Uniwersytet Warszawski

Wydział Matematyki, Informatyki i Mechaniki

Wit Jakuczun

Nr albumu: 178839

Wojciech Žiška

Nr albumu: 171888

Projektowanie filtrów typu FIR o liniowej charakterystyce fazowej

Praca licencjacka na kierunku MATEMATYKA

Praca wykonana pod kierunkiem
dr hab. Marka Kowalskiego
Instytut matematyki stosowanej i mechaniki

czerwiec 2000

Pracę przedkładam do oceny

Podpis autora

Praca jest gotowa do oceny przez recenzenta

Data Podpis autora

Klasyfikacja 94A12

Słowa kluczowe sygnał, sygnał impulsowy, dyskretny system liniowy, splot, dyskretna transformata Fouriera (DFT), system typu FIR, overlap-add method, algorytm Remeza, aproksymacja jednostajna

Streszczenie Praca opisuje wykorzystanie metody Remeza do projektowania filtrów typu FIR o liniowej fazie. W rozdziale pierwszym przedstawiamy podstawy matematyczne tego zagadnienia. Definiujemy takie pojęcia jak przestrzenie ciągów, dyskretne systemy liniowe, impuls jednotkowy, odpowiedź impulsowa systemu oraz przyczynowość i stabilność. W rozdziale drugim opisujemy tzw. dyskretną transformatę Fouriera. Przedtawimy także kilka wniosków i pojęć związanych z przetwarzaniem sygnałów ciągłych za pomocą komputera. W rozdziale trzecim zamieszczamy opis ogólnej metody Remeza oraz jej modyfikację służącą do projektowania filtrów typu FIR. Przedostani rozdział zawiera krótki opis programu projektującego filtry typu FIR oraz przetwarzającego pliki dźwiękowe. W ostatnim rozdziale zamieściliśmy kod źródłowy programu[1] (język C++)

[1] Ze względu na rozmiar kodu zamieściliśmy jedynie treść najważniejszych funkcji

Spis treści

Rozdział 1

Dyskretne systemy liniowe

1.1 Wprowadzenie

Sygnał jest to funkcja dostarczająca informacje o genezie lub zmianie określonych zjawisk fizycznych. Może to być na przykład sygnał mowy, który poprzez mikrofon jest przekształcany ze swojej pierwotnej postaci, fali akustycznej, na funkcje wyrażającą zmiany wielkości elektrycznych w czasie.

W przyjmowanych modelach sygnału abstrahuje się od fizycznej jego natury i przez to pojęcie może być wyrażona dowolna funkcja, która zawiera informacje związane ze zjawiskiem fizycznym, ekonomicznym, psychologicznym.

Wyróżnia się następujące typy sygnałów:

- *sygnał ciągły* - sygnał określony dla „ciągłej wartości czasu“;
- *sygnał analogowy* - sygnał ciągły mający „ciągły zbiór wartości“;
- *sygnał dyskretny* - sygnał określony dla dyskretnej wartości czasu;
- *sygnał cyfrowy* - sygnał dyskretny mający dyskretny zbiór wartości.

W tej pracy będziemy się zajmowali sygnałami cyfrowymi, aczkolwiek wyjaśnimy związek miedzy tymi sygnałami a sygnałami ciągłymi. Każdy sygnał ciągły, który chcemy analizować techniką1cyfrową musimy najpierw *zdyskretyzować*. W wyniku tej operacji otrzymamy ciąg liczb, który będzie dyskretnym odpowiednikiem wejściowego sygnału. Ciąg taki możemy następnie przekształcić za pomocą pewnego operatora matematycznego zwanego *systemem dyskretnym*. W zależności od własności tego operatora będzie on nosił odpowiednie miano. My ograniczymy się jedynie do *dyskretnych systemów liniowych.*

Przykład 1.1.1. sygnał mowy - preemfaza

Układ zwany preemfazą jest wykorzystywany przy przetwarzaniu sygnalu mowy do uwypuklania większych częstotliwości w widmie sygnalu. Jeżeli zdyskretyzowany sygnał mowy oznaczymy przez $\{x(n)\}$ $(n = 0, \pm 1, \ldots)$, to system ten określony jest wzorem

$$L : y(n) = x(n) - \mu x(n-1) \tag{1.1}$$

Przykład 1.1.2. sygnał EKG - filtr typu FIR

Systemy cyfrowe wykorzystywane są również w usuwaniu zniekształceń w sygnale EKG i EEG spowodowanych np. drżeniem mięśni czy szumem źródła zasilania elektrokardiografu. Układ realizujący to zadanie może być określony następująco:

$$y(n) = \sum_{m=0}^{M-1} h(m)x(n-m)$$

gdzie x jest pierwotnym, zdyskretyzowanym sygnałem EKG, y natomiast jest sygnałem otrzymanym po przekształceniu. Współczynniki $h(m),\ m = 0, 1, \ldots, M-1$ jednoznacznie wyznaczają to przekształcenie.

1.2 Model matematyczny sygnału dyskretnego i dyskretnego systemu liniowego

Definicja 1.2.1. (sygnał dyskretny)

Sygnałem dyskretnym nazwiemy dowolny ciąg liczb rzeczywistych lub zespolonych

$$x \equiv \{x(n)\},\ -\infty < n < \infty$$

Symbol $x(n)$ będzie oznaczał n-ty wyraz ciągu x.

Definicja 1.2.2. (impuls jednostkowy)

Impulsem jednostkowym nazwiemy ciąg określony wzorem

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \tag{1.2}$$

Definicja 1.2.3. ciąg przesunięty o k wyrazów

Ciągiem przesuniętym o k wyrazów nazwiemy ciąg określony wzorem:

$$x_k(n) = x(n-k)\ \forall_n \tag{1.3}$$

gdzie x oznacza pewien dany ciąg.

Definicja 1.2.4. (przestrzenie ciągów)

- S - przestrzeń wszystkich ciągów
- $S_F = \{x : \exists_{N>0}\ x(n) = 0\ \forall_{|n|>N}\}$ - przestrzeń ciągów skończonych,
- $S_\infty = \{x : \exists_{N>0}\ x(n) = 0\ \forall_{n<-N}\}$ - przestrzeń ciągów mających gdzieś początek,
- $S_i = \{x : x(n) = 0\ \forall_{n<i}\}$ - przestrzeń ciągów mających początek w momencie i.

Definicja 1.2.5. (dyskretny system liniowy)

Liniowy operator $L\colon S_\infty \to S$ nazwiemy *dyskretnym systemem liniowym* wtedy i tylko wtedy, gdy operator ten ograniczony do przestrzeni S_0 jest ciągły względem metryki:

$$\rho(x,y) = \sum_{k=-\infty}^{\infty} 2^{-|k|} \frac{|x(k)-y(k)|}{1+|x(k)-y(k)|} \tag{1.4}$$

oraz spełnia warunek *niezmienności względem przesunięcia:*

$$\forall_{x,y\in S_\infty} L[x] = y \Rightarrow \forall_k\, L[x_k] = y_k \tag{1.5}$$

Wykazuje się, że operator $L\colon S_\infty \to S$ określony równaniem

$$y(n) = \begin{cases} 0 & \text{jeżeli } x(m) = 0 \quad \forall_{m\le n} \\ \sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) & \text{w przeciwnym przypadku} \end{cases} \tag{1.6}$$

(gdzie $y = L[x]$ i $\exists_k, b_k \neq 0$) jest systemem liniowym w sensie przyjętej definicji. Operator ten opisuje bardzo istotną pod względem praktycznym klasę układów liniowych i w niektórych pracach jest on przyjmowany jako określenie dyskretnego systemu liniowego.

Definicja 1.2.6.

Dla wygody późniejszych rozważań, wprowadzimy oznaczenie na ciąg będący odpowiedzią na impuls jednostkowy

$$L[\delta] = h \tag{1.7}$$

Uwaga 1.2.1.

