VEŽBA 3 – Diskretna Furijeova Transformacija

Potrebno predznanje

- Poznavanje programskog jezika C
- Urađena Vežba 1 – Uvod u Digitalnu Obradu Signala
- Odslušana predavanja iz predmeta OAiS DSP na temu Diskretni signali i spektri
- Odslušana predavanja iz predmeta OAiS DSP na temu Brza Furijeova Transformacija (FFT)

Šta će biti naučeno tokom izrade vežbe

U okviru ove vežbe naučićete:

- Kakvi se spektralni koeficijenti preslikavaju na vrednost frekventne komponente kod diskretne Furijeove transformacije
- Kakav je uticaj veličine bloka na vrednosti izračunate diskretne Furijeove transformacije
- Kakav je uticaj prozorskih funkcija na vrednosti izračunate diskretne Furijeove transformacije
- Šta je to brza Furijeova transformacija (FFT)
- Upotreba funkcije za računanje FFT na TMS320C5535

Motivacija

Analizom signala u vremenskom domenu mozemo odrediti samo deo karakteristika signala (npr. snagu i amplitudu). Furijeova analiza (matematički aparat) je pokazala da se svaki složeni signal može predstaviti kao zbir prostoperiodičnih signala. Diskretna Furijeova transformacija nam omogućuje da saznamo od kojih prostoperiodičnih komponenti (sinus, kosinus) se sastoji signal koji analiziramo. Kombinacijom direktne i inverzne Furijeove transformacije moguće je modifikovati pojedine (željene) komponente signala u spektralnom domenu.

1 TEORIJSKE OSNOVE

1.1 Diskretna Furijeova Transformacija

Diskretna Furijeova transformacija (eng. *discrete Fourier transform*, DFT) je jedna od najčešće korišćenih transformacija u digitalnoj obradi signala. Ona omogućava da se diskretni periodični signal predstavi sumom sinusnih komponenti pridružujući svakoj komponenti odgovarajuću amplitudu i fazni pomeraj. Opšta formula za izračunavanje DFT je:

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

U opštem slučaju, DFT transformiše niz od N kompleksnih odbiraka u vremenskom domenu u niz od N kompleksnih koeficijenata u frekvencijskom domenu.

$$s(k) = x(k) + j \cdot y(k) \;<=>\; S(f) = X(f) + j \cdot Y(f) \tag{2}$$

$$X(f) = \frac{S(f) + S^*(-f)}{2} \qquad Y(f) = \frac{S(f) - S^*(-f)}{2j} \tag{3}$$

Uvođenjem smene:

$$e^{j\theta} = \cos\theta + j\sin\theta \tag{4}$$

Možemo zasebno izdvojiti funkcije za izračunavanje realnog i imaginarnog dela:

$$X(n) = \sum_{\mathrm{k=0}}^{\mathrm{N-1}} x(k) * \cos(2\pi\frac{nk}{N}) + y(k) * \sin(2\pi\frac{nk}{N}) \tag{5}$$

$$Y(n) = \sum_{\mathrm{k=0}}^{\mathrm{N-1}} -x(k) * \sin(2\pi\frac{nk}{N}) + y(k) * \cos(2\pi\frac{nk}{N}) \tag{6}$$

U slučaju da su odbirci ulaznog signala realni (*y(k)=0*), za njegovu DFT će važiti X(N-k) = X(k)*. To znači da samo N/2 kompleksnih koeficijanata sadrži informaciju o spektru signala, dok je ostatak koeficijenata redundantan. To je moguće iskoristiti za računanje istovremene transformacije dva realna signala. Niz kompleksnih odbiraka s(k) se pravi tako da se niz odbiraka prvog realnog signala posmatra kao realna komponenta, a niz odbiraka drugog realnog signala posmatra kao imaginarna komponenta prema (7).

$$s_k = x_k + j \cdot y_k \;<=>\; S_n = X_n + j \cdot Y_n \qquad \{k, n\} = 0,1,\dots,N-1 \tag{7}$$

$$X_0 = \frac{S_0 + S_0^*}{2} \qquad Y_0 = \frac{S_0 - S_0^*}{2j} \tag{8}$$

$$X_n = \frac{S_n + S_{N-n}^*}{2} \qquad Y_n = \frac{S_n - S_{N-n}^*}{2j} \qquad n = 1,2,\dots,\frac{N}{2} \tag{9}$$

