Politechnika Łódzka
Instytut Elektroniki

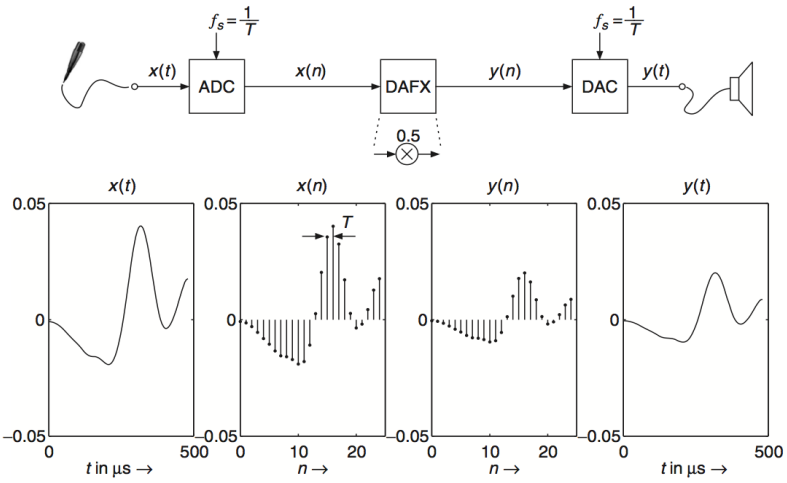
Laboratorium
Inżynierii akustycznej

Przetwarzanie dźwięku - wprowadzenie do efektów dźwiękowych, realizacja opóźnień

Wstęp teoretyczny:

Przetwarzanie sygnału dźwiękowego wiąże się bezpośrednio ze znajomością odpowiedniej konwersji sygnału analogowego. Proces, o którym mowa nosi nazwę konwersji analogowo-cyfrowej ADC (ang. analog-to-digital). ADC to proces przetwarzania sygnału analogowego x(t) na jego ekwiwalentną reprezentację cyfrową x(n). Czas pomiędzy kolejnym pobieraniem wartości analogowej reprezentującą cyfrowo dany sygnał nosi nazwę interwału próbkowania - w literaturze oraz w dalszej części będzie mowa o odwrotności tego czasu zwanego częstotliwością próbkowania (frequency sampling). Nawiązując do teorii zgodnie z twierdzeniem Nyquist'a częstotliwość próbkowania musi być dwukrotnie większa od częstotliwości sygnału analogowego.

System przetwarzania dźwięku opiera się o operacje matematyczne już na cyfrowej postaci sygnału dźwiękowego, co zostanie omówione w dalszej części na przykładach. Diagram poniżej przedstawia proces kwantyzacji sygnału analogowego, przetwarzanie matematyczne sygnału (w tym przypadku redukcja amplitudy o połowę) - traktowane dalej jak cyfrowy efekt dźwiękowy, a następnie rekonstrukcja sygnału cyfrowego do postaci analogowej, dzięki której jest możliwość usłyszenia generowanego dźwięku przez głośnik.



Próbkowanie i kwantyzacja sygnału analogowego (ADC), cyfrowe efekty dźwiękowe (DAFX), rekonstrukcja sygnału analogowego (DAC).

Ten prosty przykład można zaimplementować z użyciem środowiska MATLAB, oraz posługując się poniższym kodem.

```
---------------------------------------------------------------------
% Wczytanie pliku dźwiękowego, jego reprezentacja cyfrowa zawiera
wektor x(n) i częstotliwość próbkowania - FS
[x,FS]=wavread('input.wav');
y = x*2;

% Zapis wektora y(n) jako plik dźwiękowy z częstotliwością
próbkowania FS i głębią bitową Nbits za pomocą funkcji
wavwrite(y,FS,Nbits,'output.wav');
---------------------------------------------------------------------
```

Analiza częstotliwościowa, filtracja i opóźnienia:

Analiza sygnałów dźwiękowych najczęściej sprowadza się do ich analizy częstotliwościowej. Kluczowym zagadnieniem jest tutaj widmo sygnałów dyskretnych, które polega na transformacji całkowej sygnału z dziedziny czasu w dziedzinę częstotliwości. Transformata jest wynikiem transformacji Fouriera, która rozkłada funkcję na szereg funkcji okresowych tak, że uzyskany wynik określa jakie częstotliwości składają się na pierwotną funkcję [1]. Transformacja Fouriera jest podstawowym narzędziem analizy częstotliwościowej sygnałów. Prosta i odwrotna transformacja Fouriera zdefiniowane są wzorami:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt$$

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{-j2\pi ft} df$$

Filtracja sygnałów to ogół działań mających na celu przepuszczenie lub blokowanie sygnałów o określonym zakresie częstotliwości lub zawierających odpowiednie dla nas harmoniczne. Ze względu na szeroki zakres tego zagadnienia zostanie przedstawiony podział filtracji sygnałów dyskretnych przy użyciu komputera.

