VEŽBA 1 – Generisanje i prikazivanje signala

Potrebno predznanje

- Poznavanje programskog jezika C
- Urađena Vežba: Uvod u Digitalnu Obradu Signala
- Odslušana predavanja iz predmeta OAiS DSP na temu Uvod, signali i sistemi

Šta će biti naučeno tokom izrade vežbe

U okviru ove vežbe biće naučeno sledeće:

- Osnovne osobine sinusnog signala
- Različiti načini za generisanje periodičnih signala i njihova implementacija
- Izgled sinusnog signala u vremenskom i frekventnom domenu
- Generisanje složenijih signala (*sweep*, *multiton, square*)

Motivacija

Analiza digitalnih sistema podrazumeva određivanje karakteristika na osnovu odziva sistema na neke karakteristične signale. Iako je moguće koristiti digitalne zapise karakterističnih signala na nekom od medija za skladištenje podataka, mnogo je jednostavnije generisati date signale i time obezbediti fleksibilnost njihovog korišćenja (jednostavna kontrola parametara signala kao što su frekvencija, amplituda i faza). U ovoj vežbi ćemo naučiti neke načine generisanja osnovnih signala koji se koriste u digitalnoj obradi signala. S obzirom da u ovom kursu izučavamo linearne, vremenski nezavisne sisteme i znamo da je odziv LVI sistema na složenu pobudu moguće odrediti na osnovu odziva sistema na osnovne (jednostavne) signale, generisanje osnovnih signala je dovoljan preduslov za analizu svih digitalnih sistema koje ćemo izučavati u ovom kursu.

1 TEORIJSKE OSNOVE

1.1 Sinusni signal

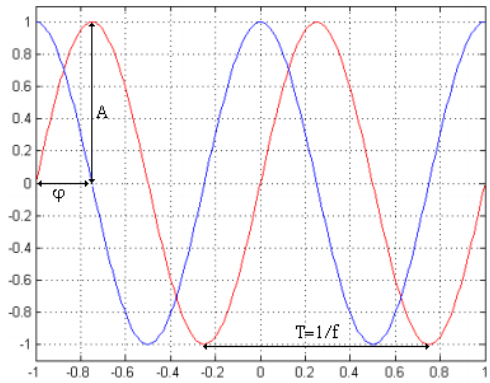
Sinusni signali spadaju među najčešće korišćene signale u oblasti digitalne obrade signala. Sinusni signal predstavlja kontinualan periodičan signal određen sa tri parametra:

- *A* - amplituda signala
- *f* – učestanost (frekvencija) signala. Predstavlja broj oscilacija u sekundi. Jednak je recipročnoj vrednosti periode signala *T*.
- φ – fazni pomeraj u opsegu –π do π

Uzevši u obzir ova tri parametra, sinusni signal kao funkciju vremena možemo predstaviti sledećom jednačinom:

$$y(t) = A \times \sin(2\pi f t + \varphi) \qquad (1)$$

Na slici su prikazana dva sinusna signala jednake frekvencije i amplitude. Signal predstavljen crvenom bojom predstavlja sinusni signal sa vrednošću faznog pomeraja 0. Plavom bojom je predstavljen signal čija je vrednost faznog pomeraja -π/2, odnosno kosinusni signal.



Slika 1 - Prikaz sinusnog signala u vremenskom domenu

Razlog zbog kojeg se sinusni signali toliko često koriste u oblasti digitalne obrade signala jeste činjenica da matematički sve talasne signale možemo predstaviti kao zbir skupa odabranih sinusnih signala određene amplitude, frekvencije i faznog pomeraja. Ovakav pristup nam omogućava da vršimo analizu ili izmene nad određenim signalom tako što ćemo analizirati ili menjati pojedinačne sinusne komponente tog signala, a ne čitav složeni signal. Ovakav pristup naziva se spektralna analiza sistema.

