych tonów DTMF

3. Obserwacje i wioski

W przypadku prostych sygnałów, będących składowymi funkcji ciągłych, co najmniej klasy C^1 (w rozpatrywanym przypadku funkcji trygonometrycznych), metody interpolacyjne wykorzystujące wielomiany trzeciego stopnia (w szczególności interpolacja `'spline'`) dają najlepsze rezultaty – generują krzywe najbardziej zbliżone do sygnału referencyjnego. Wydaje się, że wraz ze wzrostem ilości funkcji składowych sygnału zbieżność rośnie. W rozpatrywanym przypadku, sygnału składającego się z 8 sinusoidalnych fal współczynnik MSE osiągnął wartość rzędu 10^{-4}. Nie zbadano ani też nie próbowano wyliczyć jaka jest maksymalna ilość funkcji składowych dla których ta tendencja będzie się utrzymywać.

W przypadku sygnałów zawierających składowe nieciągłe (tutaj było to zakłócenie sygnału szumem Gaussa) i próby odtworzenia sygnału oryginalnego, nie zawierającego szumów, jeśli jako kryterium stosować wartość MSE to najlepsze właściwości zdaje się mieć interpolacja liniowa, chociaż tutaj różnica w wartości MSE dla poszczególnych metod jest rzędu 10^{-2}.

Pewne wyjaśnienie może tu wnieść wprowadzenie pomiaru wartości SNR, która wskazuje, że metody interpolacyjne wykorzystujące wielomiany 3-stopnia nie podbijają wartości SNR – do próbek wprowadziliśmy szum na poziomie SNR = 10. Ponadto odsłuchanie próbek pokazuje, że dane wygenerowane metodami `'linear'` i `'pchip'` wprowadzają dodatkowy metaliczny „podźwięk”. Próbki wygenerowane metodą `'spline'` generują „miękki” dźwięk, najbardziej zbliżony do referencyjnych próbek 44,1 kHz nie zawierających szumu.