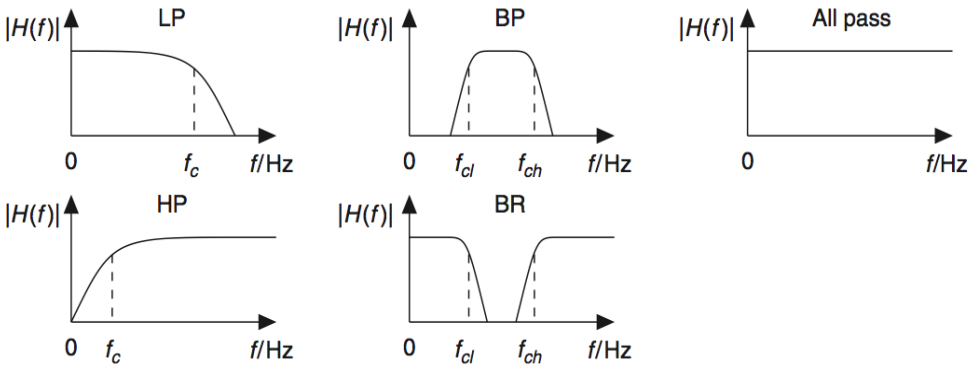
W przypadku sygnałów cyfrowych korzysta się z transformacji Fouriera dla sygnałów dyskretnych określonych parą równań:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}, k = 0,1,2,...,N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn}, n = 0,1,2,...,N-1$$

Jeżeli chodzi o filtrację sygnałów dźwiękowych ważna jest tutaj znajomość podziału filtrów w zależności od zastosowania. Rodzaj przestawionych filtrów częstotliwościowych został również zilustrowany na przykładach poniżej.

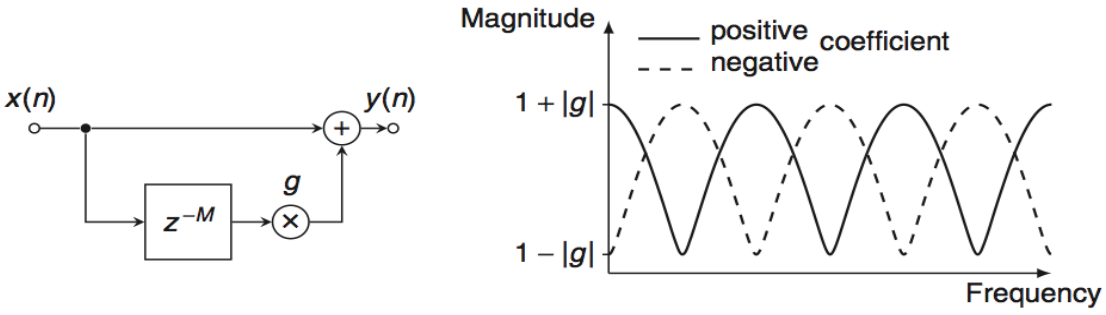
1. LP - Filtr dolnoprzepustowy
2. BP - Filtr pasmowoprzepustowy
3. HP - Filtr górnoprzepustowy
4. BR - Filtr pasmowozaporowy
5. All pass - Filtr wszechprzepustowy



Filtr dolnoprzepustowy jest bardzo często stosowany w muzyce komputerowej - symulacjach akustycznych struktur rezonansowych, filtr górnoprzepustowy pomaga usunąć niechciane niskie tony. Używając filtru pasmowo przepustowego można osiągnąć dźwięk naśladujący jakość rozmowy telefonicznej lub pozwala wyciszyć pasmo częstotliwości dla danego instrumentu. Filtracja pasmowo zaporowa umożliwia np. podział spektrum dla dwóch różnych zespołów które wydają się być skorelowane ze sobą.

1. Filtr grzebieniowy, FIR

Filtr grzebieniowy w najprostszym ujęciu jest to sieć symulująca opóźnienie, sygnału wejściowego o daną jednostkę czasu. Efekt ten jest słyszalny wówczas gdy tak przetworzony sygnał jest mieszany z sygnałem wyjściowym, który można w tym przypadku potraktować jako odniesienie. Efekt opóźnienia opisują dwa parametry - czas opóźnienia T oraz względna amplituda sygnału opóźnionego do sygnału odniesienia. Poniżej przedstawiono blokowo strukturę filtru grzebieniowego.