Dobijeni spektar je diskretne prirode. Za razliku od Furijeove transformacije kontinualnog vremena, kod koje je svakoj sinusnoj komponenti iz ulaznog signala bilo moguće pridružiti jedan koeficijent u

frekvencijskom domenu, u slučaju diskretne Furijeove transformacije slično pridruživanje (jedna sinusna komponenta je predstavljena jednim spektralnim koeficijentom) nije uvek moguće.

Pri proračunu DFT-a potrebno je da se N odabere tako da se napravi kompromis između lokacije pojedinih DFT vrednosti i vremena potrebnog da se izračuna jedna DFT vrednost. Što je veći N to je veća rezolucija u frekventnom domenu i manji je razmak između DFT komponenata, ali je i veće vreme računanja. Frekvencija f_k, koja odgovara *k*-tom koeficijentu DFT-a, dobija se sledećim izrazom:

$$f_k = \frac{k f_s}{N}, \quad \text{za } k = 0,1,\dots,N-1$$

gde je f_s frekvencija odabiranja. U slučaju kada ulazni sinusni signal ima frekvenciju f_k' koja se razlikuje od frekvencije definisane prethodnom formulom, njegova DFT sadrži značajne koeficijente na poziciji *k* koja je aritmetički najbliža vrednosti *k'*, gde je:

$$k' = \frac{N f_{k'}}{f_s}$$

ali i ne nulte vrednosti (komponente) na susednim *k* vrednostima zbog preklapanja komponenti spektra.

1.1.1 Implementacija Diskretne Furijeove transformacije

Dat je primer implementacije Diskretne Furijeove transformacije u okviru programskog jezika C. Kompleksni odbirci signala predstavljeni su sa dva niza, *inreal* koji sadrži realne komponente i *inimag* koji sadri imaginarne komponente. Analogno tome izlazni elementi su predstavljeni sa nizovima *outreal* i *outimag*. Parametar *N* predstavlja veličinu ulaznog i izlaznog signala (broj odbiraka).

```
void dft(float inreal[], float inimag[], float outreal[], float outimag[], Int16 N)
{
    int n, k;
    for ( n = 0; n < N; n++) {
        float sumreal = 0;
        float sumimag = 0;
        for (k = 0; k < N; k++) {
            sumreal +=  inreal[k] * cos(2 * PI * n * k / N) + inimag[k] * sin(2 * PI * n * k /
N);
            sumimag += -inreal[k] *  sin(2 * PI * n * k / N) + inimag[k] * cos(2 * PI * n * k
/ N);
        }
        outreal[n] = sumreal;
        outimag[n] = sumimag;
    }
}
```

Funkcija *dft* implementirana je koristeći jednačine (5) i (6). Operator sumiranja $\sum$ u programskom jeziku C se implementira kao *for* petlja u okviru koje se vrši akumulacija vrednosti izraza koji stoji uz ovaj operator. Iterator i granica kod operatora sumiranja koriste se kao iterator i granica kod *for* petlje.

$Y = \sum_{i=1}^{10} x(i)$	`for (i = 1; i <= 10; i++) {`
	`  y+=x(i);`
	`}`

Kada se vrši akumuliranje vrednosti neophodno je voditi računa da se na početku svake iteracije izvrši postavljanje vrednosti promenljivih koje služe za čuvanje rezultata na nulu.

1.1.2 Implementacija Diskretne Furijeove transformacije koristeći celobrojne tipove

Signali su često predstavljeni sa celobrojnim vrednostima, kao što je slučaj sa signalom primljenim od strane AIC3204 kodeka. Rad sa celobrojnim vrednostima kod DSP procesora koji nemaju hardversku podršku za brojeve sa pokretnim zarezom (eng. *floating point*), zahteva znatno manje vreme izvršavanja u odnosu na rad sa tipovima sa nepokretnim zarezom.