Będziemy także chcieli aby, dyskretny system liniowy spełniał następującą własność

$$y = L[x] = L[\sum_{k=-\infty}^{\infty} x(k)\delta_k] = \sum_{k=-\infty}^{\infty} x(k) L[\delta_k] \tag{1.8}$$

Warto zauważyć, że powyższa własność dla dyskretnych systemów liniowych nie zmiennych względem przesunięcia można zapisać w następujący sposób:

$$y = \sum_{k=-\infty}^{\infty} x(k) h_k \tag{1.9}$$

Z powyższego równania wynika, że dyskretny system liniowy jest jednoznacznie określony przez ciąg będący odpowiedzią na impuls jednostkowy. Opisuje ono splot dwóch ciągów i jest oznaczane

$$y = x \star h \tag{1.10}$$

Definicja 1.2.7. (dwa rodzaje dyskretnych systemów liniowych)
Wyróżnia się dwa rodzaje dyskretnych systemów liniowych niezmiennych względem przesunięcia:

- system o **S**kończonej **O**dpowiedzi **I**mpulsowej ($h \in S_F$ w równaniu (1.10)). Oznacza się przez **FIR** (finite impulse response) lub **SOI.**
- system o **N**ieskończonej **O**dpowiedzi **I**mpulsowej ($h \notin S_F$ w równaniu (1.10)). Oznacza się przez **IIR** (infinite impulse response) lub **NOI.**

W pracy tej będziemy zajmowali się tylko systemami typu **FIR.**

1.3 Przyczynowość i stabilność

Przyczynowość jest podstawowym warunkiem realizowalności systemu liniowego. Dla systemu przyczynowego każda zmiana dowolnego sygnału wyjściowego musi być poprzedzona zmianą sygnału wejściowego.

Definicja 1.3.1. system przyczynowy
System $y = L[x]$ nazwiemy *przyczynowym* jeśli

$$\forall_{x \in D} \quad x \in S_i \Rightarrow y \in S_i \tag{1.11}$$

gdzie D jest dziedziną rozpatrywanych sygnałów.

Ciąg h będący odpowiedzią na impuls jednostkowy jednoznacznie określa, czy system jest przyczynowy, czy nie. Zachodzi następujące twierdzenie:

Twierdzenie 1.3.1.
System liniowy jest przyczynowy wtedy i tylko wtedy, gdy $h \in S_0$
(tzn. $\forall_{n<0} h(n) = 0$), gdzie h jest odpowiedzią systemu na impuls jednostkowy.

Dowód:
Załóżmy, że system jest przyczynowy. Ponieważ $\delta \in S_o$, to z (1.11) wynika, że $h \in S_0$.
Przypuśćmy teraz, że $h \in S_0$. Zakładając, że $x \in S_i$ mamy

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=i}^{\infty} x(k)h(n-k)$$

zatem

$$y(n) = \begin{cases} 0 & n < i \\ \sum_{k=i}^{n} x(k)h(n-k) & n \geq i \end{cases}$$

Wynika stąd, że $y \in S_i$, a więc jest spełniona zależność (1.11).

□

Inną ważną własnością jest stabilność systemu. Własność ta oznacza, iż na każdy ograniczony sygnał wejściowy powinniśmy otrzymać ograniczony sygnał wyjściowy.

Definicja 1.3.2. System stabilny
System L: $D \to S$ nazwiemy *stabilnym*, jeżeli

$$\forall_{x \in D} \quad \| x \|_\infty < M \Rightarrow \exists_{M_1 > 0} \quad \| y \|_\infty < M_1$$

gdzie $y = L[x]$.

Twierdzenie 1.3.2.
System liniowy jest stabilny wtedy i tylko wtedy, gdy

$$\sum_{k=-\infty}^{\infty} |h(n)| < \infty$$

Dowód:
Jeżeli założymy, że :

$$\sum_{k=-\infty}^{\infty} |h(n)| < \infty \text{ oraz } \| x \|_\infty < M$$

to

$$|y(n)| = |\sum_{k=-\infty}^{\infty} x(k)h(n-k)| \leq \sum_{k=-\infty}^{\infty} |x(k)||h(n-k)| < M \sum_{k=-\infty}^{\infty} |h(k)|$$

i możemy przyjąć, że $M_1 = M \sum_{k=-\infty}^{\infty} |h(k)|$. Teraz załóżmy, że system jest stabilny. Warunek ten jest równoważny temu, iż rozpatrywany system L jest operatorem ciągłym w normie $\| \cdot \|_\infty$ w przestrzeni ciągów ograniczonych. Wynika stąd, że operator L jest ograniczony:

$$\| L \| = \sup_{\|x\|_\infty} \| L[x] \|_\infty < \infty$$

Zdefiniujmy następujący zbiór ciągów:

$$x^i(k) = \begin{cases} \frac{\overline{h(-k)}}{|h(-k)|} & h(-k) \neq 0 \wedge |k| \leq i \\ 0 & \text{dla pozostałych k} \end{cases} \qquad i = 0, 1, \ldots \quad k = 0, \pm 1, \ldots$$

Zauważmy, że są to ciągi skończone, które mogą być rożne od zera dla $k = -i, -i+1, \ldots, i$. Zachodzi również $\| x^i \|_\infty \leq 1$. Oznaczmy

$$y^i = L[x^i]$$

Mamy wtedy:

$$y^i(0) = \sum_{k=-\infty}^{\infty} x^i(k)h(0-k) = \sum_{k=-i}^{i} |h(-k)| = \sum_{k=-i}^{i} |h(k)|$$

a stąd

$$\| y^i \|_\infty = \| L[x^i] \|_\infty \leq \| L \| \| x^i \|_\infty \leq \| L \|$$

czyli

$$\sum_{k=-i}^{i} |h(k)| \leq \| y^i \|_\infty \leq \| L \| < \infty \quad \forall_i$$

Stąd wynika, że $\sum_{k=-\infty}^{\infty} |h(k)| < \infty$, co kończy dowód.

□

Nietrudno sprawdzić, że systemy typu **FIR** są zarówno przyczynowe jak i stabilne.

Rozdział 2

Dyskretna Transformata Fouriera (DFT)

2.1 Wprowadzenie

W rozdziale tym przedstawimy Dyskretną Transformatę Fouriera (**DFT**), będąca odpowiednikiem zwykłej transformaty Fouriera dla funkcji dyskretnych. Pokażemy też w jaki sposób możemy przetwarzać sygnały ciągłe (w czasie) za pomocą DFT. Powodem dla którego będziemy zajmować się właśnie tą transformatą jest fakt istnienia algorytmu tzw. szybkich transformat Fouriera (**F**ast **F**ourier **T**ransform) służacego do wyznaczania tej transformaty. Pod koniec rozdziału omawiamy jedną z wersji tego algorytmu oraz omawiamy jak można wykorzystać go do obliczania splotu dwóch ciągów [1].

2.2 Definicja oraz własności DFT

W całym rozdziale będziemy zajmować się jedynie ciągami skończonymi i mającymi początek w zerze. Ograniczenie spowodowane jest tym, że na komputerze można przetwarzać jedynie dane o skończonym wymiarze.

Definicja 2.2.1. Charakterystyka częstotliwościowa sygnału (widmo sygnału)
Charakterystyka częstotliwościowa sygnału skończonego x dana jest wzorem

$$X(e^{i\omega}) = \sum_{n=0}^{N-1} x(n)e^{-i\omega n}$$

gdzie N jest długością sygnału.

Definicja 2.2.2. Dyskretna Transformata Fouriera (**DFT**)
Dyskretną transformatę Fouriera definiujemy w następujący sposób

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-i\frac{2\pi}{N}kn} \tag{2.1}$$

gdzie x jest ciągiem skończonym o długości N.

[1] Warto zauważyć, że system niezmienny względem przesunięcia jest opisany operacją splotu (1.9)

Definicja 2.2.3. Odwrotna Dyskretna Transformata Fouriera (**IDFT**)
Odwrotną dyskretną transformatę Fouriera definiujemy w następujący sposób

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{i\frac{2\pi}{N}nk} \tag{2.2}$$

gdzie X jest ciągiem skończonym o długości N.

