VEŽBA 7 – Obrada signala u frekvencijskom domenu metodom overlap-add

Potrebno predznanje

- Poznavanje programskog jezika C
- Diskretna Furijeova transformacija
- FIIR filteri

Šta će biti naučeno tokom izrade vežbe

- Primena brze Furijeove transformacije u cilju efikasne implementacije FIR filtra korišćenjem osobina Furijeove transformacije i konvolucije.
- Kako rekonstrukuisati signal korišćenjem metode preklapanja i dodavanja (eng. *overlap-add*).
- Biće prikazan primer obrade signala koja zahteva manipulaciju spektrom, koju nije moguće direktno implementirati u vremenskom domenu

Motivacija

Obrada signala u frekvencijskom domenu predstavlja veoma važan aspekt obrade digitalnih signala. Pre svega, zbog osobina Furijeove transformacije često korišćene operacije kao što su konvolucija i korelacija u frekvencijskom domenu imaju jednostavniji oblik nego u vremenskom domenu, što omogućava računski efikasniju implementaciju FIR filtara, autokorelacije i sličnih operacija. Takođe, obrada u frekvencijskom domenu omogućava implementaciju algoritama koji zahtevaju direktnu manipulaciju nad amplitudskim ili faznim spektrom signala, što nije uvek moguće postići u vremenskom domenu. Treća važna primena je kompresija signala jer se obradom u transformisanom domenu može bolje konstrolisati raspodela greške kvantizacije i time postići perceptualno bolji kvalitet kompresovanog signala. Pošto praktično svi algoritmi za kompresiju audio signala koriste neku varijantu overlap-add metode, razumevanje ove metode je ključno za razumevanje funkcionisanja audio kodera.

1 Teorijske osnove

Kao što je rečeno u ranijim vežbama, digitalni filtri prema formi funkcije prenosa mogu da se podele u dve kategorije – filtre sa konačnim impulsnim odzivom (eng. Finite Impulse Response, FIR) i filtre sa beskonačnim impulsnim odzivom (eng. Infinite Impulse Response, IIR). Prednost IIR filtara je to što omogućavaju postizanje mnogo bolje karakteristike sa manjim redom filtra, dok je prednost FIR filtara to što su uvek stabilni i što omogućavaju postizanje linearne fazne karakteristike. S druge strane, osnovna mana FIR filtara je njihova računska kompleksnost jer je za postizanje karakteristike uporedive sa IIR filtrima osmog ili desetog reda potreban FIR filtar sa nekoliko stotina koeficijenata.

Izlazni signal FIR filtra u vremenskom domenu *y(n)* računa se kao konvolucija ulaznog signala *x(n)* i impulsnog odziva filtra *h(n)*:

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

Kao što se vidi iz ove formule, za implementaciju FIR filtra N-tog reda potebno je N množenja i N-1 sabiranje po odbirku. Međutim, prelaskom u frekvencijski domen konvolucija prelazi u obično množenje:

$$X(k) = DFT\{x(n)\}, \qquad H(k) = DFT\{h(n)\}$$

$$Y(k) = X(k)H(k)$$

$$y(n) = DFT^{-1}\{Y(k)\}$$

S obzirom da algoritam za brzu Furijeovu transformaciju ima složenost O(NlogN), za filtriranje u frekvencijskom domenu potrebno je približno $2\log_2 N+1$ kompleksnih, odnosno $8\log_2 N+4$ realnih množenja po odbirku. U tabeli 1 dato je poređenje broja množenja po odbirku potrebnih za filtriranje u vremenskom i frekvencijskom domenu u zavisnosti od reda FIR filtra.

Tabela 1 - Poređenje kompleksnosti filtriranja u vremenskom i frekvencijskom domenu

Red filtra	Broj množenja (vremenski domen)	Broj množenja (frekvencijski domen)
32	32	44
64	64	52
128	128	60
256	256	68
512	512	76

Da bi se eliminisala izobličenja u izlaznom signalu koja mogu nastati kao posledica diskontinuiteta na ivicama blokova obrade, filtriranje u frekvencijskom domenu vrši se tako da postoji izvesno preklapanje među susednim blokovima (obično 50%), a blokovi odbiraka se množe s prozorskom funkcijom pre i posle obrade u frekvencijskom domenu. Proces podele signala na blokove i transformacije u frekvencijski domen prikazan je na slici 1. Signal se najpre deli na blokove dužine N koji se obično nazivaju okviri (eng. frame).