1.2 Sinusni signal u diskretnim sistemima

U diskretnom domenu, signal je definisan samo u diskretnim vremenskim trenucima $t = nT_s$ gde je T_s period kojim je signal odabaran. Samim tim sinusni signal predstavljen u diskretnom domenu nije predstavljen kao funkcija vremena već broja odbirka pa važi:

$$x(n) = A\sin(2\pi f n T_s + \varphi) = A\sin\left(2\pi\frac{f}{f_s}n + \varphi\right) \qquad (2)$$

Gde f_s predstavlja frekvenciju kojom je signal odabran. Zbog toga je uobičajeno da se frekvencija sinusnog signala u diskretnom domenu izražava relativno u odnosu na frekvenciju odabiranja (tzv. Normalizovana frekvencija). Normalizovana frekvencija odgovara vrednosti $\frac{f}{f_s}$.

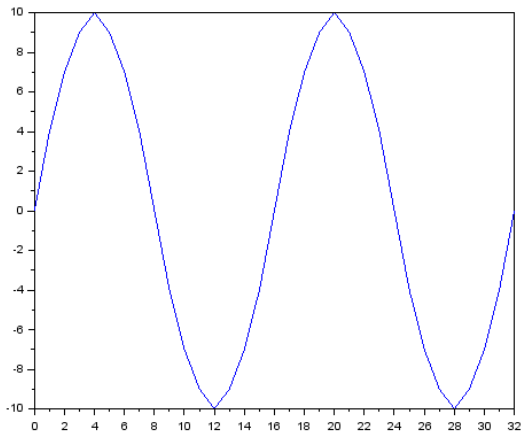
1.2.1 Predstava signala u programskom jeziku C

Kao što je već rečeno, signali u diskretnom domenu nisu predstavljeni kao neprekidna funkcija vremena, već kao nizovi diskretnih vrednosti (odbiraka). U okviru programskog jezika C za smeštanje odbiraka koriste se memorijski nizovi. Odbirci signala mogu biti predstavljeni različitim tipovima podataka (*int, double* i sl.). U okviru programskog jezika C, memorijski prostor za smeštanje niza podataka je moguće zauzeti statički ili dinamički. Statička alokacija memorije podrazumeva zauzimanje memorije u toku prevođenja programa, i na takav način memorija je zauzeta tokom čitavog trajanja izvršenja programa.

Statički, memorijski niz se deklariše dodavanjem operatora za indeksiranje niza nakon naziva promenljive "[ ]". Ukoliko se između zagrada nalazi celobrojna konstanta, taj broj predstavlja veličinu niza. U suprotnom veličinu niza proračunava kompajler na osnovu izraza za inicijalizaciju niza. Dat je primer definicije sinusoidnog signala predstavljenog sa celobrojnim vrednostima:

```
Int16 sine_wave[32] = { 0, 4, 7, 9, 10, 9, 7, 4,
                        0, -4, -7, -9, -10, -9, -7, -4,
                        0, 4, 7, 9, 10, 9, 7, 4,
                        0, -4, -7, -9, -10, -9, -7, -4 };
```

Definisani niz odgovara signalu prikazanom na slici 2.



Slika 2 - Prikaz signala koji odgovara nizu *sine_wave*

Dvodimenzionalni signali, poput signala slike, predstavljaju se kao dvodimenzionalni nizovi ili matrice.

```
Int16 niz2d[32][32]
```

Prilikom statičkog zauzimanja memorije kod TMS320C5535 arhitekture moguće je zadati poravnanje adrese početka zauzetog memorijskog niza korišćenjem *pragma* direktive za poravnanje, neposredno pre definicije promenljive:

- #pragma DATA_ALIGN(<naziv_promenljive>,<poravnanje>)

Kada za neku promenljivu kažemo da ima poravnanje N, to znači da se ta primenljiva nalazi u memorijskom prostoru na adresi koja je deljiva sa N.

Primer korišćenja pragma direktive za poravnanje:

```
#pragma DATA_ALIGN(sine_wave, 4)
Int16 sine_wave[32] = { 0, 4, 7, 9, 10, 9, 7, 4, …
```

1.3 Generisanje sinusnog signala

Rasprostranjena upotreba sinusa u mnogim aplikacijama i algoritmima koji se tiču digitalne obrade signala iziskuje čestu potrebu za generisanjem sinsusnog signala određene frekvencije, amplitude i faznog pomeraja.