Dowód poprawności definicji (2.2)
Aby pokazać poprawność definicji (2.2) podstawimy wartość $X(k)$ z równania (2.1)

$$\begin{aligned}
&\frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{-i\frac{2\pi}{N}kn} = \frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{m=0}^{N-1} x(m)e^{-i\frac{2\pi}{N}km}\right)e^{i\frac{2\pi}{N}kn} = \\
= \quad &\frac{1}{N}\sum_{m=0}^{N-1} x(m)\sum_{k=0}^{N-1} e^{i\frac{2\pi}{N}k(n-m)} = \frac{1}{N}\sum_{m=0}^{N-1} x(m)\sum_{k=0}^{N-1}\left(e^{i\frac{2\pi}{N}(n-m)}\right)^k = \\
= \quad &\frac{1}{N}Nx(n) = x(n)
\end{aligned}$$

2.2.1 Własności DFT

Elementarne własności DFT

- **Okresowość**
  DFT ciągu skończonego x jest ciągiem okresowym o okresie równym długości x

- **Liniowość**
  Niech x_1 i x_2 będą ciągami dyskretnymi skończonymi o długościach odpowiednio N_1 i N_2. Aby obydwa ciągi miały taką samą długość ustalamy $N := \max(N_1, N_2)$ i krótszy ciąg uzupełniamy zerami tak aby miał długość N. Oznaczmy przez y ciąg będący kombinacją liniową tak powstałych ciągów

  $$y(n) = a_1x_1(n) + a_2x_2(n)$$

  oraz niech $Y = DFT[y]$, wtedy

  $$\begin{aligned}
  &Y(k) = \sum_{n=0}^{N-1} y(n)e^{-i\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1}(a_1x_1(n) + a_2x_2(n))e^{-i\frac{2\pi}{N}kn} = \\
  = \quad &a_1\sum_{n=0}^{N-1} x_1(n)e^{-i\frac{2\pi}{N}kn} + a_2\sum_{n=0}^{N-1} x_2(n)e^{-i\frac{2\pi}{N}kn} = a_1X_1(k) + a_2X_2(k)
  \end{aligned}$$

  czyli Y jest kombinacją liniową obrazów ciągów x_1 i x_2 przy przekształceniu DFT.

- **Splot liniowy**
  Niech x,y będą ciągami skończonymi o długości N oraz niech $X = DFT[x]$ i $Y = DFT[y]$, zachodzi wtedy

  $$(x \star y) = IDFT[XY] \tag{2.3}$$

Dowód:

$$
\begin{aligned}
(x \star y)(t) &= \sum_{j=0}^{N-1} x(j)y(t-j) \\
&= \sum_{j=0}^{N-1} \frac{1}{N} \sum_{n=0}^{N-1} X(n) e^{i\frac{2\pi}{N}nj} \frac{1}{N} \sum_{m=0}^{N-1} Y(m) e^{i\frac{2\pi}{N}m(t-j)} = \\
&= \frac{1}{N^2} \sum_{j=0}^{N-1} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} X(n)Y(m) e^{i\frac{2\pi}{N}nj} e^{i\frac{2\pi}{N}mt} e^{-i\frac{2\pi}{N}mj} = \\
&= \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \sum_{j=0}^{N-1} X(n)Y(m) e^{i\frac{2\pi}{N}nj} e^{i\frac{2\pi}{N}mt} e^{-i\frac{2\pi}{N}mj} = \\
&= \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} X(n)Y(m) e^{i\frac{2\pi}{N}mt} \sum_{j=0}^{N-1} e^{i\frac{2\pi}{N}j(n-m)} = \\
&= \frac{1}{N} \sum_{n=0}^{N-1} X(n)Y(n) e^{i\frac{2\pi}{N}nt} = IDFT[XY]
\end{aligned}
$$

Szybki algorytm wyznaczania DFT (algorytm FFT)

Przypuśćmy, że ciąg $x \in S_F$ oraz jego długość jest potęgą dwójki ($N = 2^k$). Wprowadźmy następujące oznaczenia na ciągi utworzone z parzystych oraz z nieparzystych wyrazów ciągu x.

- $x^p(n) = x(2n)$ dla $n = 0, 1, \ldots, \frac{N}{2} - 1$
- $x^{np}(n) = x(2n+1)$ dla $n = 0, 1, \ldots, \frac{N}{2} - 1$

Zapiszmy DFT dla ciągów x^p oraz x^{np}.

- $X^p(k) = \sum_{n=0}^{\frac{N}{2}-1} x^p(n) e^{-i\frac{2\pi}{N/2}nk}$
- $X^{np}(k) = \sum_{n=0}^{\frac{N}{2}-1} x^{np}(n) e^{-i\frac{2\pi}{N/2}nk}$

Twierdzenie 2.2.1. Operacja motylkowa

Przy powyższych oznaczeniach zachodzi zależność nazywana *operacją motylkową*

$$X(k) = X^p(k) + e^{-i\frac{2\pi}{N}k} X^{np}(k) \tag{2.4}$$

$$X(k + \frac{N}{2}) = X^p(k) - e^{-i\frac{2\pi}{N}k} X^{np}(k) \tag{2.5}$$

dla $k = 0, 1, \ldots, \frac{N}{2} - 1$.

Dowód:
Dowód pierwszej równości

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i\frac{2\pi}{N}nk} = \sum_{n=0}^{\frac{N}{2}-1} x(2n) e^{-i\frac{2\pi}{N}2nk} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) e^{-i\frac{2\pi}{N}(2n+1)k} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x^p(n) e^{-i\frac{2\pi}{N/2}nk} + e^{-i\frac{2\pi}{N}k} \sum_{n=0}^{\frac{N}{2}-1} x^{np} e^{-i\frac{2\pi}{N}nk} =$$

$$= X^p(k) + e^{-i\frac{2\pi}{N}k} X^{np}(k)$$

Dowód drugiej równości jest natychmiastowy jeśli uwzględnić, to że ciągi X^p oraz X^{np} są ciągami okresowymi o okresie $\frac{N}{2}$ oraz że $e^{-i\frac{2\pi}{N}(k+\frac{N}{2})} = e^{-i\frac{2\pi}{N}} e^{-i\pi} = -e^{-i\frac{2\pi}{N}k}$, bowiem wtedy na podstawie dowiedzionej przed chwilą równości otrzymujemy

$$X(k+\frac{N}{2}) = X^p(k+\frac{N}{2}) + e^{-i\frac{2\pi}{N}(k+\frac{n}{2})} X^{np}(k+\frac{N}{2}) = X^p(k) - e^{-i\frac{2\pi}{N}k} X^{np}(k)$$

co kończy dowód.

□

Widać więc, że wyznaczenie N wartości DFT możemy sprowadzić do dwukrotnego wyznaczenia wartości DFT o długości $\frac{N}{2}$ i zastosowania operacji motylkowych. Poniższy fragment kodu powinien wszystko wyjaśnić:

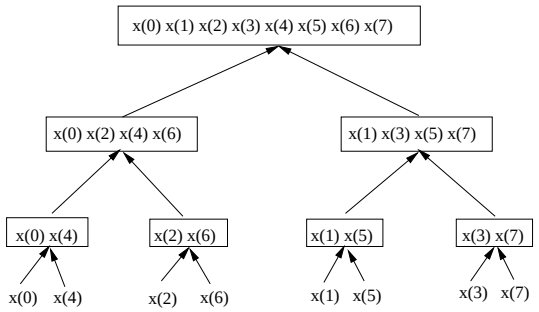
```
Wektor RekFFT(Wektor A)
{
        //wynikowy wektor reprezentujący DFT wektora A
    Wektor wynik;
    //dlugosc wektora A
    int N;
    int k;
    //pomocnicze zmienne zespolone
    Zespolona omega,omegaN;

    //pobierz długość wektora A
    N=długość(A);
    if(n==1) then
        //wartość DFT ciągu odługości 1 jest równa jemu samemu
        return A;
    //inicjalizacja zmiennych
    omegaN=exp(-i(2*pi)/N);
    omega=1;
    Tablica Xp,Xnp;
    //oblicz DFT dla ciągu złożonego z parzystych wyrazów ciągu A
    Xp=RekFFT(parzyste(A));
    //oblicz DFT dla ciągu złożonego z nieparzystych wyrazów ciągu A
    Xnp=RekFFT(nieparzyste(A));
    for(k=0;k<N/2-1;k++) {
        //operacja motylkowa
        wynik[k]=Xp(k)+omega*Xnp[k];
        wynik[k+N/2]=Xp(k)-omega*Xnp[k];
        omega=omega*omegaN;
    };
    return wynik;
};
```