Implementacija diskretne Furijeove transformacije za rad nad celobrojnim vrednostima sadrži određene razlike u odnosu na prethodno datu implementaciju. Lista parametara funkcije je jednaka, s tim da su svi parametri označene celobrojne vrednosti (u ovom primeru Int16).

- `void dft(Int16 inreal[], Int16 inimag[], Int16 outreal[], Int16 outimag[], Int16 N)`

Druga razlika koja se javlja jeste računanje vrednosti kosinusa. U implementaciji sa nepokretnim zarezom korišćene su standardne funkcije *sin* i *cos* iz biblioteke *math.h.* U ovom slučaju date su funkcije koje računaju vrednosti sinusa i kosinusa na osnovu tabele pretraživanja:

- `Int16 sin_LT(Uint16 x);`
- `Int16 cos_LT(Uint16 x);`

Obe funkcije kao parametar primaju vrednost ugla u radijanima u opsegu [0, 2π), skaliranu na opseg [0, 2^{16}). U ovakvoj predstavi vrednost π predstavljen je brojem 2^{15} (32768), π/2 brojem 2^{14} (16384), itd. Parametar koji se prosleđuje kao parametar ovih funkcija potrebno je prethodno skalirati na pomenuti opseg tako što se vrednost pomoži sa 2^{16} i potom podeli sa 2 π.

$$2\pi\frac{nk}{N} * \frac{65536}{2\pi} = nk * \frac{65536}{N}$$

S obzirom da je *N* konstantno tokom izvršenja funnkcije nije neophodno uvek računati izraz 65536/N, već ga je moguće izračunati jednom na početku funkcije i kasnije taj rezultat koristiti. Dakle poziv funkcija za računanje sinusa i kosinusa u okviru funkcije za računanje DFT vrši se na sledeći način:

```
Uint16 scale_factor = 65536/N;
...
cos_LT(n * k * scale_factor);
sin_LT(n * k * scale_factor);
```

Kao povratnu vrednost funkcije za računanje sinusa i kosinusa vraćaju vrednost sinusa, odnosno kosinusa ugla skaliranog na opseg [-2^{15}, 2^{15}). U ovakvoj predstavi broj -2^{15} predstavlja vrednost -1.0, a 2^{15} vrednost 1.0.

Kod izvršenja operacije množenja sa ovako predstavljenim brojem, neophodno je nakon množenja rezultat podeliti sa brojem koji predstavlja vrednost 1.0 kako bi se dobio stvaran rezultat. Na primer ako se broj X množi sa 0.5 očekivani rezultat je X/2. U slučaju da je 0.5 predstavljen sa 16384, kako bi se

dobio očekivani rezultat potrebno je podeliti rezultat množenja sa brojem koji predstavlja 1 (u ovom slučaju 32768), i tako se dobija X*16384/32768 = X/2. S obzirom da broj 32768 je ujedno i 2^{16} umesto deljenja moguće je izvršiti i logički pomeraj u desno za 16 mesta (X*16384>>16).

Prilikom množenja brojeva celobrojnog tipa neophodno je voditi računa o mogućem prekoračenju opsega. Da bi se sa sigurnošću izbeglo prekoračenje prilikom množenja dva 16-bitna broja, neophodno je rezultat smestiti u 32-bitnu promenljivu. Potrebno je voditi računa i o konverziji tipova prilikom računanja. U okviru programskog jezika C, implicitna konverzija tipova se dešava samo u slučaju izvršenja operacije čiji su operandi različitog tipa.

```
Int16 x, y;
Int32 z = x*y;
```

U prethodnom primeru prvo će se izvršiti operacija množenja. S obzirom da su tipovi oba operanda Int16, izračunati rezultat će biti takođe celobrojna 16-bitna vrednost. Tek prilikom operacije dodele vrednosti promenljivoj z doći će do implicitne konverzije u 32-bitni tip. Dakle ukoliko se vrši ovakvo množenje prekoračenje opsega neće biti izbegnuto. Ovaj problem se rešava eksplicitnom konverzijom jednog od operanada izraza čiji rezultat zahteva da bude veće preciznosti. Eksplicitna konverzija tipova (*explicit cast*) u C-u vrši se tako što se za datom izrazu doda prefiks u formi (*naziv_tipa*). Naredni primer ilustruje eksplicitnu konverziju tipa promenljive x. Prilikom operacije između dve promenljive različitih tipova, tip manje preciznosti implicitno se konvertuje u tip veće preciznosti, i samim tim, u ovom slučaju dobija se 32-bitni rezultat množenja.

```
Int16 x, y;
Int32 z = (Int32)x*y;
```