1.3.1 Generisanje sinusnog signala korišćenjem ugrađene funkcije sin

Najprostiji način za implementaciju generatora sinusnog signala u programskom jeziku C jeste upotrebom funkcije *sin* iz standardnog zaglavlja sa matematičkim funkcijama *(math.h).* Funkcija *sin* kao parametar prima vrednost *x* koja predstavlja veličinu ugla u radijanima. Povratna vrednost ove funkcije jeste izračunata vrednost sinusa zadatog ugla.

- `double sin(double x);`

Generisanje sinusnog signala upotrebom ove funkcije vršimo tako što vrednost svakog odbirka računamo zasebno, na osnovu jednačine (2).

```
for (i=0; i<broj_odbiraka; i++)
{
    sine_wave[i] = amplituda*sin(2*PI*frekvencija*i+fazni_pomeraj);
}
```

Jedan od nedostataka ovakvog pristupa generisanju sinusnog signala jeste brzina izvršavanja. Implementacija standardne `sin` funkcije zavisi od arhitekture, samim tim i brzina njenog izvršavanja. S obzirom da se digitalna obrada signala najčešće vrši u okviru namenskih računarskih sistema u realnom vremenu, sa čvrstim vremenskim zahtevima brzina izvršavanja predstavlja bitan faktor prilikom implementacije bilo kog algoritma. Još jedna stavka koja zavisi od implementacije `sin` funkcije jeste tačnost generisanog signala odnosno prisutnost šuma u generisanom signalu. Ukoliko `sin` funkcija iz standardne biblioteke ne zadovoljava zahteve algoritma koji se implementira, za generisanje signala može se koristiti neka od postojećih tehnika.

1.3.2 Generisanje sinusnog signala upotrebom tabele pretraživanja

Metoda generisanja sinusnog signala upotrebom tabele pretraživanja (eng. *Lookup table*) podrazumeva čitanje unapred generisanih vrednosti odbiraka sinusnog signala iz memorije. Vrednosti signala koji se nalazi u memoriji dobijaju se odabiranjem analognog signala, ili su unapred izračunate pomoću određenog matematičkog algoritma. Tabela pretraživanja sadrži jednu periodu sinusnog signala predstavljenu sa *N* odbiraka, gde *N* ujedno predstavlja i veličinu tabele. Sinusni signal željene frekvencije

generišemo tako što čitamo vrednosti iz tabele pretraživanja sa konstantnim korakom *Δ* gde je *Δ* definisano sa:

$$\Delta = N\frac{f}{f_s} \tag{3}$$

Dakle, ukoliko želimo da generišemo sinusni signal od *L* odbiraka *x(l)*, gde je *i* = 0, 1, . . . , *L* − 1, to ćemo uraditi čitanjem podataka iz table pretraživanja sa:

$$x(i) = sineTable[k], k = (m + i * \Delta)_{modN} \tag{4}$$

U datoj jednačini *m* predstavlja zadati fazni pomeraj predstavljen sa brojem elemenata tabele pretraživanja. Fazni pomeraj predstavljen sa brojem elemenata i fazni pomeraj predstavljen u radijanima povezani su relacijom:

$$m = \varphi\frac{N}{2\pi} \tag{5}$$

Generisanje sinusnog signala korišćenjem tabele pretraživanja predstavlja brzu tehniku, jednostavnu za implementiranje, sa većom preciznošću od tehnika koje koriste matematičke aproksimacije, međutim i ona sama ima određena ograničenja.

Pitanje: Šta se dešava ukoliko rezultat izraza (3) nije ceo broj?

Ukoliko pokušavamo da generišemo signal frekvencije *f*, takve da na osnovu izraza (3) dobijemo vrednost koraka koja nije ceo broj, dolazi do greške prilikom odabiranja zbog činjenice da indeksiranje nizova možemo vršiti samo upotrebom celobrojnih vrednosti. Postoje načini da se estimira vrednost traženog odbirka. Najprostiji je da zaokružujemo vrednost indeksa na najbližu celobrojnu vrednost. Drugi, malo složeniji, jeste da vršimo interpolaciju susednih odbiraka. U svakom slučaju generisani signal će sadržati određenu grešku zbog ove estimacije.