Przedstawiona wersja procedury obliczającej DFT jest wersją rekurencyjną, a zatem nie jest zbyt optymalna. W rzeczywistości używa się wersji nie rekurencyjnej:

```
Wektor IterFFT(Wektor A) {
    Wektor wynik;
    int N; //długość ciągu wejściowego
    N=długość(A);
    //Sortuj z odwróconą kolejnością bitów
    int j=0;
    int i=-1;
    do {
        i=i+1;
        if(i<j) {
            //zamień A[i] z A[j]
            wynik[i]=A[j];
            wynik[j]=A[i];
        } else //po prostu przekopiuj
            wynik[i]=A[i];
        int l=N/2;
        while(l≤j) {
            j=j-l;
            l=l/2;
        };
        j=j+l;
    } while(i<N-2);
    int m=2; //długość DFT na danym poziomie poziom rekursji
    while(m<N) {
        zespolona omega,omega_m;
        omega_m=exp(-i*2*pi/m);
        omega=1;
        for(int j=0;j<m/2-1;j++) {
            for(int k=j;k<N-1;k+=m) {
                zespolona t,u;
                t=wynik[k+m/2]*omega;
                u=wynik[k];
                //operacja motylkowa
                wynik[k]=u+t;
                wynik[k+m/2]=u-t;
            };
            omega=omega*omega_m;
        };
        m=m*2; //przejdz poziom wyżej
    };
    return wynik;
};
```

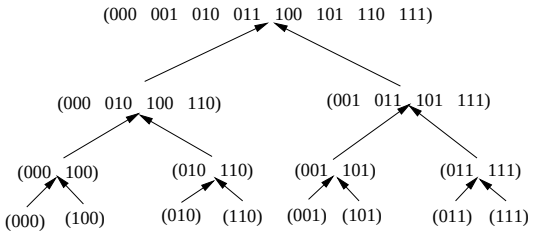
Powyższy kod może wydawać się dość zawiły, lecz w rzeczywistości po uważnym przeczytaniu staje się jasny. Komentarza wymaga część dotycząca tzw. *sortowania z odwróconą kolejnością bitów*. Widać, że jeśli dane wejściowe występują w takiej kolejności jak w liściach drzewa (2.1), to możemy naśladować działanie procedury *RekFFT* w następujący sposób:

Na początku rozważamy pary kolejnych elementów ($m = 2$), zastępując każdą z nich poprzez DFT tej pary. Otrzymujemy $\frac{N}{2}$ takich dwuelementowych wyników.



Rysunek 2.1: Drzewo wywołań rekurencyjnych funkcji RekFFT (węzły oznaczają parametr A)

Następnie bierzemy po dwie takie pary ($m = 4$), obliczamy DFT dla każdej otrzymanej w ten sposób czwórki elementów tablicy za pomocą dwóch operacji motylkowych i zastępujemy owe dwuelementowe wyniki jednym czteroelementowym. Tablica zawiera teraz $\frac{N}{4}$ 4-ro elementowych wyników DFT. Postępujemy, tak aż w tablicy będą dwa wyniki DFT $\frac{N}{2}$ elementowe, które łączymy za pomocą $\frac{N}{2}$ operacji motylkowych w jedną wynikową tablicę DFT o N elementach. Pozostaje problem jak ustawić elementy tablicy wejściowej w porządku takim, jak liście w drzewie z rysunku (2.1). Jak wiadomo każdą liczbę naturalną można przedstawić w postaci ciągu zer i jedynek, zapisując ją w systemie dwójkowym. Przyjrzyjmy się rysunkowi:



Rysunek 2.2: Drzewo wywołań rekurencyjnych funkcji RekFFT (węzły oznaczają indeksy wejściowego ciągu A w kolejnych wywołaniach)

Widać, że np. element numer 3 (011) trafia na liść o numerze 6 (110), a element o numerze 6 (110) na liść o numerze 3 (110). Podobnie rzecz się ma z innymi elementami. Wynika z tego, że powinno się zamienić pary elementów o indeksach będących swymi łustrzanymi odbiciami".

Na koniec omawiania algorytmu FFT krótka uwaga na temat obliczania odwrotnej transformaty Fouriera. Otóż wystarczy zmienić znak w wywołaniu funkcji $\exp$ oraz otrzymany ciąg podzielić przez N. Można też skorzystać z wzoru

$$IDFT[x] = \frac{1}{N}\overline{DFT}[\overline{X}]$$

Powstaje pytanie dlaczego stosować algorytm FFT zamiast bezpośrednio wyliczać DFT. Otóż powód jest bardzo prosty. Koszt wykonania algorytmu bezpośredniego jest rzędu $O(N^2)$ mnożeń i $O(N(N-1))$ dodawań, zaś algorytmu FFT jest rzędu $O(\frac{N}{2}\log N)$ mnożeń i $O(N \log N)$ dodawań. Przyśpieszenie jest już widoczne dla $N = 64$.

Splot dwóch ciągów

- Splot dwóch ciągów o zadanych długościach

  Przypuśćmy, że mamy dwa ciągi x_1 i x_2 o długościach odpowiednio N_1 i N_2, tzn.

  $$x_1(n) = 0 \quad \text{dla } n < 0 \text{ lub } n > N_1 - 1$$
  $$x_2(n) = 0 \quad \text{dla } n < 0 \text{ lub } n > N_2 - 1$$

  Określimy ciąg x_3

  $$x_3(n) = \sum_{k=-\infty}^{\infty} x_1(k)x_2(n-k) = \sum_{k=0}^{N_1-1} x_1(k)x_2(n-k)$$

  Zauważmy, że

  $$x_3(n) = 0 \quad \text{dla } n < 0 \text{ lub } n > N_1 + N_2 - 2$$

  Przyjmijmy, że $N = N_1 + N_2 - 1$ i zdefiniujmy ciągi pomocnicze

  $$y_1(n) = \begin{cases} x_1(n) & 0 \le n \le N_1 - 1 \\ 0 & N_1 \le n \le N - 1 \end{cases}$$
  $$y_2(n) = \begin{cases} x_2(n) & 0 \le n \le N_2 - 1 \\ 0 & N_2 \le n \le N - 1 \end{cases}$$

  dla pozostałych n zachodzi $y_1(n) = y_2(n) = 0$. Zdefiniujmy następnie przedłużenia okresowe tych ciągów

  $$\tilde{y}_1(n) = \sum_{r=-\infty}^{\infty} y_1(n + rN)$$
  $$\tilde{y}_2(n) = \sum_{r=-\infty}^{\infty} y_2(n + rN)$$

  Zauważmy, że zachodzi zależność:

  $$x_3(n) = \sum_{k=0}^{N-1} \tilde{y}_1(k)\tilde{y}_2(n-k) \quad \text{dla } n = 0, 1, \ldots, N-1$$

  Korzystając z zależności (2.3) możemy zapisać, że

  $$X_3(k) = \tilde{Y}_1(k)\tilde{Y}_2(K) \quad \text{dla } k = 0, 1, \ldots, N-1$$

  Ponieważ $\tilde{Y}(k) = Y(k)$, możemy napisać

  $$x_3 = IDFT[X_3] = IDFT[Y_1Y_2]$$

- Splot dwóch ciągów, z których jeden ma teoretycznie nieograniczoną długość

  Niech ciągiem „krótkim” będzie ciąg h, a ciągiem teoretycznie nieskończonym ciąg x. Nie możemy wykonać splotu tych ciągów w sposób opisany przed

chwilą. Dokonamy więc podziału ciągu x na segmenty o długości $N > M$, gdzie M jest długością ciągu h.

$$x_k(n) = \begin{cases} x(n) & kN \leq n \leq (k+1)N - 1 \\ 0 & \text{w pozostałych przypadkach} \end{cases}$$

$k = 0, 1, \ldots$
Wyznaczając kolejne sploty ciągu h z x_k dla $k = 0, 1, \ldots$ otrzymujemy

$$y_k = (h \star x_k)$$

gdzie $y_k(n) = 0$ dla $n < kN$ lub $n > (k+1)N + M - 2$. Sumując tak otrzymane ciągi y_k otrzymujemy ciąg wyjściowy y:

$$y = h \star x = h \star \sum_{k=0}^{\infty} x_k = \sum_{k=0}^{\infty} h \star x_k = \sum_{k=0}^{\infty} y_k$$

Chwili uwagi wymaga zauważenie, że wystarczy jedynie dodawać, po obliczeniu kolejnego ciągu y_k, zakładki o długości $M - 1$:

$$y_k(n) := y_k(n) + y_{k-1}(n) \quad \text{dla } n = (k-1)N, \ldots, (k-1)N + M - 2$$

Powyższa procedura zwana jest *sumowanie z nakładaniem* lub *overlap-add method* i może być wykorzystana do realizacji systemu typu **FIR**

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$$

W następnym rozdziale opiszemy jak tworzyć takie systemy za pomocą algorytmu Remeza.