Równanie różnicowe oraz funkcja przenoszenia :

$$y(n) = x(n) + gx(n - M)$$

$$\text{with} \quad M = \tau / f_s$$

$$H(z) = 1 + gz^{-M}.$$

Odpowiedz czasowa filtru składa się z sygnału bezpośredniego oraz wersji sygnału z opóźnieniem. To względnie proste działanie w dziedzinie czasu wynika z bardziej skomplikowanego mechanizmu w dziedzinie częstotliwości. Dla dodatnich wartości g, filtr wzmacnia wszystkie częstotliwości, które są wielokrotnościami 1/T i tłumi wszystkie częstotliwości, które znajdują się między nimi. Funkcja przenoszenia takiego filtru przedstawiono jako szereg kolców co do złudzenia przypomina grzebień, stąd pochodzi nazwa filtru grzebieniowego. Dla ujemnych wartości f, filtr tłumi częstotliwości które są wielokrotnościami 1/T i wzmacnia te, które znajdują się pomiędzy nimi. Przyrost waga się między 1+g, oraz 1-g. Poniżej przedstawiono kod źródłowy implementacji filtru grzebieniowego.

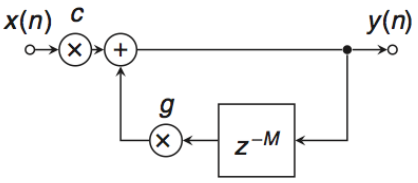
```
x=zeros(100,1);x(1)=1; % unit impulse signal of length 100

g=0.5;
Delayline=zeros(10,1);% memory allocation for length 10

for n=1:length(x);

      y(n)=x(n)+g*Delayline(10);

      Delayline=[x(n);Delayline(1:10-1)];

end;
```

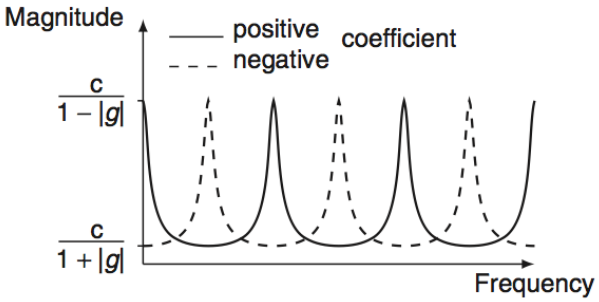
Podobnie jak w przypadku akustycznych opóźnień FIR - filtr grzebieniowy ma wpływ zarówno w dziedzinie czasu jak i częstotliwości. Nasze ucho jest bardziej wrażliwe na jeden bodziec lub na wiele w zależności od czasu opóźnienia. W przypadku dużych wartości T, można usłyszeć echo, które jest wyraźne na tle sygnału macierzystego. Częstotliwości które są wzmacniane przez filtr grzebieniowy są położone tak blisko siebie, że z trudnością można zidentyfikować efekt spektralny. Dla mniejszych wartości T nasze ucho nie może odseparować poszczególnych wydarzeń dźwiękowych w czasie ale może dostrzec efekt spowodowany filtracją sygnału.

2. Filtr grzebieniowy, IIR

Filtr grzebieniowy IIR jest to sieć opóźnienia ze sprzężeniem zwrotnym w którym występuje oddziaływanie sygnału wyjściowego na sygnał wejściowy. Sygnał wyjściowy jest złożeniem sygnału wyjściowego przetworzonego oraz sygnału wejściowego.

Za każdym razem, gdy sygnał przechodzi przez linię opóźnienia jest osłabiany przez ***g***. Czasami konieczne jest do skalowania sygnału wejściowego przez parametr ***c***, aby zrekompensować wysokie wzmocnienie produkowane przez strukturę sprzężenia zwrotnego. Struktura jest zilustrowana na rysunku poniżej