Povećanjem tabele pretraživanja smanjuje se razlika između susednih odbiraka signala unutar tabele. Samim tim dobijamo i manju grešku estimacije. Naravno, pri tom se mora voditi računa i o memorijskim zahtevima.

Jedan od problema prilikom implementacije generatora sinusa koristeći tabelu pretraživanja, koji se odnosi na efikasnost samog rešenja, jeste implementacija uvećanja po modulu. U okviru programskog jezika C postoji operator za računanje ostatka pri deljenju „%“. Međutim, ovaj operator implementiran je upotrebom instrukcije deljenja, koja kod procesora sa redukovanim instrukcijskim setom najčešće nije prisutna, i njeno izvršenje zahteva veliki broj instrukcijskih ciklusa. Drugim rečima korišćenje ovog operatora rezultuje neefikasnim kodom.

Uvećanje broja po modulu moguće je efikasnije implementirati uvođenjem nekih dodatnih ograničenja. Jedan primer jeste korišćenje tabele pretraživanja čija je veličina stepen broja 2. Ukoliko veličina tabele pretraživanja predstavlja stepen broja 2, moguće je moduo operator zameniti operatorom logičkog „i“ („&“) sa maskom koja predstavlja maksimalnu moguću vrednost indeksa.

$$x \,\%\, N \;\equiv\; x \,\&\, (N-1)$$

Tokom implementacije generatora signala, prilikom izračunavanja indeksa za adresiranje tabele pretraživanja (*k,* jednačina (4)) neophodno je voditi računa o prekoračenju opsega. Za velike vrednosti *l i Δ,* proizvod *l*Δ* može da izlazi van opsega celobrojnog tipa. Kako bi se to izbeglo, umesto množenja *l*Δ* u svakoj iteraciji petlje, može se vršiti dodavanje vrednosti *Δ* na trenutnu vrednost *k,* gde pre početka petlje *k* ima početnu vrednost *m* (početni fazni pomeraj).

```
void gen_sinus_table(int n, float a, float f, float ph, float buffer[])
{
    int i = 0;
    int delta = f * N;
    int k = (ph/(2*PI)*N);
    int mask = (N-1);

    for (i = 0; i < n; i++)
    {
        k = k & mask;
        buffer[i] = a*p_sine_table[k];
        k+=delta;
    }
}
```

Na prikazanom primeru *N* predstavlja veličinu tabele pretraživanja za koju važi uslov da je $N=2^x$. Parametri funkcije su redom dužina generisanog signala, amplituda, normalizovana frekvencija, početni fazni pomeraj i niz za smeštanje generisanog signala.

S obzirom na periodičnu prirodu sinusnog signala, određenim modifikacijama algoritma, memorijske zahteve za čuvanje tabele pretraživanja možemo umanjiti 2 ili 4 puta. Za sinusni signal važe sledeće jednakosti za $0 \leq x < \frac{\pi}{2}$:

$$\begin{aligned} \sin\left(\frac{\pi}{2}+x\right) &= \sin\left(\frac{\pi}{2}-x\right) \\ \sin(\pi+x) &= -\sin(x) \\ \sin\left(\frac{3\pi}{2}+x\right) &= -\sin\left(\frac{\pi}{2}-x\right) \end{aligned} \tag{6}$$

Na osnovu ovih jednakosti vidimo da ukoliko znamo vrednosti sinusa u prvom kvadrantu, na osnovu tih vrednosti možemo izračunati vrednosti i ostatka signala.

1.3.3 Generisanje sinusnog signala upotrebom matematičkih aproksimacija

Postoji više rešenja za matematičku aproksimaciju vrednosti signala. Najčešće korišćene metode su:

- Rekurzivni metod – odbirak signala računamo na osnovu vrednosti prethodnih odbiraka, upotrebom:

$$H(z) = \frac{y(n)}{x(n)} = \frac{\sin \omega T \cdot z^{-1}}{1 - 2\cos \omega T \cdot z^{-1} + z^{-2}} \tag{7}$$

O ovakvom pristupu ćemo razgovarati više u sklopu digitalnih filtara sa beskonačnim impulsnim odzivom (eng. *IIR*).