Množenje brojem koji predstavlja vrednost u opsegu [-1.0, 1.0) skaliranu na opseg [-2^{15}, 2^{15}), zahteva izvršenje operacija množenja, deljenja (ili logički pomeraj ukoliko je broj sa kojim se deli stepen broja 2), i vođenje računa o konverziji tipova. Jednostavniji način da se izvrši množenje drugog broja celobrojnog tipa jeste korišćenje unutrašnje kompajlerske funkcije (eng. *intrinsic*) namenjene toj operaciji. Unutrašnje funkcije predstavljaju funkcije čije telo nije dato u kodu ili nekoj biblioteci, već su deo samog kompajlera, i kompajler sam zna na koji način treba da prevede poziv te funkcije. Najčešće su to proste operacije na osnovu kojih se generišu instrukcije specifične za dati procesor. Funkcija koja vrši množenje dva 16-bitna broja, i kao rezultat vraća gornjih 16 bita data je sa:

- `Int16 _smpy(Int16 x, Int16 y);`

Dat je primer akumuliranja vrednosti nakon primene pomenutih izmena koje se tiču celobrojnih vrednosti.

Brojevi u pokretnom zarezu:

```
sumreal +=  inreal[k] * cos(2 * PI * n * k / N) + inimag[k] * sin(2 * PI * n * k / N);
```

Celobrojni tipovi:

```
sumreal +=  _smpy(inreal[k], cos_LT(n * k * scale_factor)) + _smpy(inimag[k], sin_LT(n * k *
scale_factor));
```

1.2 Prozoriranje

Prilikom primene diskretne Furijeove transformacije na signale čiji sadržaj se ne ponavlja sa periodom N signal se podeli na blokove dužine *N* i na svaki blok se primeni DFT. Da bi se izbegao uticaj blokovske obrade signala koristi se tehnika pod nazivom prozoriranje. Prozoriranjem se blok odbiraka koji se obrađuje uobličava u cilju smanjenja diskontinuiteta na početku i kraju bloka (pretpostavka prilikom računanja DFT je da se signal sastoji od periodično ponovljenih blokova odbiraka koji se analiziraju, što u opštem slučaju nije tačno). Prozorske funkcije se razlikuju po širini prve arkade *B* koja određuje spektralnu rezoluciju i obvojnicom O koja određuje brzinu opadanja uticaja ostalih spektralnih komponenti. Najčešće korišćene prozorske funkcije date su u tabeli 1.

Tabela 1 – Prozorske funkcije

Naziv	Prozorska funkcija	B	O
Pravougao	$w(t) = 1$	$\frac{2F_s}{N}$	f^{-1}
Hann	$w(t) = \sin^2\left(\pi \frac{t}{T_w}\right)$	$\frac{4F_s}{N}$	f^{-3}
Hamming	$w(t) = 0.54 - 0.46 \cdot \cos\left(2\pi \frac{t}{T_w}\right)$	$\frac{4F_s}{N}$	f^{-1}
Blackmann	$w(t) = 0.42 - 0.5 \cdot \cos\left(2\pi \frac{t}{T_w}\right) + 0.08 \cdot \cos\left(2\pi \frac{2t}{T_w}\right)$	$\frac{6F_s}{N}$	f^{-3}
Gauss	$w(t) = e^{-\lambda\left(\frac{t-T_w/2}{T_w/2}\right)^2}$	$\frac{2\lambda F_s}{N}$	$e^{-\frac{f^2}{\lambda}}$

1.3 Brza Furijeova Transformacija (FFT)

Brza Furijeova transformacija (FFT) je algoritam koji omogućuje efikasnije računanje DFT, baziran na simetričnosti DFT koeficijenata. FFT je sukcesivni algoritam koji se realizuje u više uzastopnih koraka, pri čemu se u svakom koraku računaju DFT manje dužine. Decimacija dužine DFT može biti ili u vremenskom ili u frekvencijskom domenu. Jedan od algoritama za brzo računanje DFT-a jeste Radix-2 algoritam.