2.3 Związek między sygnałem analogowym i dyskretnym

Dotychczas zajmowaliśmy się jedynie sygnałami dyskretnymi nic nie wspominając o tym skąd one się biorą. Oczywiście część sygnałów dyskretnych może być takimi ze swojej natury, ale często powstają one w wyniku dyskretyzacji jakiegoś sygnału ciągłego (analogowego). Pojawia się wtedy problem zachowania podstawowych własności sygnału analogowego.

Dyskretyzacja sygnału analogowego jest dokonywana metodą *próbkowania*, tzn. przez ustalenie wartości sygnału analogowego w dyskretnych punktach czasowych. Powszechnie stosuje się próbkowanie o z góry ustalonym okresie ustalania wartości sygnału.

Będziemy przyjmować, że sygnał analogowy i jego próbki (sygnał dyskretny) są sobie równoważne, jeżeli próbkowanie zachowa charakter widma sygnału oraz jeżeli teoretycznie będzie istniała możliwość całkowitego odtworzenia pierwotnego sygnału analogowego na podstawie jego próbek.

Będziemy również zakładać, że sygnał analogowy s ma *ograniczone pasmo*, tzn.

$$X_a(\Omega) = 0 \quad \text{dla } |\Omega| > \Omega_0$$

gdzie X_a oznacza widmo sygnału analogowego s

$$X_a(\Omega) = \int_{-\infty}^{\infty} s(t)e^{-i\Omega t}\mathrm{d}t$$

$$s(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X_a(\Omega)e^{i\Omega t}\mathrm{d}\Omega = \frac{1}{2\pi}\int_{-\Omega}^{\Omega} X_a(\Omega)e^{i\Omega t}\mathrm{d}\Omega$$

Równomierne próbkowanie polega na wybieraniu wartości sygnału analogowego w równoodległych chwilach T

$$x(n) = s(nT)$$

T zwane jest *okresem próbkowania*, natomiast jego odwrotność $f = \frac{1}{T}$ *częstotliwością próbkowania*. Zachodzi następujące twierdzenie

Twierdzenie 2.3.1. twierdzenie Shannona

Niech s będzie sygnałem opisywanym powyżej, oznaczmy $f_0 = \frac{1}{\Omega_0}$. Wtedy s może zostać odtworzony z próbek $x(n) = s(nT)$, gdzie $T \leq \frac{1}{2f_0}$ i dany jest wzorem

$$s(t) = T\sum_{n=-\infty}^{\infty} s(nT)\frac{\sin 2\pi f_0(t-nT)}{2\pi f_0(t-nT)}$$

□

Z twierdzenia tego wynika, że jeśli chcemy zachować charakter widma sygnału analogowego, częstotliwość próbkowania powinna być co najmniej dwukrotnie większa niż ograniczenie pasma widma tego sygnału. Częstotliwość $2f_0$ nazywana jest *częstotliwością Nyquista*.

Rozdział 3

Metoda Remeza w projektowaniu filtrów FIR o liniowej fazie

3.1 Liniowa aproksymacja jednostajna (ogólny algorytm Remeza)

Niech $C(\mathrm{B})$ oznacza przestrzeń funkcji ciągłych na zbiorze $\mathrm{B} \subset \mathbb{R}$, $f\colon \mathrm{B} \to \mathbb{R}$, natomiast V_n n-wymiarową podprzetrzeń $C(\mathrm{B})$.

Definicja 3.1.1. Norma jednostajna

Niech $f \in C[\mathrm{B}]$, gdzie B jest zwartym odcinkiem, wtedy

$$\| f \| = \max_{x \in F} |f(x)| \tag{3.1}$$

Sformułowanie zadania

Zadanie liniowej aproksymacji jest określone następująco:
dla $f \in C(\mathrm{F})$ znaleźć $h_0 \in V_n$ takie, aby

$$\| f - h_0 \| = \inf_{h \in V_n} \| f - h \|$$

Funkcję h_0 nazywamy *optymalną* dla f.

Definicja 3.1.2. warunek Haara

Mówimy, że podprzestrzeń liniowa $U \subset C_{\mathrm{F}}$ skończonego wymiaru n spełnia *warunek Haara*, jeżeli każda funkcja z U, nie równa tożsamościowo zeru, znika co najwyżej w $n-1$ punktach zbioru F.

Twierdzenie 3.1.1. o alternansie

Jeżeli V_n spełnia warunek Haara, $f \in C(\mathrm{B})$, B jest zbiorem zwartym o mocy większej niż n, to $h \in V_n$ jest funkcją optymalną dla F wtedy i tylko wtedy, kiedy istnieją punkty $x_1 < x_2 < \ldots < x_{n+1}$, $x_k \in \mathrm{B}$, spełniające:

$$e(x_k) = (-1)^k \zeta \| e \| \quad k = 1, 2, \ldots, n+1$$

gdzie

$$\zeta = -1, 0, 1 \quad \text{i} \quad \zeta = 0 \Leftrightarrow f \in V_n$$

Algorytm REMEZA

Algorytm ten polega na konstruowaniu dyskretnych zbiorów $\mathrm{M}_i = \{x_1^i, \ldots, x_{n+1}^i\}$ takich, że $\mathrm{M}_i \to \mathrm{M}$, gdzie M jest alternansem funkcji f. Przyjmijmy dla wygody oznaczeń, że $\mathrm{B} = [a, b]$.

Na początku ustala się zbiór startowy M_0.

Załóżmy teraz, że znamy zbiór M_i i chcemy skonstruować zbiór M_{i+1}.

Niech h_i oznacza funkcję optymalną dla f, ale przy ograniczeniu dziedziny zbioru B do zbioru M_i. Z twierdzenia o alternansie wynika, że cały zbiór M_i (ma on jedynie $n+1$ punktów) jest alternansem dla $f|_{\mathrm{M}_i}$. Można więc napisać

$$f(x_k^i) - h_i(x_k^i) = (-1)^k \lambda_i, \quad k = 1, 2, \ldots, n+1 \tag{3.2}$$

gdzie

$$\lambda_i = \zeta \parallel e_i \parallel$$

Załóżmy, że ciąg funkcji $g_m(m+0, \ldots, n-1)$ stanowi bazę przestrzeni V_n. Istnieją więc stałe $a^i(m)$ spełniające

$$h_i(x) = \sum_{m=0}^{n-1} a^i(m) g_m(x) \tag{3.3}$$

Bezpośrednio z (3.2) i (3.3) wynika

$$f(x_k^i) - \sum_{m=0}^{n-1} a^i(m) g_m(x_k^i) = (-1)^k \lambda_i, \quad k = 1, 2, \ldots, n+1 \tag{3.4}$$

a stąd

$$\sum_{m=0}^{n-1} a^i(m) g_m(x_k^i) + (-1)^k \lambda_i = f(x_k^i), \quad k = 1, 2, \ldots, n+1 \tag{3.5}$$

Otrzymaliśmy w ten sposób układ $n+1$ równań liniowych z $n+1$ niewiadomymi $(a^i(m), m = 0, \ldots, n-1, \lambda_i)$. Rozwiązując ten układ otrzymamy funkcję h_i, a dokładniej jej rozwinięcie w bazie V_n oraz wartość λ_i. Dzięki temu można wyznaczyć wartość funkcji błędu e_i w dowolnym punkcie $x \in \mathrm{B}$.

$$e_i(x) = f(x) - h_i(x) \tag{3.6}$$

Jeżeli funkcja h_i rozszerzona na całą dziedzinę B nie jest optymalną dla f, to istnieje punkt $x \in \mathrm{B}$ taki, że

$$|e_i(x)| > |e_i(x_k^i)| \tag{3.7}$$

Oznaczmy przez z_k^i zera funkcji e_i. Jest ich co najmniej n, gdyż wiemy, że e_i zmienia znak co najmniej n razy. Wybierzmy zera tak, aby $x_k^i < z_k^i < x_{k+1}^i, \quad k = 1, \ldots, n$ i dodatkowo oznaczmy $z_0^i = a$ oraz $z_{n+1}^i = b$. W ten sposób dokonaliśmy podziału B na odcinki $[z_k^i, z_{k+1}^i]$. W każdym z tak otrzymanych odcinków znajdujemy wartości ekstremalne funkcji e_i. Oznaczmy je przez $x_{ex_k}^i$. Zbiór M_{i+1} powstaje poprzez odpowiednią zamianę punktów zbioru M_i na punkty $x_{ex_k}^i$. Zamiana ta musi być dokonana w ten sposób, aby zachować naprzemienny znak wartości funkcji e_i w punktach zbioru M_{i+1}.