- Aproksimacija funkcije sinusa razvojem u Tejlorov red:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots \tag{8}$$

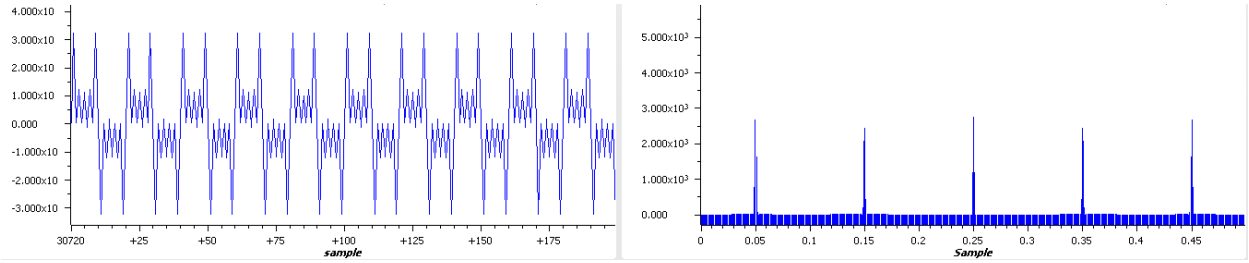
Ova aproksimacija najčešće je korišćena za implementaciju *sin* funkcije u okviru *math.h* zaglavlja standardne biblioteke. To je slučaj i sa standardnom bibliotekom za *Texas Instruments C55x* procesore.

1.4 Generisanje složenijih signala

Pored prostih sinusnih, postoji određeni broj složenijih signala koji se takođe vrlo često koriste u oblasti digitalne obrade signala.

1.4.1 Multiton

Multiton je signal koji se sastoji iz sume više sinusnih signala. Jedna varijacija multiton signala jeste signal koji sadrži frekvencije iz određenog opsega frekvencija (*f1, f2*), takve da su sve frekventne komponente signala ekvidistantne (rastojanje između dve susedne frekvencije jednako *Δf*).



Slika 3 - Multiton signal za f0=0.05, df=0.1 prikazan u vremenskom i frekventnom domenu

Generisanje odbiraka multiton signala vrši se tako što se generiše odbirak za svaku njegovu frekventnu komponentu, i potom izračuna suma generisanih odbiraka.

1.4.2 „frequency sweep“ signali

Još jedan tip signala koji se često koristi je tzv. *sweep* signal, koji predstavlja sinusni signal konstantne amplitude čija se frekvencija menja u vremenu. U zavisnosti od tipa promene frekvencije definiše se linearni *sweep* signal, čija se frekvencija menja linearno od *f1* do *f2* i logaritamski, čija se frekvencija menja eksponencijalno od *f1* do *f2*. U opštem slučaju, sinusni signal se može definisati kao:

$$x(t) = A\sin(\varphi(t)) \tag{9}$$

Gde je *φ(t)* trenutna faza. Trenutna frekvencija *f(t)* se definiše kao:

$$f(t) = \frac{1}{2\pi}\frac{\partial\varphi(t)}{\partial t}, \text{ odakle sledi } \varphi(t) = 2\pi \cdot \int f(t)dt. \tag{10}$$

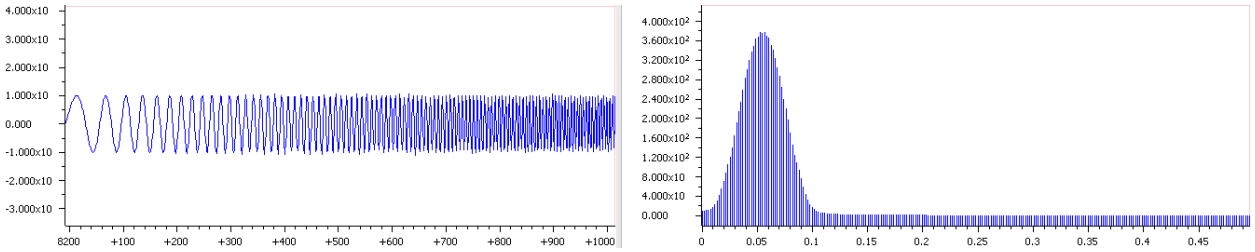
Za linearni *sweep* signal važi da je:

$$f(t) = f_1 + t\frac{f_2 - f_1}{T}, \text{ odakle } \varphi(t) = 2\pi\left(f_1 t + \frac{f_2 - f_1}{2T}t^2\right) + \varphi_0 \quad (11)$$