Radix-2 predstavlja FFT algoritam u kojem se u svakom koraku računaju DFT sa prepolovljenom dužinom. Primenjiv je samo za DFT čija je dužina $N=2^E$. Sa njime se zahteva $\left(\frac{N}{2}\right) ld(N)$ množenja i sabiranja umesto N^2. Osnovna operacija u okviru radix-2 FFT jeste *Butterfly* operacija u kojoj se dve vrednosti izračunavaju na osnovu dve vrednosti iz prethodnog koraka, a koristi se samo jedno množenje. U ld(N) sukcesivnih koraka se uvek računa N/2 *butterfly* operacija, čime je i definisan broj množenja potreban za FFT.

Tačnost FFT zavisi od dužine N zbog nagomilavanja grešaka kod množenja i sabiranja (obrada sa fiksnim zarezom). Ova degradacija tačnosti se donekle može ublažiti primenom obrada sa blokovskim fiksnim zarezom.

Kompleksna FFT je efikasna primena FFT u obradi gde se koristi osobina opisana u jednačini (3) za izračunavanje dve DFT od dva diskretna signala tako što se FFT primeni na kompleksne odbirke napravljene tako da realni deo čini jedan signal a imaginarni deo drugi.

U okviru biblioteke *dsplib* za TMS320C5535 platformu data je funkcija koja sadži implementaciju algoritma za računanje brze Furijeove transformacije. Data implementacija koristi i određena hardverska proširenja koja nudi pomenuti procesor kako bi njeno izvršavanje bilo efikasnije.

- `void rfft(DATA *x, ushort nx, ushort scale);`

gde je:

- *x* – Pokazivač na ulazni vektor koji sadrži *nx* realnih elemenata ulaznog signala. Izračunate vrednosti FFT-a koje predstavljaju izlaz funkcije takođe su smeštene u ovaj vektor. Na izlazu u *x* vektor smeštena je prva polovina (*nx/2*) kompleksnih odbiraka. Pošto realna Furijeova transformacija predstavlja simetričnu funkciju oko Nikvistove granične frekvencije, prva polovina elemenata FFT-a sadrži sve informacije o signalu. Kompleksni FFT elementi na izlazu nalaze se u sledećem formatu:

y(0)Re	y(nx/2)Im	y(1)Re	y(1)Im	y(2)Re	y(2)Im	...	y(nx/2)Re	y(nx/2)Im

- *nx* – Broj elemenata u ulaznom vektoru *x*. Podržane veličine bloka su 16, 32, 64, 128, 256, 512, 2048
- *scale* – Za *scale*=1 rezultati su skalirani prilikom svakog međukoraka FFT-a kako bi se sprečilo prekoračenje opsega

Ograničenje ove funkcije jeste da ulazni vektor mora biti poravnat na 32 bita.

2 ZADACI

Napomena: U toku izrade, za svaki signal za koji je rečeno da je potrebno iscrtati, potrebno je sačuvati prikaz kao slikovnu datoteku. Za čuvanje koristiti *PrtScn* i neki od prostih alata za obradu slike (dovoljan je i *MS Paint*). Nazivi slikovnih datoteka treba da budu u formatu *Zadatak_X_Slika_Y.*

2.1 Zadatak 1

U okviru ovog zadatka ispitaćemo uticaj veličine bloka i prozorske funkcije na rezultat FFT transformacije.