$$y(n) = cx(n) + gy(n - M), \text{ with } M = \tau/f_s$$

$$H(z) = c/(1 - gz^{-M}).$$

Ze względu na sprzężenie zwrotne, czas reakcji filtru, jest nieskończony. Po każdym opóźnieniu o czasie ***T*** kopię sygnału wyjściowego z amplitudą g^P, gdzie ***p*** jest liczbą cykli którą sygnał przeszedł przez linię czasu. Można łatwo zauważyć, że | g | ≤ 1 jest warunkiem stabilności. W przeciwnym wypadku sygnał będzie rosnąć w nieskończoność. Częstotliwości, których dotyczy filtr grzebieniowy IIR są podobne do tych, których dotyczy filtr grzebieniowy FIR. Przyrost waha się od 1 / (1 - g) i 1 / (1 + g). Głównymi różnicami pomiędzy filtrem IIR i FIR jest to, iż wzmocnienie rośnie bardzo wysoko, a maksymalne amplitudy częstotliwości uzyskuje się węższe gdy | g | zbliża się do 1. Poniżej przedstawiono przykład implementacji filtru grzebieniowego IIR.

```
x=zeros(100,1);

x(1)=1; % unit impulse signal of length 100

g=0.5;
Delayline=zeros(10,1); % memory allocation for length 10

for n=1:length(x);

        y(n)=x(n)+g*Delayline(10);

        Delayline=[y(n);Delayline(1:10-1)];

end;
```

---

3. Filtr grzebieniowy uniwersalny

Połączenie filtru grzebieniowego typu FIR i IIR prowadzi do powszechnego filtru grzebieniowego. Struktura filtru oraz jego implementacja przedstawiona została poniżej.

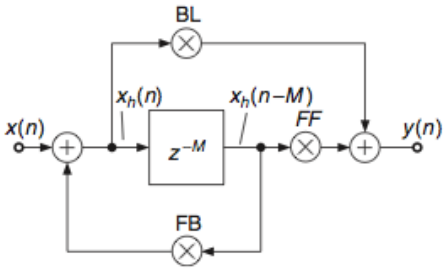


Figure 2.29 Universal comb filter.

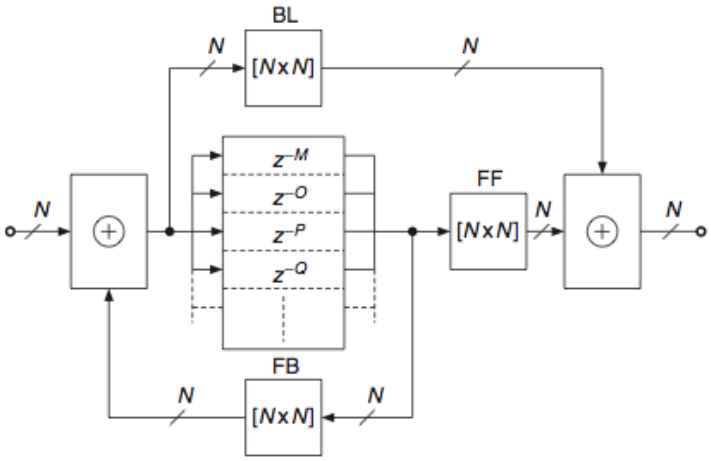
Table 2.6 Parameters for universal comb filter.

	BL	FB	FF
FIR comb filter	1	0	g
IIR comb filter	c	g	0
Allpass	a	$-a$	1
Delay	0	0	1

```
x=zeros(100,1);
x(1)=1; % unit impulse signal of length 100
BL=0.5;
FB=-0.5;
FF=1;
M=10;
Delayline=zeros(M,1);  % memory allocation for length 10 for
n=1:length(x);
    xh=x(n)+FB*Delayline(M);
    y(n)=FF*Delayline(M)+BL*xh;
    Delayline=[xh;Delayline(1:M-1)];
end;
```

Istnieje możliwość rozszerzenia filtru grzebieniowego uniwersalnego poprzez zrównoleglenie połączeń N filtrów co zostało przedstawione w następnym rozdziale. Prowadzi to do NxN macierzy współczynników dla sygnałów wejściowych I wyjściowych ze sprzężenia. W zależności od wartości współczynników prowadzi to do uzyskania nowych efektów dźwiękowych bazujących na opóźnieniu sygnału (np. Slapback, Echo, Reverb)

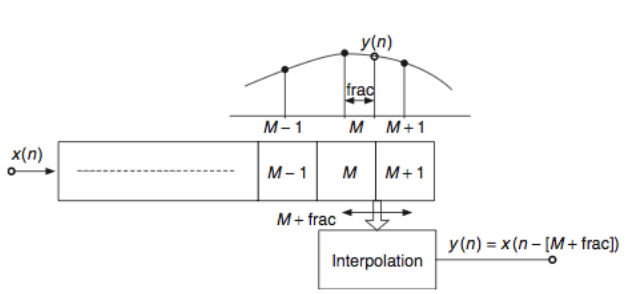
	Delay	BL	FB	FF
Slapback	50 ms	1	0	X
Echo	>50 ms	1	$0 < X < 1$	0
Reverb		Matrix	Matrix	Matrix