Wykazuje się, że tak tworzone kolejne zbiory M_i zbiegają do alternansu funkcji f. I iteracja w algorytmie Remeza jest kończona, gdy:

- nie ma punktów do wymiany, co oznacza, że została osiągnięta funkcja optymalna;
- została przekroczona ustalona duża liczba iteracji, co świadczy o bardzo wolnej zbieżności;
- w i-tym kroku zachodzi $|\lambda_i| > |\lambda_{i+1}|$, co świadczy o braku zbieżności spowodowanym błędami arytmetyki.

Uwagi

Z powodu kosztowności dwóch kroków algorytmu wykonuje się je trochę inaczej. Chodzi o rozwiązywanie układu równań (3.5) oraz wyznaczania wartości funkcji e_i. Stosuje się przybliżenie interpolacyjne oparte na węzłach x_k^i, $k = 1, \ldots, n$. Koszt związany z wyznaczaniem punktów ekstremalnych zmniejsza się przez wprowadzenie zamiast zbioru B siatki $\mathrm{B}_i \subset \mathrm{B}$ o liczbie punktów wyraźnie większej niż n. W takim przypadku jest poszukiwana funkcja optymalna nie dla $f \in C(\mathrm{B})$, lecz dla $f \in C(\mathrm{B}_i)$. Wprowadzony w ten sposób dodatkowy błąd zależy od rozdzielczości siatki. Jeżeli jest ona równomierna z rozdzielczości ϵ, to błąd dodatkowy jest rzędu ϵ^2.

3.2 Zastosowanie algorytmu Remeza w projektowaniu filtrów typu FIR

W tym podroździale chcielibyśmy pokazać jak można zastosować ogólny algorytm Remeza w przypadku tworzenia filtrów typu FIR o fazie liniowej.

3.2.1 Konsekwencje liniowości fazy filtru

Rozważymy teraz cztery przypadki kiedy filtr typu FIR ma liniową fazę. W każdym z tych przypadków charakterystykę częstotliwościową filtru można przedstawić w postaci

$$H(e^{i\omega}) = R(\omega)e^{-i\left[\frac{N-1}{2}\omega - r\right]}$$

gdzie N jest rzędem filtru tzn. długością ciągu będącego odpowiedzią na impuls jednostkowy.

- **Przypadek 1**
  N-nieparzyste
  $h(n) = h(N-n-1)$ dla $n = 0, \ldots, N-1$
  Uwzględniając te warunki otrzymujemy

$$\begin{aligned} H(e^{i\omega}) &= \sum_{n=0}^{\frac{N-1}{2}-1} h(n)\left(e^{-i\omega n} + e^{-i\omega(N-n-1)}\right) + h\left(\frac{N-1}{2}\right)e^{-i\omega\frac{N-1}{2}} = \\ &= e^{-i\omega\frac{N-1}{2}}\left(h\left(\frac{N-1}{2}\right) + \sum_{n=0}^{\frac{N-1}{2}-1} h(n)\left(e^{-i\omega(n-\frac{N-1}{2})} + e^{i\omega(n-\frac{N-1}{2})}\right)\right) = \\ &= e^{-i\omega\frac{N-1}{2}}\left(h\left(\frac{N-1}{2}\right) + \sum_{n=0}^{\frac{N-1}{2}-1} 2h(n)\cos\left[\omega\left(n-\frac{N-1}{2}\right)\right]\right) = \end{aligned}$$

$$= \quad e^{-i\omega\frac{N-1}{2}}\left(h\left(\frac{N-1}{2}\right)+\sum_{n=1}^{\frac{N-1}{2}}2h\left(\frac{N-1}{2}-n\right)\cos(\omega n)\right)$$

zatem

$$R(\omega)=h\left(\frac{N-1}{2}\right)+\sum_{n=1}^{\frac{N-1}{2}}2h\left(\frac{N-1}{2}-n\right)\cos(\omega n) \tag{3.8}$$

$$r=0 \tag{3.9}$$

- **Przypadek 2**
  N-nieparzyste
  $h(n)=-h(N-n-1)$ dla $n=0,\ldots,N-1$
  $h(\frac{N-1}{2})=0$
  Uwzględniając te warunki otrzymujemy

$$\begin{aligned}
H(e^{i\omega}) &= \sum_{n=0}^{\frac{N-1}{2}-1}h(n)\left(e^{-i\omega n}-e^{-i\omega(N-n-1)}\right)= \\
&= e^{-i\omega\frac{N-1}{2}}\left(\sum_{n=0}^{\frac{N-1}{2}-1}h(n)\left(e^{-i\omega(n-\frac{N-1}{2})}-e^{i\omega(n-\frac{N-1}{2})}\right)\right)= \\
&= e^{-i\omega\frac{N-1}{2}}\left(\sum_{n=0}^{\frac{N-1}{2}-1}2h(n)i\sin\left[\omega\left(\frac{N-1}{2}-n\right)\right]\right)= \\
&= e^{-i\omega\left(\frac{N-1}{2}-\frac{\pi}{2}\right)}\left(\sum_{n=0}^{\frac{N-1}{2}-1}2h(n)\sin\left[\omega\left(\frac{N-1}{2}-n\right)\right]\right)= \\
&= e^{-i\omega\left(\frac{N-1}{2}-\frac{\pi}{2}\right)}\left(\sum_{n=1}^{\frac{N-1}{2}}2h\left(\frac{N-1}{2}-n\right)\sin(\omega n)\right)
\end{aligned}$$

zatem

$$R(\omega)=\sum_{n=1}^{\frac{N-1}{2}}2h\left(\frac{N-1}{2}-n\right)\sin(\omega n) \tag{3.10}$$

$$r=\frac{\pi}{2} \tag{3.11}$$

- **Przypadek 3**
  N-parzyste
  $h(n)=h(N-n-1)$ dla $n=0,\ldots,N-1$
  Uwzględniając te warunki otrzymujemy

$$\begin{aligned}
H(e^{i\omega}) &= \sum_{n=0}^{\frac{N}{2}-1}h(n)\left(e^{-i\omega n}+e^{-i\omega(N-n-1)}\right)= \\
&= e^{-i\omega\frac{N-1}{2}}\left(\sum_{n=0}^{\frac{N}{2}-1}h(n)\left(e^{-i\omega(n-\frac{N-1}{2})}+e^{i\omega(n-\frac{N-1}{2})}\right)\right)=
\end{aligned}$$

$$= e^{-i\omega\frac{N-1}{2}}\left(\sum_{n=0}^{\frac{N}{2}-1} 2h(n)\cos\left[\omega\left(n-\frac{N-1}{2}\right)\right]\right) =$$

$$= e^{-i\omega\frac{N-1}{2}}\left(\sum_{n=1}^{\frac{N}{2}} 2h\left(\frac{N}{2}-n\right)\cos\left[\omega\left(n-\frac{1}{2}\right)\right]\right)$$

zatem

$$R(\omega) = \sum_{n=1}^{\frac{N-1}{2}} 2h\left(\frac{N}{2}-n\right)\cos\left[\omega\left(n-\frac{1}{2}\right)\right] \tag{3.12}$$

$$r = 0 \tag{3.13}$$

- **Przypadek 4**
  N-parzyste
  $h(n) = -h(N-n-1)$ dla $n = 0,\ldots,N-1$
  Uwzględniając te warunki otrzymujemy

$$H(e^{i\omega}) = \sum_{n=0}^{\frac{N}{2}-1} h(n)\left(e^{-i\omega n} - e^{-i\omega(N-n-1)}\right) =$$

$$= e^{-i\omega\frac{N-1}{2}}\left(\sum_{n=0}^{\frac{N}{2}-1} h(n)\left(e^{-i\omega(n-\frac{N-1}{2})} - e^{i\omega(n-\frac{N-1}{2})}\right)\right) =$$