1. Uvući projektni zadatak 1 u radni prostor
2. Postaviti za ulaznu datoteku: *sine_1kHz.wav*
3. Pokrenuti program
4. Zaustaviti izvršenje upotrebom tačke prekida i iscrtati vrednost signala na jednom od ulaznih kanala u vremenskom domenu kodisteći alat *Tools->Graph->Single Time* sa sledećim parametrima:
   a. *Acquisition Buffer Size = 1024*
   b. *DSP Data Type = 16-bit signed integer*
   c. *Sampling rate = 48000*
   d. *Display Data Size = 1024*
5. Iscrtati vrednost amplitude signala u frekventnom domenu upotrebom alata *Tools -> Graph -> FFT Magnitude*

Alat *FFT Magnitude* vrši učitavanje vrednosti odbiraka sa DSP uređaja, zatim vrši brzu Furijeovu transformaciju nad odbircima i iscrtava vrednost amplitude na grafiku. Moguće je podešavati parametre računanja FFT-a.

   a. Podesiti parametre definisane pod 5a, 5b i 5c isto kao u prethodnom koraku.
   b. Postaviti stil iscrtavanja *(Data Plot Style)* na *Bar*. Kada se koristi ova opcija svaki koeficijent DFT-a predstavljen je tačno jednom linijom što nam omogućava da lakše pročitamo sa grafika koliko spektralnih koeficijenata je prisutno i kojim frekvencijama odgovaraju
   c. Postaviti red računanja FFT-a na 6, tako da veličina FFT bloka iznosi $2^6 = 64$
6. Ponoviti korak 6 za veličine FFT bloka 256 i 1024 i komentarisati uticaj veličine FFT bloka na spektralne komponente koje se pojavljuju na grafiku.
7. Ponoviti korak 6 za različite vrednosti prozorske funkcije. Komentarisati uticaj prozorske funkcije na spektralnu karakteristiku signala.

2.2 Zadatak 2

U okviru ovog zadatka upoznaćete se sa korišćenjem funkcije za računanje FFT-a implementirane u okviru date biblioteke *dsplib* za TMS320C5535 platformu.

1. Uvući projektni zadatak 2 u radni prostor
2. U okviru *main* funkcije izračunati vrednost FFT-a nad jednim ulaznim kanalom upotrebom funkcije *rfft*. Veličina bloka (*nx*) definisana je sa *AUDIO_IO_SIZE* u okviru ezdsp5535_aic3204_dma.h datoteke.
3. Izračunati spektar snage signala koristeći osobinu: $|X|^2 = Re(X)^2 + Im(X)^2$. Prilikom računanja voditi računa o prekoračenju opsega tako što ćete rezultat množenja dva 16-bitna broja smestiti u 32-bitnu promenljivu.
4. Postaviti za ulaznu datoteku: *sine_1kHz.wav*
5. Pokrenuti program
6. Zaustaviti izvršenje upotrebom tačke prekida i iscrtati vrednost ulaznog signala koristeći alat FFT Magnitude kao u prethodnom zadatku. Podesiti red FFT-a da odgovara veličini bloka za koju ste računali FFT.
7. Iscrtati vrednost izračunatog spektra snage ulaznog signala koristeći alat *Tools->Graph->Single Time* sa sledećim parametrima:
   a. *Acquisition Buffer Size = nx/2*
   b. *DSP Data Type = 32-bit signed integer*
   c. *Sampling rate = 1*
   d. *Display Data Size = nx/2*
   e. *Data Plot Style = Bar*
8. Uporediti prikaz spektra signala upotrebom alata FFT-magnitude sa prikazom izračunatog spektra.
9. Ponoviti čitav postupak za različite ulazne stream-ove i različite veličine bloka za računanje FFT-a (AUDIO_IO_SIZE).

2.3 Zadatak 3

1. U okviru prethodnog zadatka, pre računanja FFT transformacije, izvršiti množenje odbiraka ulaznog signala sa Hamming-ovom prozorskom funkcijom. Vrednosti prozorske funkcije date su okviru niza *window*. Vodite računa da veličina ulaznog bloka i veličina prozorske funkcije budu jednake. Za množenje koristiti funkciju **`_smpy.`**
2. Pokrenuti program i prikazati vrednosti spektra ulaznog signala i izračunatu vrednost spektra kao u prethodnom zadatku. Za iscrtavanje vrednosti spektra upotrebom *FFT Magnitude* alata koristiti *Hamming* prozorsku funkciju. Uporediti dva prikaza.