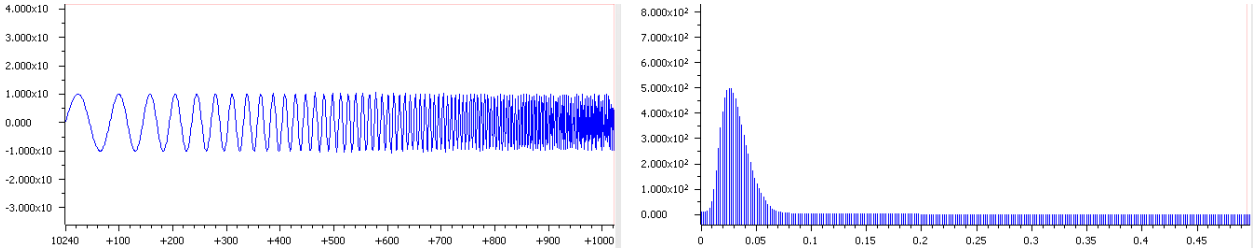
Gde je T dužina perioda kojom se generiše signal. Može se primetiti da važi da je: *f(0) = f_1* i *f(T) = f_2* .

Za logaritamski *sweep* signal važi da je:

$$f(t) = f_1 e^{\frac{t}{T}\ln\frac{f_2}{f_1}} \text{ odakle } \varphi(t) = 2\pi\frac{Tf_1}{\ln\frac{f_2}{f_1}}e^{\frac{t}{T}\ln\frac{f_2}{f_1}} + \varphi_0 \quad (12)$$



Slika 4 - Linearni "sweep" signal za f0=0.01 i f1=0.1



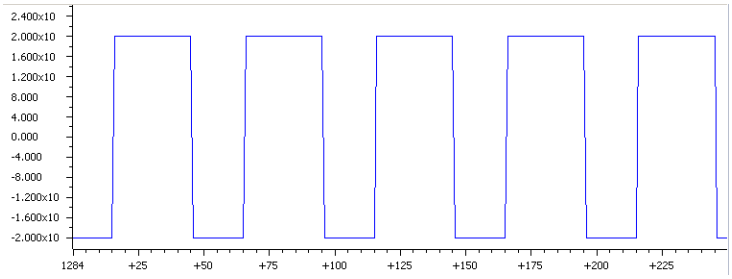
Slika 5 - Logaritamski "sweep" signal za f0=0.01 i f1=0.1

Implementacija generatora *sweep* signala sastoji se iz računanja trenutnog faznog pomeraja u trenutku u kome se generiše odbirak i računanja vrednosti sinusnog signala za zadati fazni pomeraj. Prikazana je implementacija generatora logarimtaskog *sweep* signala. Funkcije *exp, log* i *sin* predstavljaju funkcije standardne *math.h* biblioteke.

```
void gen_log_sweep(int n, float a, float f1, float f2, float ph, float buffer[])
{
    int i = 0;
    float ph_t = 0;
    for(i = 0; i < n; i++)
    {
        ph_t = (n*f1/log(f2/f1))*exp((i*log(f2/f1))/n)*2*PI+ph;
        buffer[i] = a*sin(ph_t);
    }
}
```

1.4.3 Periodični signal pravougaonih impulsa (*square* signal)

Periodični signal pravougaonih impulsa predstavlja signal u kome se ustaljenom frekvencijom smenjuju vrednosti amplitude između maksimalne i minimalne vrednosti.