$$= e^{-i\omega\frac{N-1}{2}}\left(\sum_{n=0}^{\frac{N}{2}-1} 2h(n)i\sin\left[\omega\left(\frac{N-1}{2}-n\right)\right]\right) =$$

$$= e^{-i\omega\left(\frac{N-1}{2}-\frac{\pi}{2}\right)}\left(\sum_{n=0}^{\frac{N}{2}-1} 2h(n)\sin\left[\omega\left(\frac{N-1}{2}-n\right)\right]\right) =$$

$$= e^{-i\omega\left(\frac{N-1}{2}-\frac{\pi}{2}\right)}\left(\sum_{n=1}^{\frac{N}{2}} 2h\left(\frac{N}{2}-n\right)\sin\left[\omega\left(n-\frac{1}{2}\right)\right]\right)$$

zatem

$$R(\omega) = \sum_{n=1}^{\frac{N}{2}} 2h\left(\frac{N}{2}-n\right)\sin\left[\omega\left(n-\frac{1}{2}\right)\right] \tag{3.14}$$

$$r = \frac{\pi}{2} \tag{3.15}$$

Mając już rozpatrzone wszystkie przypadki pokażemy, że w każdym z nich funkcję $R(\omega)$ można zapisać w postaci

$$R(\omega) = Q(\omega)\sum_{n=0}^{M} a(n)\cos(\omega n) \tag{3.16}$$

Ograniczymy się jedynie do podania gotowych wzorów, gdyż wyprowadzenie ich jest dość żmudne i opiera się na pewnych przekształceniach trygonometrycznych.

- **Przypadek 1**
  Z zależności (3.8) wynika, że

$$R(\omega) = \sum_{n=0}^{\frac{N-1}{2}} a(n)\cos(\omega n)$$

  gdzie

$$\begin{aligned} a(0) &= h\left(\frac{N-1}{2}\right) \\ a(n) &= 2h\left(\frac{N-1}{2} - n\right) \quad \text{dla} \quad n = 1, 2, \ldots, \frac{N-1}{2} \end{aligned}$$

  Czyli $M = \frac{N-1}{2}$ i $Q(\omega) = 1$.

- **Przypadek 2**
  Z równości (3.10) wynika, że

$$R(\omega) = \sum_{n=1}^{\frac{N-1}{2}} \hat{a}(n)\sin(\omega n)$$

  gdzie

$$\hat{a}(n) = 2h\left(\frac{N-1}{2} - n\right) \quad \text{dla } n = 1, 2, \ldots, \frac{N-1}{2}$$

  Przekształcając ten wzór otrzymamy

$$R(\omega) = \sin\omega \sum_{n=0}^{\frac{N-1}{2}-1} a(n)\cos\omega n$$

  gdzie

$$\begin{cases} \hat{a}(1) = a(0) - \frac{1}{2}a(2) & \\ \hat{a}(n) = \frac{1}{2}[a(n-1) - a(n+1)] & \text{dla } n = 2, \ldots, \frac{N-1}{2} - 2 \\ \hat{a}(n) = \frac{1}{2}a(n-1) & \text{dla } n = \frac{N-1}{2} - 1, \frac{N-1}{2} \end{cases} \tag{3.17}$$

  czyli $M = \frac{N-1}{2} - 1$ i $Q(\omega) = \sin\omega$.

- **Przypadek 3**
  Ze wzoru (3.12) wynika

$$R(\omega) = \sum_{n=1}^{\frac{N}{2}} \hat{a}(n)\cos\left[\omega\left(n - \frac{1}{2}\right)\right]$$

  gdzie

$$\hat{a}(n) = 2h\left(\frac{N}{2} - n\right) \quad \text{dla } n = 1, \ldots, \frac{N}{2}$$

Można wykazać, że

$$R(\omega) = \cos\frac{\omega}{2}\sum_{n=0}^{\frac{N}{2}-1} a(n)\cos\omega n$$

gdzie

$$\begin{cases} \hat{a}(1) = a(0) + \frac{1}{2}a(1) \\ \hat{a}(n) = \frac{1}{2}[a(n-1) + a(n)] \quad \text{dla } n = 2,\ldots,\frac{N}{2}-1 \\ \hat{a}(\frac{N}{2}) = \frac{1}{2}a(\frac{N}{2}-1) \end{cases} \tag{3.18}$$

czyli $M = \frac{N}{2} - 1$ i $Q(\omega) = \cos\frac{\omega}{2}$.

- **Przypadek 4**
  Wzór (3.14) umożliwia napisanie

$$R(\omega) = \sum_{n=1}^{\frac{N}{2}} \hat{a}(n)\sin\left[\omega\left(n - \frac{1}{2}\right)\right]$$

gdzie

$$\hat{a}(n) = 2h\left(\frac{N}{2} - n\right) \quad \text{dla } n = 1,\ldots,\frac{N}{2}$$

Stosując odpowiednie wzory trygonometryczne otrzymamy

$$R(\omega) = \sin\frac{\omega}{2}\sum_{n=0}^{\frac{N}{2}-1} a(n)\cos\omega n$$

gdzie

$$\begin{cases} \hat{a}(1) = a(0) - \frac{1}{2}a(1) \\ \hat{a}(n) = \frac{1}{2}[a(n-1) - a(n)] \quad \text{dla } n = 2,\ldots,\frac{N}{2}-1 \\ \hat{a}(\frac{N}{2}) = \frac{1}{2}a(\frac{N}{2}-1) \end{cases} \tag{3.19}$$

czyli $M = \frac{N}{2} - 1$ i $Q(\omega) = \sin\frac{\omega}{2}$.

3.2.2 Opis algorytmu

Mając wyprowadzoną postać (3.16) funkcji R możemy projektowanie filtru z fazą liniową sprowadzić do zadania aproksymacji jednostajnej nad przestrzenią funkcji V_{M+1} rozpiętej na bazie $\cos\omega n,\ n = 0,\ldots,M$. Przyjmijmy następujące oznaczenia

$P(\omega) = \sum_{n=0}^{M} a(n)\cos\omega n, P \in V_{M+1}$ - wielomian aproksymacyjny

$H_d(\omega)$ - zadana (docelowa) charakterystyka filtru

$W(\omega)$ - funkcja wagowa regulująca wymagania dotyczące błędu w różnych pasmach częstotliwości

Funkcję błędu aproksymacji możemy przedstawić w postaci

$$e(\omega) = W(\omega)[H_d(\omega) - Q(\omega)P(\omega)] = W(\omega)Q(\omega)\left[\frac{H_d(\omega)}{Q(\omega)} - P(\omega)\right] \qquad (3.20)$$

Będziemy poszukiwać takich współczynników $\{a\}$, które minimalizują błąd określony przez (3.20). Wynika z tego, że poszukujemy funkcji $P \in V_{M+1}$, dla której jest osiągane

$$\inf_{P \in V_{M+1}} \| e \| \quad \omega \in B,\ B \subset [0, \pi)$$

Zbiór B dobieramy tak, aby uniknąć wszelkich nieciągłości wynikających z podzielenia przez funkcję $Q(\omega)$ i z ustalenia charakterystyki H_d filtru idealnego.