2.4 Zadatak 4

Data je funkcija *dft* koja sadrži implementaciju diskretne Furijeove transformacije koristeći aritmetiku pokretnog zareza po jednačinama datim pod (5) i (6). U okviru ovog zadatka potrebno je implementirati funkciju za računanje DFT koristeći celobrojnu aritmetiku. Opis funkcije koju je potrebno implementirati dat je u poglavlju 1.1.2. Deklaracija funkcije data je sa:

- ```
  void dft_int(Int16 inreal[], Int16 inimag[], Int16 outreal[], Int16 outimag[], Int16 N);
  ```

gde je:

- `inreal` - ulazni vektor koji sadrži realne delove N ulaznih kompleksnih odbiraka
- `inimag` - ulazni vektor koji sadrži imaginarne delove N ulaznih kompleksnih odbiraka
- `outreal` - izlazni vektor koji sadrži realne delove N kompleksnih koeficijenata DFT
- `outimag` - izlazni vektor koji sadrži imaginarne delove N kompleksnih koeficijenata DFT
- `N` – dužina ulaznog i izlaznog vektora

1. Uvući projektni zadatak 3 u radni prostor
2. Analizirati funkciju **`dft`** unutar *dft.c* datoteke
3. Implementirati funkciju **`dft_int`** unutar *dft.c* datoteke.
4. U okviru *main* funkcije izračunati vrednost FFT-a nad jednim ulaznim kanalom upotrebom funkcije *dft_int*. Veličina bloka (*N*) definisana je sa *AUDIO_IO_SIZE* u okviru ezdsp5535_aic3204_dma.h datoteke. S obzirom da je ulazni signal realan, *dft_int* funkciji kao imaginarnu komponentu proslediti niz dužine *N* popunjen nulama.
5. Izračunati amplitudni spektar signala koristeći osobinu: $|X|^2 = Re(X)^2 + Im(X)^2$. Prilikom računanja voditi računa o prekoračenju opsega tako što ćete rezultat množenja dva 16-bitna broja smestiti u 32-bitnu promenljivu.
6. Postaviti za ulaznu datoteku: *sine_1kHz.wav*
7. Pokrenuti program
8. Zaustaviti izvršenje upotrebom tačke prekida i iscrtati vrednost ulaznog signala koristeći alat FFT Magnitude. Podesiti red FFT-a da odgovara veličini bloka za koju ste računali FFT.
9. Iscrtati prvu polovinu vrednosti izračunatog spektra snage ulaznog signala koristeći alat *Tools->Graph->Single Time* sa sledećim parametrima:
  ```

   a. *Acquisition Buffer Size = N/2*
   b. *DSP Data Type = 32-bit signed integer*
   c. *Sampling rate = 1*
   d. *Display Data Size = N/2*
   e. *Data Plot Style = Bar*
10. Uporediti prikaz spektra signala upotrebom alata FFT-magnitude sa prikazom izračunatog spektra.
11. Ponoviti čitav postupak za različite ulazne stream-ove i različite veličine bloka za računanje FFT-a (AUDIO_IO_SIZE).

3 Zaključak

Tokom izrade ovog zadatka upoznali ste se sa računanjem diskretne Fruijeove transformacije. Videli ste da DFT, za razliku od kontinualne Furijeove transformacije, sadrži konačan skup spektralnih koeficijenata. Svaki koeficijent odgovara jednoj učestanosti iz kontinualnog skupa vrednosti 0 - Nikvistova granična učestanost. Spektralne komponente signala koje se nalaze između dva koeficijenta razložene su na najbliže koeficijente. Videli ste na koji način veličina bloka DFT utiče na preciznost izračunatih vrednosti. Uvideli ste i na koji način različite prozorske funkcije umanjuju uticaj blokovske obrade na vrednosti izračunate DFT.

Na primeru obrade zvuka u realnom vremenu videli ste kako se može iskoristiti implementacija brze Furijeove transformacije (FFT) za računanje DFT na TMS320C5535 procesoru. Predstava signala u frekventnom domenu ima široku upotrebu u oblasti digitalne obrade signala. Neke od upotreba su algoritam za brzo izračunavanje konvolucije ili detekcija prisutnosti određenih spektralnih komponenti u signalu.