Slika 1 - Povorka pravougaonih impulsa

Ovakav signal definisan je dužinom trajanja maksimalne vrednosti amplitude T_{max} i dužinom trajanja mininimalne vrednosti amplitude T_{min}. Perioda ovakvog signala definisana je zbirom ove dve vrednosti $T = Tmax + Tmin.$

Drugi oblik za predstavljanje istog signala jeste preko faktora *T* koji označava periodu signala i faktora ispunjenosti *D* koji označava procenat periode *T* kada signal ima maksimalnu vrednost amplitude.

$$D = \frac{Tmax}{T} * 100$$

Jedan od načina za generisanje povorke pravougaonih impulsa koristeći programski jezik C dat je na primeru ispod.

```
void gen_square(int n, float a, float f, float D, float buffer[])
{
    int i = 0, j=0;
    int T = 1/f;
    int Tmax = D*T/100;
    for(i = 0; i < n; i++)
    {
        if(j < Tmax)
        {
            buffer[i] = a;
        }
        else if(j < T)
        {
            buffer[i] = 0;
        }
        else
        {
            j = 0;
            buffer[i] = a;

        j++;
    }
}
```

2 ZADACI

Napomena: Očekivani izlaz iz svakog zadatka jeste prikaz generisanog signala. Za čuvanje koristiti *Print Screen* opciju i neki od programa za uređivanje slika (npr *MS Paint*). Generisane slike sačuvati u radnom direktorijumu pod nazivom Vezba2_*Ime_Prezime_BrojIndeksa*.

2.1 Zadatak 1

Uvući projekat Vezba 1 u radno okruženje.

Implementirati funkciju:

- `void gen_sinus(int n, float a, float f, float ph, float buffer[])`

koja generiše *n* odbiraka sinusnog signala amplitude a, frekvencije *f* i faznog pomeraja *ph* upotrebom *sin* funkcije iz standardne biblioteke. Rezultat smestiti u promenljivu *buffer.*
Pomoću realizovane funkcije generisati:

- Sinusni signal amplitude 10, frekvencije 0.02, faznog pomeraja 0 i trajanja 1000 odbiraka.
- Sinusni signal amplitude 5, frekvencije 0,05, faznog pomeraja π/2 i trajanja 1000 odbiraka.

Date signale prikazati u vremenskom i frekventnom domenu uz pomoć alata unutar *Code Composer Studio* razvojnog okruženja (*Tools->Graph*). Sačuvati *snapshot* i jednog i drugog grafika za oba signala.

Napomena: Za prikaz signala u vremenskom domenu koristiti alat *Single Time,* a za prikaz u frekventnom *FFT Magnitude.* I kod jednog I kod drugog količina podataka (*Acquisition Buffer Size*) treba da odgovara veličini vašeg signala (broju odbiraka). Polje *Dsp Data Type* postaviti na *32 bit floating point*. Polje *Starting Address* treba da sadrži adresu početka niza (vrednost ili naziv promenljive).
Za *Single Time* alat polje koje određuje koliko od pročitanih odbiraka želimo da prikažemo (*Display Data Size*) takođe treba da odgovara veličini signala, jer želimo da prikažemo čitav signal.
Za *FFT Magnitude,* polje *Data Plot Style* postaviti na *Bar. FFT Order* postaviti na 12. Ova podešavanja koristiti prilikom iscrtavanja svih signala.

2.2 Zadatak 2

Implementirati funkciju:

- `void gen_sinus_multiton(int n, float a, float f0, float df, float ph, float buffer[]);`

koja generiše multiton signal trajanja *n* odbiraka, koji se sastoji od sinusnih signala amplitude a, faznog pomeraja *ph* i frekvencija od f_0 do 0.5, pri čemu je udaljenost između susednih frekvencija *df*.

Pomoću realizovane funkcije gnenerisati:

- Multiton signal amplitude 10, koji sadrži frekvencije od 0.05 do 0.5 sa korakom 0.1, faznog pomeraja 0 i trajanja 1000 odbiraka.
- Multiton signal amplitude 1, koji sadrži frekvencije od 0.1 do 0.5 sa korakom 0.03, faznog pomeraja PI/2 i trajanja 1000 odbiraka.

Date signale prikazati u vremenskom i frekventnom domenu uz pomoć alata unutar *Code Composer Studio* razvojnog okruženja (*Tools->Graph*). Sačuvati *snapshot* i jednog i drugog grafika za sve signale.

2.3 Zadatak 3

Analizirati datu funkciju: **gen_log_sweep**

Pomoću date funkcije generisati:

- Logaritamski „*sweep*“ signal amplitude 5, koji sadrži frekvencije od 0.01 do 0.1, faznog pomeraja 0 i trajanja 1000 odbiraka.