Ponieważ dla $B \subset [0, \pi) V_{M+1}$ jest przestrzenią posiadającą własność Haara więc algorytm Remeza, na mocy (3.1.1), będzie zbieżny. Opiszemy teraz modyfikację ogólnego algorytmu Remeza mającą zastosowanie w projektowaniu filtrów FIR o liniowej fazie. Wprowadźmy dodatkowe oznaczenia

$$\widehat{H_d}(\omega) = \frac{H_d(\omega)}{Q(\omega)}$$

$$\widehat{W}(\omega) = W(\omega)Q(\omega)$$

Teraz funkcję błędu możemy zapisać w postaci

$$e(\omega) = \widehat{W}(\omega)[\widehat{H_d}(\omega) - P(\omega)]$$

natomiast układ (3.5) przyjmuje postać

$$\widehat{H_d}(\omega_k^i) - P(\omega_k^i) = \frac{(-1)^k \lambda}{\widehat{W}(\omega_k^i)}, \quad k = 1, 2, \ldots, M+2$$

czyli

$$\sum_{n=0}^{M} a^i(n) \cos(\omega_k^i n) + (-1)^k \frac{\lambda_i}{\widehat{W}(\omega_k^i)} = \widehat{H_d}(\omega_k^i)$$

dla $k = 1, \ldots, M+2$, gdzie i określa krok iteracyjny. Bezpośrednio z tego wzoru możemy wyznaczyć

$$\lambda_i = \frac{\sum_{k=1}^{M+1} b_k^i \widehat{H_d}(\omega_k^i)}{\sum_{k=1}^{M+1} (-1)^{k+1} \frac{b_k^i}{\widehat{W}(\omega_k^i)}}$$

gdzie

$$b_k^i = \prod_{r=1, r \neq k}^{M+1} \frac{1}{\cos(\omega_k^i) - \cos(\omega_r^i)}$$

Zamiast wyznaczać w każdym kroku iteracyjnym wartości $a^i(n)$ zastosujemy wzór interpolacyjny umożliwiający wyznaczanie wartości P w dowolnym punkcie $\omega \in B$

$$P_i(\omega) = \frac{\sum_{k=1}^{M} \frac{a_k^i}{\cos\omega - \cos(\omega_k^i)} \left[\widehat{H_d}(\omega_k^i) - (-1)^{k+1} \frac{\lambda_i}{\widehat{W}(\omega_k^i)} \right]}{\sum_{k=1}^{M} \frac{\alpha_k^i}{\cos\omega - \cos(\omega_k^i)}} \tag{3.21}$$

gdzie

$$\alpha_k^i = \prod_{r=1, r\neq k}^{M} \frac{1}{\cos(\omega_k^i) - cos(\omega_r^i)}$$

Aby w ostatnim kroku znać rozwinięcie funkcji P w bazie V_{M+1} skorzystamy ze wzoru interpolacyjnego na P. Jeżeli oznaczymy, że

$$P(\omega) = \sum_{n=0}^{M} a(n) cos\omega n$$

to

$$\begin{aligned} a(0) &= \frac{1}{2M+1} \left[P(\omega_0) + 2 \sum_{k=1}^{M} P(\omega_k) \right] \\ a(n) &= \frac{2}{2M+1} \left[P(\omega_0) + 2 \sum_{k=1}^{M} P(\omega_k) \cos(\omega_k n) \right] \quad \text{dla } n = 1, \ldots, M \end{aligned}$$

gdzie $\omega_k = \frac{2}{2M+1} k, k = 0, \ldots, M$.

Mając już wyznaczone wartości współczynników $a(n)$ w przypadkach 2, 3 i 4 wyznaczamy $\hat{a}(n)$ ze wzorów (3.17), (3.18), (3.19) a następnie wyznaczamy ciąg h korzystając z zależności między $\hat{a}(n)$ a $h(n)$.

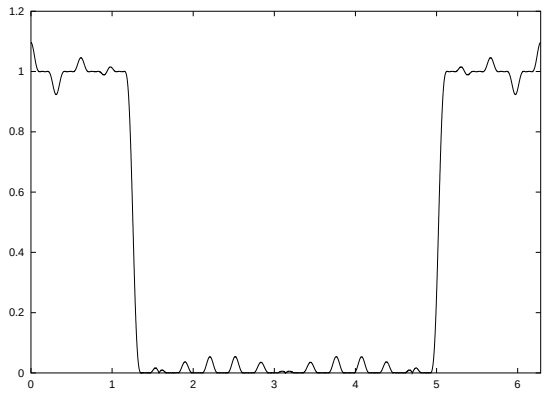
Rozdział 4

Program EQUALIZER

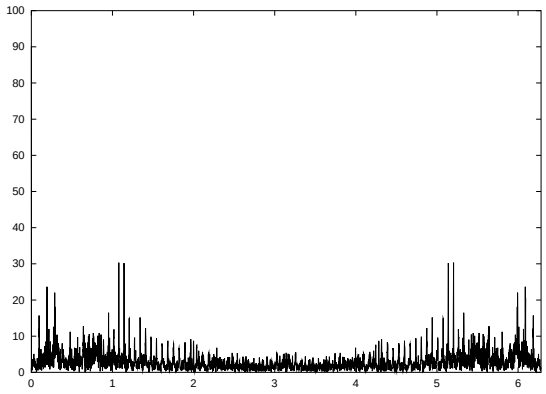
4.1 Opis programu

Program służy do zmiany charakterystyki częstotliwościowej sygnałów nagranych w formacie WAVE. Sterowanie program odbywa się poprzez linię komend (patrz następna sekcja).

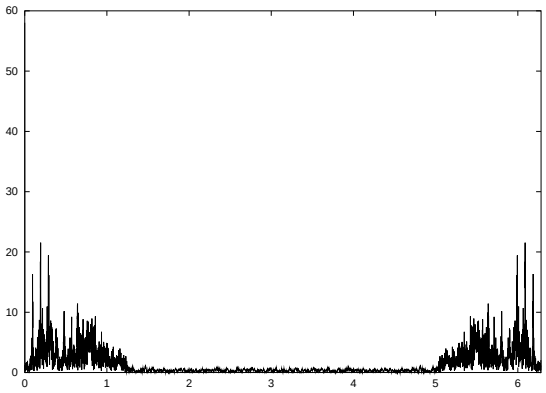
Ogólnie program dostaje na wejściu n pasm z zadanym wartościami żądanej charakterystyki częstotliwościowej filtru. Stosując algorytm Remeza tworzony ciąg h zwany *jądrem filtru*, mający charakterystykę zbliżoną w sensie normy jednostajnej, do charakterystyki żądanej oraz będący odpowiedzią na impuls jednostkowy szukanego filtru. Mając już ten ciąg program dokonuje filtracji sygnału wejściowego stosując omawianą metodę *overlap-add*. W wyniku działania programu otrzymujemy sygnał o zmienionej charakterystyce częstotliwościowej. Poniższe wykresy pokazują przykładowe wynik działania programu.



Rysunek 4.1: Filtr przepuszczający niskie częstotliwości (wynik działania programu)



Rysunek 4.2: Widmo sygnału oryginalnego



Rysunek 4.3: Widmo sygnału przetworzonego przez filtr (4.1)

Widać, że w wyniku działania filtru wyższe częstotliwości zostały praktycznie usunięte z widma sygnału.

4.2 Instrukcja użytkowania

Program EQUALIZER realizuje system typu **FIR**. Program napisany jest pod system Linuks, ale można go, po niewielkich przeróbkach, uruchomić pod np. systemem Windows.

- Program instaluje się poleceniem
  *g++ -o equalizer -mpentium -O3 *.cpp+*
- Uruchomienie programu następuje w wyniku instrukcji
  ./equalizer equalizer.fir <plik wejściowy> <plik wynikowy>
  gdzie plik wejściowy - plik typu WAVE (.wav)
- Plik equalizer.fir
  Plik ten definiuje filtr. Pierwsze dwie linie to odpowiednio: długość filtru, ilość

pasm. Kolejne [ilość pasm] linii definiuje początek i koniec każdego pasma (pasma nie mogą na siebie zachodzić). Następnie definiuje się [ilość pasm] wartości żądanej charakterystyki na każdym pasmie (wartość 1.0 -> nic nie zmieniaj). Ostatnia część to definicja [ilość pasm] wag.

- Uwagi
  a) Czasami program zgłasza komunikat, że przekroczona została maksymalna ilość iteracji i wyniki mogą być inne od oczekiwanych. Wynika to z niedoskonałości metody Remeza i może być spowodowane np. zbyt dużymi odstępami między pasmami.
  b) Zakłada się, że pasma są z przedziału $[0; 0.5]$ oraz że się nie pokrywają. Liczby bliskie 0.5 odpowiadają dużym częstotliwościom, a liczby bliskie 0 odpowiadają małym częstotliwościom.

- Kody źródłowe:
  Pliki remez.cpp,remez,waveform.cpp,waveform są autorstwa pana Jake Janovetz.
  Plik fft.cpp,fft pochodzi z książki *Numerical recipes in C*
  Plik filtruj.cpp jest autorstwa Wita Jakuczuna i Wojciecha Žiški

Literatura

[1] W. Borodziewicz, K. Jaszczak „Cyfrowe przetwarzanie sygnałów”

[2] Oppenheim A.V., Schafer R.W. „Cyfrowe przetwarzanie sygnałów”

[3] `http:\\www.dspguide.com`