4. Kod źródłowy

```
function [t,v] = DTMF(digit, time, fs)

%   TONES      '0'  '1'  '2'  '3'  '4'  '5'  '6'  '7'  '8'  '9'  'a'  'b'  'c'  'd'  '*'  '#'
low_tones = [  941  697  697  697  770  770  770  852  852  852  697  770  852  941  941  941
];
hi_tones =  [ 1336 1209 1336 1477 1209 1336 1477 1209 1336 1477 1633 1633 1633 1633 1209 1477
];

lfs = low_tones(digit+1) * 2*pi;
hfs = hi_tones(digit+1) * 2*pi;

t = 0:1/fs:time-1/fs;
v = (sin(lfs*t) + sin(hfs*t)) / 2;

end

function lab1

%
% Generacja tonów DTMF
%

[t1_8k, d1_8k] = DTMF(1, 1, 8000);
[t1_8k, d5_8k] = DTMF(5, 1, 8000);
[t1_8k, d9_8k] = DTMF(9, 1, 8000);
[t1_8k, dD_8k] = DTMF(13, 1, 8000);

d15_8k = (d1_8k + d5_8k) / 2;
d159_8k = (d1_8k + d5_8k + d9_8k) / 3;
d159D_8k = (d1_8k + d5_8k + d9_8k + dD_8k) / 4;

[t1_44k, d1_44k] = DTMF(1, 1, 44100);
[t1_44k, d5_44k] = DTMF(5, 1, 44100);
[t1_44k, d9_44k] = DTMF(9, 1, 44100);
[t1_44k, dD_44k] = DTMF(13, 1, 44100);

d15_44k = (d1_44k + d5_44k) / 2;
d159_44k = (d1_44k + d5_44k + d9_44k) / 3;
d159D_44k = (d1_44k + d5_44k + d9_44k + dD_44k) / 4;

dtmf_snd = { d1_8k, d15_8k, d159_8k, d159D_8k;
             d1_44k, d15_44k, d159_44k, d159D_44k };

TV = [ 0.00 0.01 -1.4 1.4 ];

%
% Wizualizacja sygnałów DTMF
%

for i = 1 : 1 : size(dtmf_snd,2)
    figure(2*i - 1);
    stairs(t1_8k(1:0.01*8000),dtmf_snd{1,i}(1:0.01*8000)), axis([0 0.01 -1.3 1.3]);
    xlabel('Time (s)');
    ylabel('Voltage (V)');

    figure(2*i);
    stairs(t1_44k(1:0.01*44100),dtmf_snd{2,i}(1:0.01*44100),'Color',[0 0.8 0]), axis(TV);
    xlabel('Time (s)');
    ylabel('Voltage (V)');
end

%
% Detekcja tonów DTMF algorytmem Goertzel'a wraz z wizualizacją
%

f = [697 770 852 941 1209 1336 1477 1633];
freq_indices = f + 1;

for i = 1 : 1 : size(dtmf_snd,2)
    dft_data = goertzel(dtmf_snd{1,i},freq_indices);

    figure(10 + (i-1));
    stem(f,abs(dft_data)), axis([600 1700 0 inf]);
```

```
    xlabel('Frequency (Hz)');
    ylabel('DFT Magnitude');
end

%
% Interpolacja metodami 'linear'(1), 'pchip'(2), 'spline'(3) próbek DTMF
% 8kHz -> 44.1kHz wraz z wizualizacją
%

dtmf_int = cell(3, size(dtmf_snd,2));

for i = 1 : 1 : size(dtmf_int,2)
    dtmf_int{1,i} = interp1(t1_8k, dtmf_snd{1,i}, t1_44k, 'linear');
    dtmf_int{2,i} = interp1(t1_8k, dtmf_snd{1,i}, t1_44k, 'pchip');
    dtmf_int{3,i} = interp1(t1_8k, dtmf_snd{1,i}, t1_44k, 'spline');

    for j = 1 : 1 : size(dtmf_int,1)
        figure(20 + (j-1) + 3*(i-1));
        stairs(t1_44k(1:0.01*44100),dtmf_int{j,i}(1:0.01*44100),'Color','red'), axis(TV);
        xlabel('Time (s)');
        ylabel('Voltage (V)');
    end
end

%
% Obliczenie wartości błędu średniokwadratowego uzyskanych interpolacji
% metodą 'linear'(1), 'pchip'(2), 'spline'(3) wraz tabelaryzacją
%

mse_int = cell(size(dtmf_int));

for i = 1 : 1 : size(mse_int,2)
    for j = 1 : 1 : size(mse_int,1)
        mse_int{j,i} = sqrt(mean((dtmf_snd{2,i}(1:44095) - dtmf_int{j,i}(1:44095)).^2));
    end
end

T1 = cell2table(mse_int,'VariableNames',{'DTMF1' 'DTMF15' 'DTMF159'
'DTMF159D'},'RowNames',{'linear' 'pchip' 'spline'})

%
% Dodanie do próbek DTMF białego szumu gaussowskiego
%

dtmf_nse = cell(size(dtmf_snd));

for i = 1 : 1 : size(dtmf_nse,2)
    dtmf_nse{2,i} = awgn(dtmf_snd{2,i}, 10, 'measured');
    dtmf_nse{1,i} = interp1(t1_44k, dtmf_nse{2,i}, t1_8k, 'nearest');

    figure(40 + 2*(i-1));
    stairs(t1_8k(1:0.01*8000),dtmf_nse{1,i}(1:0.01*8000)), axis([0 0.01 -1.3 1.3]);
    xlabel('Time (s)');
    ylabel('Voltage (V)');

    figure(40+ 2*(i-1) + 1);
    stairs(t1_44k(1:0.01*44100),dtmf_nse{2,i}(1:0.01*44100),'Color',[0.6 0.6 0.6]), axis(TV);
    xlabel('Time (s)');
    ylabel('Voltage (V)');
end

%
% Interpolacja metodami 'linear'(1), 'pchip'(2), 'spline'(3) próbek DTMF
% z dodanym białym szymem gaussowskim 8kHz -> 44.1kHz wraz z wizualizacją
%

dtmf_nsi = cell(3, size(dtmf_nse,2));

for i = 1 : 1 : size(dtmf_nsi,2)
    dtmf_nsi{1,i} = interp1(t1_8k, dtmf_nse{1,i}, t1_44k, 'linear');
    dtmf_nsi{2,i} = interp1(t1_8k, dtmf_nse{1,i}, t1_44k, 'pchip');
    dtmf_nsi{3,i} = interp1(t1_8k, dtmf_nse{1,i}, t1_44k, 'spline');

    for j = 1 : 1 : size(dtmf_nsi,1)
        figure(50 + (j-1) + 3*(i-1));
        stairs(t1_44k(1:0.01*44100),dtmf_nsi{j,i}(1:0.01*44100),'Color','red'), axis(TV);
        xlabel('Time (s)');
```

```
        ylabel('Voltage (V)');
    end
end

%
% Obliczenie wartości błędu średniokwadratowego uzyskanych interpolacji
% metodą 'linear'(1), 'pchip'(2), 'spline'(3) dla próbek zawierających szum
% wraz tabelaryzacją
%

mse_nsi = cell(size(dtmf_nsi));
mse_nse = cell(size(dtmf_nsi));

for i = 1 : 1 : size(dtmf_nsi,2)
    for j = 1 : 1 : size(dtmf_nsi,1)
        mse_nsi{j,i} = sqrt(mean((dtmf_snd{2,i}(1:44095) - dtmf_nsi{j,i}(1:44095)).^2));
        mse_nse{j,i} = sqrt(mean((dtmf_nse{2,i}(1:44095) - dtmf_nsi{j,i}(1:44095)).^2));
    end
end

T2 = cell2table(mse_nsi,'VariableNames',{'DTMF1' 'DTMF15' 'DTMF159'
'DTMF159D'},'RowNames',{'linear' 'pchip' 'spline'})
T3 = cell2table(mse_nse,'VariableNames',{'DTMF1' 'DTMF15' 'DTMF159'
'DTMF159D'},'RowNames',{'linear' 'pchip' 'spline'})

%
% Obliczenie wartości SNR dla uzyskanych interpolacji metodą 'linear'(1),
% 'pchip'(2), 'spline'(3) z próbek zawierających szum względem "czystego"
% sygnału wraz tabelaryzacją
%

snr_nsi = cell(size(dtmf_nsi));

for i = 1 : 1 : size(dtmf_nsi,2)
    for j = 1 : 1 : size(dtmf_nsi,1)
        snr_nsi{j,i} = snr(dtmf_snd{2,i}(1:44095), dtmf_snd{2,i}(1:44095) -
dtmf_nsi{j,i}(1:44095));
    end
end

T4 = cell2table(snr_nsi,'VariableNames',{'DTMF1' 'DTMF15' 'DTMF159'
'DTMF159D'},'RowNames',{'linear' 'pchip' 'spline'})

% sound(dtmf_snd{1,1}, 8000);
% pause(2);
% sound(dtmf_int{1,1}, 44100);
% pause(2);
% sound(dtmf_int{2,1}, 44100);
% pause(2);
% sound(dtmf_int{3,1}, 44100);
% pause(2);
% sound(dtmf_snd{2,1}, 44100);

end
```