Date signale prikazati u vremenskom i frekventnom domenu uz pomoć alata unutar *Code Composer Studio* razvojnog okruženja (*Tools->Graph*). Sačuvati *snapshot* i jednog i drugog grafika za sve signale.

Na osnovu date funkcije implementirati funkciju:

- `void gen_lin_sweep(int n, float a, float f1, float f2, float ph, float buffer[])`

koja generiše *n* odbiraka linearnog „*sweep*“ signala amplitude a, faznog pomeraja *ph* i frekvencije koja se linearno menja od f_1 do f_2.

Pomoću realizovane funkcije gnenerisati:

- Linearan „*sweep*“ signal amplitude 5, koji sadrži frekvencije od 0.01 do 0.1, faznog pomeraja 0 i trajanja 1000 odbiraka.

Date signale prikazati u vremenskom i frekventnom domenu uz pomoć alata unutar *Code Composer Studio* razvojnog okruženja (*Tools->Graph->Single Time* i *Tools->Graph->FFT Magnitude*). Sačuvati *snapshot* i jednog i drugog grafika za sve signale.

2.4 Zadatak 4

Data je funkcija:

- `void gen_sinus_table(int n, float a, float f, float ph, float buffer[])`

koja generiše *n* odbiraka sinusnog signala amplitude a, frekvencije *f* i faznog pomeraja *ph* korišćenjem tabele pretraživanja. Rezultat smestiti u promenljivu *buffer.*
Tabela pretraživanja data je u okviru datoteke *sin_table.c.*
Pomoću realizovane funkcije generisati:

- Sinusni signal amplitude 10, frekvencije 0.02, faznog pomeraja 0 i trajanja 1000 odbiraka.
- Sinusni signal amplitude 5, frekvencije 0,05, faznog pomeraja π/2 i trajanja 1000 odbiraka.

Date signale prikazati u vremenskom i frekventnom domenu uz pomoć alata unutar *Code Composer Studio* razvojnog okruženja (*Tools->Graph*). Sačuvati *snapshot* i jednog i drugog grafika za oba signala.
Uporediti generisane signale sa signalima iz prvog zadatka.

2.5 Zadatak 5

Data je funkcija:

- `void gen_square(int n, float a, float f, float D, float buffer[])`

koja generiše povorku od pravougaonih impulsa u trajanju od *n* odbiraka. Maksimalna amplituda signala zadata je sa *a,* normalizovana frekvencija signala sa *f*, a faktor ispunjenosti *D* izražen je u procentima. Rezultat smestiti u promenljivu *buffer.* Perioda signala se može izračunati na osnovu frekvencije, *T=1/f.*

Pomoću realizovane funkcije generisati i prikazati povorku pravougaonih impulsa trajanja 1000 odbiraka, frekvencije 0.05, faktora ispunjenosti 50.00 % i amplitude 10.

Generisani signal prikazati u vremenskom domenu uz pomoć alata unutar *Code Composer Studio* razvojnog okruženja (*Tools->Graph->Single Time*). Sačuvati *snapshot* grafika.

2.6 Zadatak 6

Modifikovati funkciju iz zadatka 4, tako da tabela pretraživanja sadrži samo vrednosti sinusnog signala u prvom kvadrantu.

Pomoću realizovane funkcije generisati:

- Sinusni signal amplitude 10, frekvencije 0.02, faznog pomeraja 0 i trajanja 1000 odbiraka.
- Sinusni signal amplitude 5, frekvencije 0,05, faznog pomeraja π/2 i trajanja 1000 odbiraka.

Generisani signali moraju biti identični signalima iz zadatka 4.

3 Zaključak

Ova vežba vas je upoznala sa nekim od osnovnih periodičnih signala koji se koriste u oblasti DSP-a. Upoznali ste se sa izgledom pomenutih signala u vremenskom i frekventnom domenu. Naučili ste kako je te signale moguće generisati upotrebom programskog jezika C. Uočene su određene mane i prednosti različitih pristupa generisanju sinusnih signala. Svi generisani signali u okviru ove vežbe bili su u diskretnom obliku. U naredne dve vežbe upoznaćete se sa detaljima prevođenja analognih signala u diskretni oblik i obrnuto.