4. Opóźnienie ułamkowe – Fractional Delays

Zmiennej długości opóźnienia sygnału wejściowego są używane do symulacji różnych efektów akustycznych. Dlatego opóźnienia sygnału wejściowego o niecałkowitych wartościach w czasie pobierania próbki są niezbędne. Poniżej przedstawiono równanie opisujące opóźnienie sygnału wejściowego przez M liczbę próbek plus ułamkową część w przedziale 0 ≤ frac ≤1.

$$y(n) = x(n - [M + \text{frac}])$$



Poszukiwanie wartości próbek o niecałkowitych wartościach indeksu prowadzi do zastosowania interpolacji .Algorytm ten musi obliczyć wynik próbki y (n), który znajduje się w przedziale dla obu próbek w czasie o indeksie M i M + 1. Poniżej przedstawiono kilka algorytmów interpolacji, które zostały zaproponowane do aplikacji audio:

- Linear interpolation

$$y(n) = x(n - [M + 1])\text{frac} + x(n - M)(1 - \text{frac})$$

- Allpass interpolation

$$y(n) = x(n - [M + 1])\text{frac} + x(n - M)(1 - \text{frac}) - y(n - 1)(1 - \text{frac})$$

- Sinc interpolation
- Spline interpolation - trzeciego rzędu

$$\begin{aligned} y(n) = {} & x(n - [M + 1]) \cdot \frac{\text{frac}^3}{6} \\ & + x(n - M) \cdot \frac{(1 + \text{frac})^3 - 4 \cdot \text{frac}^3}{6} \\ & + x(n - [M - 1]) \cdot \frac{(2 - \text{frac})^3 - 4(1 - \text{frac})^3}{6} \\ & + x(n - [M - 2]) \cdot \frac{(1 - \text{frac})^3}{6}. \end{aligned}$$

Wybór algorytmu interpolacji zależy oczywiście od konkretnego zastosowania.

Zadania z użyciem środowiska MATLAB

1. Dla dowolnego pliku dźwiękowego mono spróbkowanego z częstotliwością 44100Hz, wykonać próbkowanie z częstotliwością 22050Hz, 5512.5Hz i 2756.25Hz. Zapisać wyniki do plików wav oraz m-plik realizujący tę funkcję.(Podpowiedź: można wykorzystać pętlę *for* oraz indeksowanie tablicy).

2. Dla dowolnego pliku dźwiękowego zmniejszyć głębię bitową odpowiednio do 8,4 i 2 bitów. Wyniki zapisać do plików wav oraz m-plik realizujący tę funkcję.(Podpowiedź: wykonując operację przesunięć bitowych oraz zaokrągleń na próbce, można w prosty sposób zmniejszyć głębię bitową).

3. Wygenerować sinusoidę o częstotliwości 5000Hz i amplitudzie 1. Częstotliwość próbkowania 20000Hz. Liczba wygenerowanych próbek powinna być dokładnie taka, aby wyświetlić 10 okresów sinusoidy.

*4. Wygenerować sygnał składający się z dwóch sinusoid. Jedna o amplitudzie 1 i częstotliwości 2000Hz a druga o amplitudzie 0.7 i częstotliwości 4300Hz. Częstotliwość próbkowania wynosi 44100Hz. Wyświetlić sygnał w dziedzinie czasu i częstliwości.(Podpowiedź: należy skorzystać z funkcji *fft, shiftfft, abs*).

5. Wykonać prosty filtr FIR dolnoprzepustowy z częstotliwością odcięcia 3kHz. Przefiltrować wybrany plik dźwiękowy i następnie go zapisać. Działanie filtru sprawdzić w programie audacity (spectogram) lub używając transformaty Fouriera w Matlabie. (Podpowiedź: zapoznać się z funkcjami *fir1, fft, shiftfft, filter*).

6. Napisać skrypt umożliwiający wczytanie dowolnego pliku muzycznego lub nagranie dźwięku bezpośrednio, zastosowanie filtru grzebieniowego FIR o dowolnych parametrach. Wykonać wykresy sygnału wejściowego I wyjściowego w dziedzinie czasu I częstotliwości.

*7. Zaimplementować filtr uniwersalny dla efektu Delay oraz IIR, przetestować działanie jak w poprzednim zadaniu.

*8. Zaimplementować efekt Chorus lub Flanger, zgodnie z blokowym schematem poniżej.

Delay range (ms) (Typ.)	Modulation (Typ.)	Effect name
0 ... 20	–	Resonator
0 ... 15	Sinusoidal	Flanging
10 ... 25	Random	Chorus
25 ... 50	–	Slapback
> 50	–	Echo