Zatim se tekući i prethodni okvir spajaju u jedan blok obrade $x_m(n)$ dužine L=2N i množe prozorskom funkcijom $w(n)$:

$$x_m(n) = x((m-1)N + n)w(n), \;\; n = 0,1,..2N-1$$

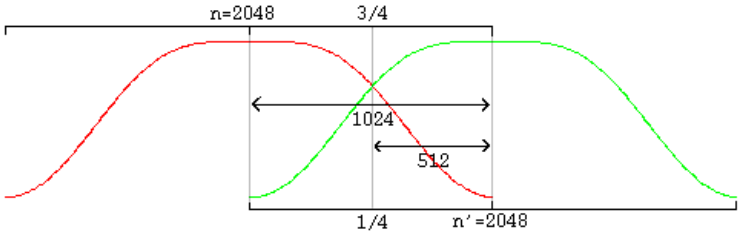
Da bi se obezbedila rekonstrukcija bez izobličenja, prozorska funkcija mora da zadovoljava uslov

$$w(n)^2 + w(N-1-n)^2 = 1$$

Primer jedne takve prozorske funkcije jeste Vorbis (projektovana za potrebe istoimenog dekodera). Ova prozorska funkcija data je sa:

$$w(n) = sin\left(\frac{\pi}{2}\sin^2\left(\frac{\left(n+\frac{1}{2}\right)\pi}{N}\right)\right), \;\; n = 0..N-1$$

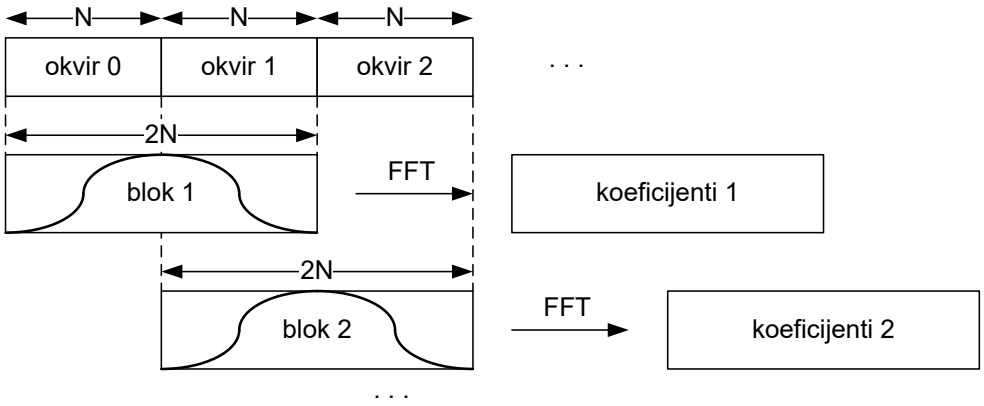
Na sledećoj slici prikazan je izgled prozorske funkcije (prikazane su preklopljene 2 funkcije tačno na polovini, na isti način kako ćemo u narednom koraku vršiti preklapanje i sabiranje uzastopnih okvira). Sa slike se može videti da je jednako pojačanje tačno na ¼ okvira.



Slika 1 - prozorska funkcija Vorbis

Posle prozoriranja, blok obrade se transformiše u frekvencijski domen brzom Furijeovom transformacijom dužine 2N, čime se dobija 2N kompleksnih koeficijenata:

$$X_m(k) = DFT\{x_m(n)\}$$



Slika 2 - Podela ulaznog signala na blokove obrade

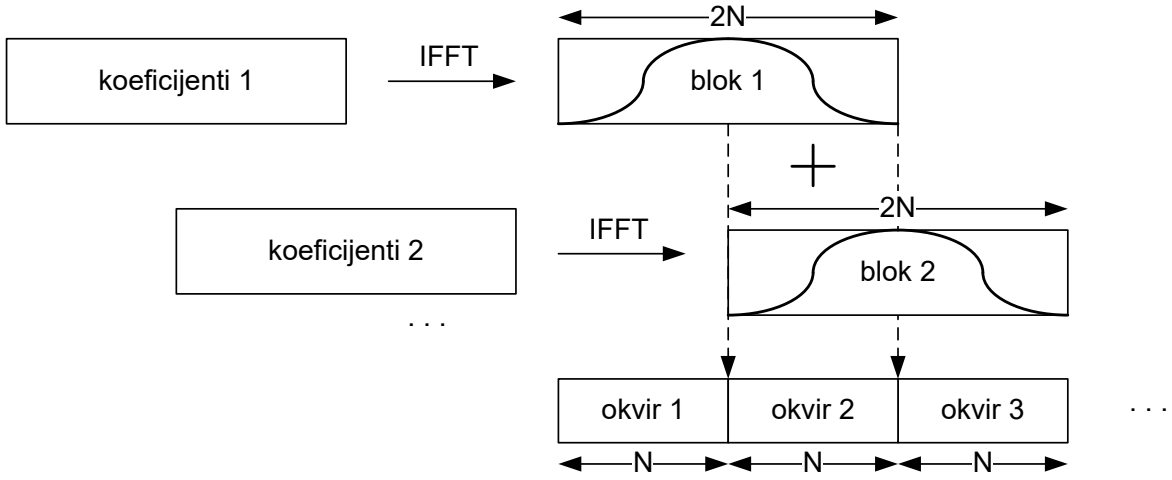
U prethodnim poglavljima prikazano je da kod implementacije algoritama za rad u realnom vremenu, funkcija obrade se pravi tako da vrši obradu nad blokom podataka. Potom, kada god je na ulazu u sistem spreman blok novih podataka, funkcija se poziva. Kod opisanog algoritma, izlazni blok ne zavisis samo od trenutnog ulaznog bloka, već i od prethodnog. Zbog toga, neophodno je napraviti bafer za pamćenje prethodnog bloka, i ažurirati ga na kraju funkcije obrade, kako bi u narednoj iteraciji ti podaci mogli biti iskorišćeni. Sastavljanje tekućeg i prethodnog okvira, u programskom jeziku C se svodi na prepisivanje sadržaja dva niza dužine N u treći niz, dužine 2N. Istovremeno, prilikom prepisivanja vrši se i množenje prozorskom funkcijom.

```
for (i = 0; i < N; ++i)
{
    buffer[i] = _smpy(in[i], window[i]);
    buffer[N+i] = _smpy(in_delay[i] * window[N+i]);
}
```

Nakon poziva funkcije za računanje brze furijeove transformacije, dobija se spektar signala predstavljen sa N kompleksnih brojeva. Taj niz od N kompleksnih brojeva u programskom jeziku C predstavljen je kao niz dužine 2N, gde članovi niza sa parnim indeksom 2k predstavljaju realni deo k-tog kompleksnog broja, a članovi sa neparnim indeksom 2k+1 imaginarni deo k-tog kompleksnog broja (pogledati vežbu 5).

$Re(y_0)$	$Re(y_{nx/2})$	$Re(y_1)$	$Im(y_1)$	$Re(y_2)$	$Im(y_2)$	...	$Re(y_{nx/2-1})$	$Im(y_{nx/2-1})$

Posle obrade u frekvencijskom domenu obrađeni blok odbiraka $Y_m(k)$ rekonstruiše se metodom *Overlap-add* kao što je prikazano na slici 3.



Slika 3 - Rekonstrukcija izlaznog signala Overlap-add metodom

Obrađeni blok koeficijenata se prvo vrati u vremenski domen primenom inverzne brze Furijeove transformadije kako bi se dobio izlazni blok $y_m(n)$ dužine L=2N odbiraka:

$$y_m(n) = DFT^{-1}\{Y_m(k)\}$$

Zatim se prvih N odbiraka tekućeg bloka i drugih N odbiraka prethodnog bloka ponovo množe prozorskom funkcijom w(n) i sabiraju kako bi se rekonstruisao izlazni okvir od N odbiraka:

$$y(mN+n) = y_m(n)w(n) + y_{m-1}(n+N)w(n+N),\ \ n = 0,1,..N-1$$

Kako bi se omogućilo korišćenje drugih N odbiraka prethodnog bloka prilikom računanja izlaznog okvira, neophodno je uvesti bafer za čuvanje ovih vrednosti (u narednom primeru *out_delay*).

```
for (i = 0; i < N; ++i)
{
    out[i] = _smpy(buffer[i], window[i]) + _smpy(out_delay, window[N+i]);
}
```

Pre završetka funkcije neophodno je ulazni okvir smestiti u bafer za čuvanje prethodnog ulaznog okvira, kako bi mogao biti iskorišćen prilikom narednog poziva funkcije. Pored toga, potrebno je drugu polovinu bloka dobijenog računanjem inverzne Furijeove transformcije smestiti u bafer za čuvanje prethodnog izlaznog okvira

Zadaci

Zadatak 1

U ovom zadatku potrebno je realizovati prelazak u spektralni domen i rekonstrukciiju signala koristeći *overlap-add* metodu. Opisani algoritam potrebno je realizovati u okviru funkcije

- `void overlap_add(Int16* in, Int16* out, Int16 N)`

Funkcija ima sledeće argumente:

- `in` – ulazni okvir
- `out` – izlazni okvir
- `N` – dužina okvira

U implementaciji koristiti sledeće pomoćne bafere koji su definisani u izvornoj datoteci:

- `fft_buffer` – niz dužune 2N, služi za obradu
- `in_delay` – prethodni ulazni okvir dužine N
- `out_delay` – drugih N odbiraka prethodnog izlaznog bloka

Funkciju realizovati prateći sledeće korake:

1. Sadržaj niza za pamćenje prethodnog ulaznog okvira prepisati u prvu polovinu niza *fft_buffer*
2. Sadržaj ulaznog okvira prepisati u drugu polovinu niza *fft_buffer*
3. Sadržaj ulaznog okvira prepisati u niz za pamćenje prethodnog ulaznog okvira.
4. Pomnožiti novoformirani niz podataka (*fft_buffer*)prozorskom funkcijom

5. Pozvati funkciju za računanje brze Furijeove transformacije (rfft). Funkcija računa spektrar signala. N kompleksnih spektralnih koeficijenata predstavljeno je kao niz od 2N brojeva, gde elementi sa parnim indeksom predstavljaju realni deo, a elementi sa neparnim indeksom imaginarni deo komplesnog broja (za više detalja pogledati vežbu 3).
6. Pozvati funkciju za račnanje inverzne Furijeove transformacije (rifft).
7. Prvu polovinu dobijenog okvira (prvih N odbiraka) pomnožiti sa prvom polovinom prozorske funkcije. Sadržaj niza za pamćenje prethodnog izlaznog okvira pomnožiti sa drugom polovinom prozorske funkcije. Sabrati ova dva niza i rezultat upisati u izlazni niz (*out*).
8. NAPOMENA: rfft i rifft smanjuju signal 2 puta (obe). Da bi se vratila originalna amplituda signala potrebno je sve odbirke u nizu *out* pomnožiti sa 4.
9. Drugu polovinu niza *fft_buffer* prepisati u niz za pamćenje prethodnog izlaznog okvira.
10. Rešenje testirati datim ulaznim datotekama. U izlaznoj datoteci ne sme biti primetnih izobličenja.

Zadatak 2

Proširiti funkciju implementiranu u prethodnom zadatku tako da vrši filtriranje ulaznog signala pojasnim filtrom sa opsegom frekvencija 300 – 4000 Hz. Frekvencija odabiranja u Hz definisana je simbolom *SAMPLE_RATE*.

1. Izračunati redni broj spektralnih koeficijenata k1 i k2 koji odgovaraju graničnim frekvencijama filtera.
2. Između poziva funkcija fft i ifft, sve spektralne koeficijente između k1 i k2 nulirati.
3. Pokrenuti program i izvršiti spektralnu analizu ulaznog i izlaznog signala.

Zadatak 3

U ovoj vežbi potrebno je realizovati zvučni efekat „robotski glas“. Proširiti funkciju implementiranu u zadatku 1 tako da se spektar ulaznog signala modifikuje tako da amplitudski spektar ostane nepromenjen a da faza koeficijenta X(n) bude $0.25\pi n^2$.

1. Između poziva funkcija fft i ifft iterirati kroz spektar signala. Za svaki spektralni koeficijent k_n:
   a. Izračunati amplitudu kao $A = \sqrt{Re^2(k_n) + Im^2(k_n)}$
   b. Izračunati novu vrednost faze kao $Ph = 0.25\pi n^2$
   c. Izračunati novu vrednost realnog i imaginarnog dela na sledeći način:
      $Re(k_n) = A*cos(Ph)$
      $Im(k_n) = A*sin(Ph)$

Za računanje vrednosti sinusa I kosinusa koristiti funkcije iz math.h zaglavlja. Za računanje kvadratnog korena koristiti funkciju *sqrt_16* iz DSPlib.h.

2. Rešenje testirati datim ulaznim datotekama. Poslušati izlazne datoteke, uporediti sa ulaznim i komentaristati postignuti efekat.

Prikazati amplitudni i fazni spektar ulaznog i izlaznog signala i uporediti ih.